



[MDK-3D]

Development Kit

L6021

User's Manual

Revision 1.0
1/15/2010

Copyright[©] 2005-2010. Socle Technology Corp. All Rights Reserved.

This document contains information that is confidential and proprietary to Socle Technology Corp. and may be disclosed only to those employees of Socle Technology with a need to know, or as otherwise permitted in writing by Socle Technology. Any copying, reproducing, modifying, use or disclosure of this information (in whole or in part) which is not expressly permitted in writing by Socle Technology is strictly prohibited. At a minimum, this information is protected under trade secret, unfair competition, and copyright laws. Violations thereof may result in criminal penalties and fines.

Socle Technology reserves the right to change the information contained in this document to improve function, design or otherwise. Socle Technology does not assume any liability arising out of the application or use of this information, or of any error or omission in such information. Any warranties, whether express, statutory, implied or otherwise, including but not limited to the implied warranties of merchantability or fitness for a particular purpose, are excluded. Any license under patent rights or any other intellectual property rights owned by Socle Technology or third parties shall be conveyed by Socle Technology in a separate license agreement between Socle Technology and the licensee.

Trademark

SoC_ImP[®], μPlatform[®] and the Socle logo, are the trademarks of Socle Technology. All other trademarks referred to herein are the property of their respective owners.

Revision History

Rev.	Date	Description	Author
1.0	2009/12/16	First Version	

TABLE OF CONTENT

CHAPTER 1	PRODUCT OVERVIEW.....	14
INTRODUCTION	14	
KEY FEATURES.....	14	
APPLICATION FIELD	19	
SYSTEM BLOCK DIAGRAM.....	20	
SYSTEM ARCHITECTURE	21	
PIN DESCRIPTIONS	22	
SYSTEM ADDRESS MAP.....	37	
INTERRUPT SOURCES	40	
CHAPTER 2	SYSTEM CONTROLLER.....	42
OVERVIEW	42	
KEY FEATURES.....	42	
ARCHITECTURE.....	43	
<i>Block Diagram.....</i>	43	
<i>Block Descriptions.....</i>	43	
REGISTERS	44	
<i>Registers Summary.....</i>	44	
<i>Detail Register Description</i>	45	
FUNCTIONAL DESCRIPTION.....	50	
<i>PLL0 Configuration (for AXI/AHB/APB System Clocks)</i>	50	
<i>PLL1 Configuration (for ARM1176JZF CPU Clock).....</i>	51	
<i>PLL2 Configuration (for ARM926EJ CPU Clock).....</i>	51	
<i>System Clock Generation (CPU/AXI/AHB/APB).....</i>	52	
<i>System Reset Generation.....</i>	52	
CHAPTER 3	DDR2 SDRAM CONTROLLER	53
OVERVIEW	53	
KEY FEATURES.....	53	
ARCHITECTURE.....	54	
<i>Block Diagram.....</i>	54	
<i>Block Descriptions.....</i>	54	
REGISTERS	55	
<i>Registers Summary.....</i>	55	

<i>Detail Register Description</i>	58
FUNCTIONAL DESCRIPTION	89
<i>DDR2 Initialization Sequence</i>	89
<i>Memory Mapping for Address Space</i>	90
<i>Multiple-Port Arbiter</i>	90
<i>Command Queue and Placement Logics</i>	95
CHAPTER 4 STATIC MEMORY CONTROLLER	101
OVERVIEW	101
KEY FEATURES.....	101
ARCHITECTURE.....	101
<i>Block Diagram</i>	101
<i>Block Descriptions</i>	101
REGISTERS	102
<i>Registers Summary</i>	102
<i>Detail Register Description</i>	103
FUNCTIONAL DESCRIPTION	105
<i>AHB read/ write from Flash/ROM</i>	105
CHAPTER 5 NAND FLASH CONTROLLER.....	107
OVERVIEW	107
KEY FEATURES.....	107
ARCHITECTURE.....	107
<i>Block Diagram</i>	107
<i>Block Descriptions</i>	108
REGISTERS	108
<i>Registers Summary</i>	108
<i>Detail Register Description</i>	111
FUNCTIONAL DESCRIPTION	127
POINTER CONTROL OF PAGE-PROGRAM OPERATION	127
NAND-FLASH CONTROLLER DATA ACCESS SEQUENCE	128
<i>Small Page (512bytes) NAND-Flash</i>	128
<i>Large Page (2048bytes) NAND-Flash</i>	129
<i>NAND Flash Reset Sequence</i>	129
<i>Page-Program Operation (Small Page Device)</i>	129
<i>Page-Program Operation (Large Page Device)</i>	130

<i>Page-Read</i>	130
<i>Read for Copy-Back</i>	131
<i>Program for Copy-Back</i>	131
<i>Block Erase</i>	132
<i>Read-ID</i>	132
CHAPTER 6 VECTORED INTERRUPT CONTROLLER	133
OVERVIEW	133
KEY FEATURES.....	133
ARCHITECTURE.....	134
<i>Block Diagram</i>	134
<i>Block Descriptions</i>	134
REGISTERS	135
<i>Registers Summary</i>	135
<i>Detail Register Description</i>	139
FUNCTIONAL DESCRIPTION	143
<i>Daisy-Chained Interrupt Scheme</i>	143
CHAPTER 7 DUAL-CORE COMMUNICATION UNIT	145
OVERVIEW	145
KEY FEATURES.....	145
ARCHITECTURE.....	146
<i>Block Diagram</i>	146
<i>Block Descriptions</i>	146
REGISTERS	147
<i>Registers Summary</i>	147
<i>Detail Register Description</i>	149
FUNCTIONAL DESCRIPTION	161
<i>Mailbox Programming Sequence</i>	161
<i>ECT Programming Sequence</i>	162
CHAPTER 8 DMA CONTROLLER	163
OVERVIEW	163
KEY FEATURES.....	163
ARCHITECTURE.....	164
<i>Block Diagram</i>	164

<i>Block Descriptions</i>	164
REGISTERS	164
<i>Registers Summary</i>	164
<i>Detail Register Description</i>	166
FUNCTIONAL DESCRIPTION	173
<i>S/W Trigger DMA Mode</i>	173
<i>H/W Trigger DMA Mode</i>	173
<i>H/W Trigger DMA with Slice Mode</i>	173
CHAPTER 9 LCD CONTROLLER	175
OVERVIEW	175
KEY FEATURES.....	175
ARCHITECTURE.....	176
<i>Block Diagram</i>	176
<i>Block Descriptions</i>	176
REGISTERS	179
<i>Register Summary</i>	179
<i>Detail Register Description</i>	180
FUNCTIONAL DESCRIPTION	191
<i>Operation</i>	191
CHAPTER 10 CCIR-656 VIP CONTROLLER	193
OVERVIEW	193
KEY FEATURES.....	193
ARCHITECTURE.....	193
<i>Block Diagram</i>	193
<i>Block Descriptions</i>	194
REGISTERS	194
<i>Registers Summary</i>	194
<i>Detail Register Description</i>	195
FUNCTIONAL DESCRIPTION	200
<i>Operation</i>	200
CHAPTER 11 3D/2D GRAPHICS ACCELERATOR.....	203
OVERVIEW	203
KEY FEATURES.....	203

ARCHITECTURE.....	204
<i>Block Diagram</i>	204
<i>Block Descriptions</i>	204
REGISTERS	205
<i>Registers Summary</i>	207
<i>Detail Register Description</i>	215
FUNCTIONAL DESCRIPTION.....	270
<i>Geometry Processor Vertex Shader</i>	270
<i>Geometry Processor Polygon List Builder</i>	273
<i>Pixel Processor Fragment Shader</i>	274
<i>Memory Management</i>	275
<i>Configuration interfaces</i>	278
<i>Interrupts</i>	278
<i>Performance counters</i>	278
<i>Watchdog Timers</i>	279
CHAPTER 12 ETHERNET 10/100M MAC CONTROLLER.....	280
OVERVIEW	280
KEY FEATURES.....	280
ARCHITECTURE.....	281
<i>Block Diagram</i>	281
<i>Block Descriptions</i>	281
REGISTERS	283
<i>Registers Summary</i>	283
<i>Detail Register Description</i>	283
FUNCTIONAL DESCRIPTION.....	296
<i>Descriptors/buffers architecture overview</i>	296
<i>Receive Descriptors</i>	298
<i>Transmit Descriptors</i>	302
<i>Setup Frames</i>	305
<i>DMA Controllers</i>	307
<i>Transmit Process</i>	308
<i>Receive Process</i>	309
<i>Interrupt Control</i>	310
<i>Collision Handling</i>	311

<i>Deferring</i>	313
<i>Receive address filtering</i>	314
<i>General purpose timer</i>	315
CHAPTER 13 USB2.0 HS OTG.....	317
OVERVIEW	317
KEY FEATURES.....	317
ARCHITECTURE.....	318
<i>Block Diagram</i>	318
<i>Block Descriptions</i>	318
REGISTERS	319
<i>Registers Summary</i>	319
<i>Detail Register Description</i>	320
FUNCTIONAL DESCRIPTION.....	358
<i>Host Mode Operation</i>	358
<i>Device Mode Operation</i>	360
<i>OTG Mode Operation</i>	379
CHAPTER 14 SD/MMC/SDIO HOST CONTROLLER.....	382
OVERVIEW	382
KEY FEATURES.....	382
ARCHITECTURE.....	382
<i>Block Diagram</i>	382
<i>Block Descriptions</i>	382
REGISTERS	383
<i>Registers Summary</i>	384
<i>Detail Register Description</i>	385
FUNCTIONAL DESCRIPTION.....	418
<i>Clock and Reset</i>	418
<i>SD Transaction Generation</i>	418
<i>Error Recovery</i>	428
<i>Suspend/Resume Mechanism</i>	429
<i>Multi-Slot Operation</i>	431
<i>DMA Operation</i>	431
CHAPTER 15 UART (16550)	439

OVERVIEW	439
KEY FEATURES.....	439
ARCHITECTURE.....	440
<i>Block Diagram</i>	440
<i>Block Descriptions</i>	440
REGISTERS	440
<i>Registers Summary</i>	440
<i>Detail Register Description</i>	441
FUNCTIONAL DESCRIPTION	447
<i>Clock Signals</i>	447
<i>Operation</i>	447
CHAPTER 16 GENERAL PURPOSE IO (GPIO)	449
OVERVIEW	449
KEY FEATURES.....	449
ARCHITECTURE.....	449
<i>Block Diagram</i>	449
<i>Block Descriptions</i>	449
REGISTERS	450
<i>Registers Summary</i>	450
<i>Detail Register Description</i>	450
FUNCTIONAL DESCRIPTION	453
<i>Operation</i>	453
<i>Programming Sequence</i>	453
CHAPTER 17 INTERNAL TIMERS	459
OVERVIEW	459
KEY FEATURES.....	459
ARCHITECTURE.....	460
<i>Block Diagram</i>	460
<i>Block Descriptions</i>	460
REGISTERS	460
<i>Registers Summary</i>	460
<i>Detail Register Description</i>	460
FUNCTIONAL DESCRIPTION	461
<i>Operation</i>	461

CHAPTER 18	PWM TIMER	463
OVERVIEW		463
<i>Key Features</i>		463
ARCHITECTURE.....		463
<i>Block Diagram</i>		463
<i>Block Descriptions</i>		463
REGISTERS		464
<i>Registers Summary</i>		464
<i>Detail Register Description</i>		464
FUNCTION DESCRIPTION		465
<i>Operation</i>		465
<i>Pre-Scale function</i>		466
<i>Interrupt Feature</i>		466
CHAPTER 19	WATCHDOG TIMER (WDT)	467
OVERVIEW		467
KEY FEATURES.....		467
ARCHITECTURE.....		468
<i>Block Diagram</i>		468
<i>Block Descriptions</i>		468
REGISTERS		468
<i>Registers Summary</i>		468
<i>Detail Register Description</i>		469
FUNCTIONAL DESCRIPTION		469
<i>Operation</i>		469
CHAPTER 20	REAL TIME CLOCK (RTC).....	471
OVERVIEW		471
KEY FEATURES.....		471
ARCHITECTURE.....		472
<i>Block Diagram</i>		472
<i>Block Descriptions</i>		472
REGISTERS		473
<i>Registers Summary</i>		473
<i>Detail Register Description</i>		473

FUNCTIONAL DESCRIPTION	477
<i>Operation</i>	477
<i>Programming sequence</i>	479
CHAPTER 21 SPI CONTROLLER	481
OVERVIEW	481
KEY FEATURES.....	481
ARCHITECTURE.....	482
<i>Block Diagram</i>	482
<i>Block Descriptions</i>	482
REGISTERS	483
<i>Registers Summary</i>	483
<i>Detail Register Description</i>	484
FUNCTIONAL DESCRIPTION	494
<i>Operation</i>	494
CHAPTER 22 I2C CONTROLLER	499
OVERVIEW	499
KEY FEATURES.....	499
ARCHITECTURE.....	499
<i>Block Diagram</i>	499
<i>Block Descriptions</i>	500
REGISTERS	501
<i>Registers Summary</i>	501
<i>Detail Register Description</i>	501
FUNCTIONAL DESCRIPTION	505
<i>Operation</i>	505
PROGRAMMING SEQUENCE.....	511
CHAPTER 23 I2S CONTROLLER.....	516
OVERVIEW	516
KEY FEATURES.....	516
ARCHITECTURE.....	516
<i>Block Diagram</i>	516
<i>Block Descriptions</i>	517
REGISTERS	518

<i>Registers Summary</i>	518
<i>Detail Register Description</i>	518
FUNCTIONAL DESCRIPTION	522
<i>Operation</i>	522
<i>Programming sequence</i>	524
CHAPTER 24 MECHANICAL DATA	527
PACKAGE DIMENSIONS.....	527

Chapter 1 Product Overview

Introduction

L6021 is a 16/32-bit RISC microprocessor, which is designed to provide high performance application Processor solution for multimedia and general applications. It built-in both ARM1176JZF and ARM926EJ processor core, and could active either one processor core when system operation.

Key Features

- **ARM1176JZF Processor**

- 32/16-bit RISC architecture (ARMv6)
- 32-bit ARM instruction set for maximum performance and flexibility
- 16-bit Thumb instruction set for increased code density
- An 8-stage pipeline
- Branch prediction and return stack
- Integrated Vector Floating Point coprocessor
- Vectored interrupt interface speeds interrupt response
- MMU, which supports OS including Symbian OS, Windows CE, Linux
- Integrated 4-way set associate 16KB I-cache and 16KB D-cache
- Integrated 64KB I-TCM and 64KB D-TCM

- **ARM926EJ Processor**

- 32/16-bit RISC architecture (ARMv5)
- 32-bit ARM instruction set for maximum performance and flexibility
- 16-bit Thumb instruction set for increased code density
- An 5-stage pipeline
- DSP instruction extensions and single cycle MAC
- MMU, which supports OS including Symbian OS, Windows CE, Linux
- Integrated 4-way set associate 16KB I-cache and 16KB D-cache
- Integrated 64KB I-TCM and 64KB D-TCM

- **Memory Controller (NOR-Flash/NAND-Flash/DDR2)**

- Support NOR-Flash/NAND-Flash/DDR2 access
- Support 2 Flash bank and 1 DDR2 bank

- Support 8, and 16-bit wide NOR-Flash from 2Mbyte to 64Mbyte
 - Support 16, and 32-bit wide DDR2 from 32Mbyteit to 256Mbyte
 - Support from 128Mbyte to 1Gbyte SLC/MLC NAND-Flash
 - Provide hardware ECC circuitry with 6 symbol correct capability per 512-bytes for NAND-Flash
 - Programmable DDR2 and NOR-Flash timing parameters
- **System Controller**
 - Provide system decoder Remap function
 - System clock and reset control
 - System power mode control
 - **Vectored Interrupt Controller**
 - Supports for 40 vectored IRQ interrupts
 - Fixed hardware interrupt priority levels
 - Programmable interrupt priority levels
 - Hardware interrupt priority level masking
 - IRQ and FIQ generation
 - Support for ARM1176JZF processor VIC port in synchronous mode, enabling faster interrupt servicing
 - **USB2.0 HS OTG**
 - 3-channel USB2.0 HS OTG
 - Complies with the OTG specification supplement to the USB 2.0 protocol
 - Compliant with INTEL EHCI-TT specification in host mode
 - Programs as USB2.0 device only, USB2.0 host only, and OTG
 - Supports high speed (480Mbps), full speed (12Mbps) and low-speed (1.5Mbps)
 - 4 endpoints for USB device
 - On-chip transceiver
 - **10/100M Ethernet MAC**
 - 2-channel 10/100M Ethernet MAC
 - Media Independent Interface (MII)
 - Support 10/100M bps transfer data rate
 - Support full or half duplex operation

- Support 16 flexible physical address filtering, and 512-bit hash table for multicast address
- **TFT LCD Controller**
 - Support for color TFT panel
 - Resolution programmable up to 1280x1024
 - 16, 18, 24-bit bpp for color TFT display
 - Hardware YUV to RGB convert support (YUV 4:2:2 or 4:2:0)
 - Color indexing through color LUT with 8-bit index
 - Programmable timing for different display panels
- **CCIR-656 VIP (Camera Interface)**
 - Fully compatible with 8-bit CCIR-656
 - Support NTSC and PAL format
 - Support YCbCr 4:2:2 and 4:2:0 format
 - Embedded H & V sync
 - Support up to D1 (720x480) resolution, free format in interlaced mode, 30fps
- **3D/2D Graphics Accelerator (Mali-200)**
 - Compatible with OpenGL ES 2.0, OpenGL ES 1.1, and OpenVG 1.0
 - Alpha blending
 - 4 times and 16 times Full Scene Anti-Aliasing (FSAA)
 - Programmable fragment shader
 - Programmable vertex shader
 - Flexible input and output formats
 - Indexed and non-indexed geometry input
- **Dual-Core Communication Unit (Mailbox+CTM)**
 - Each mailbox element includes a data word, a command word and a flag bit that could represent an interrupt
 - 1 interrupts to built-in ARM1176JZF or ARM926EJ interrupt controller
 - 1 interrupts to client interrupt controller
 - Support daisy-chain debugging on ARM Multi-ICE through EJTAG interface
- **DMA Controller**
 - 4-channel DMA controller

- Support memory-to-memory, IO-to-memory, memory-to-IO transfer
- **General Purpose Input/Output**
 - 16 individually programmable input/output pins
- **SPI Interface**
 - 4-channel SPI with interrupt-based operation
 - Support master or slave mode
 - Four transfer protocols available with selectable clock polarity phase
 - Programmable output clock frequency
 - Full duplex synchronous serial data transfer
- **I2C-Bus Interface**
 - 3-channel I2C
 - Compatible with I2C specification v2.1
 - Support Master or Slave device mode of I2C bus
 - Support multi-master operation
 - Support 7-bit or 10-bit address mode operation
 - Programmable clock frequency and transfer rate up to 400Kbps
- **I2S-Bus Interface**
 - 1-channel with DMA-based or interrupt-based operation I2S for audio interface
 - Support 8, 16, 20, or 24-bits audio resolution
 - Support audio sample rate from 32 to 96 KHz
 - Support I2S digital serial audio data interface
 - Support full duplex serial data transfer
- **SD/MMC/SDIO Host**
 - 2-channel SD with DMA-based or interrupt-based operation
 - Compliant with SD Memory Card spec. version 2.0 (including SDHC)
 - Compliant with SDIO Host spec. version 2.0
 - Compliant with MMC Card version 4.2 (including MMCplus)
 - SD1/SD4/MMC-8 modes of operation
 - Suspend/Resume mechanism for SDIO cards
 - Read Wait mechanism for SDIO cards

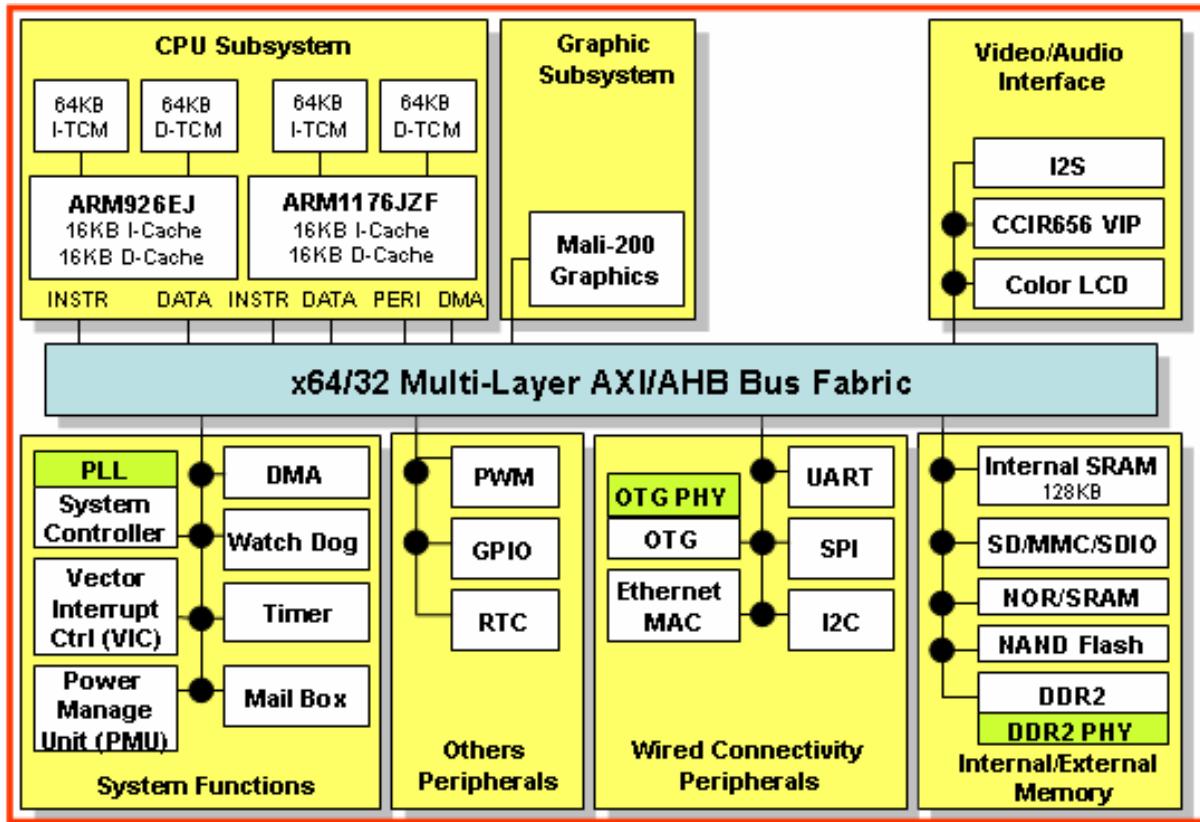
- Programmable output clock frequency
- **UART**
 - 3-channel UART with interrupt-based operation
 - Each channel has internal 16-byte TX FIFO and 16-byte RX FIFO
 - Programmable baud rate generator, up to 115.2bps
 - False start bit detection
 - Line break generation and detection
- **Timer**
 - 4-channel 32-bit timer with Pulse Width Modulation (PWM) / 3-channel 32-bit internal timer
 - Programmable duty-cycle, and frequency output
- **RTC (Real Time Clock)**
 - 24-hour time mode with highest precision of tenth of a second
 - 32.768KHz operation
 - Alarm interrupt
- **AMBA Expansion**
 - 1-channel asynchronous 64-bit AXI master interface
 - 1-channel asynchronous 64-bit AXI slave interface
 - 1-channel asynchronous 32-bit AHB master interface
 - 1-channel asynchronous 32-bit AHB slave interface
- **Operating Voltage**
 - 1.0 volt for core
 - 3.3/2.5/1.8 volt for I/O
- **Operating Temperature**
 - 0°C ~ 70°C
- **Operating Frequency**
 - ARM1176JZF : Up to 1000MHz
 - ARM926EJ : Up to 800MHz
 - AXI Bus : Up to 400MHz

- AHB Bus: Up to 200MHz
 - APB Bus: Up to 50MHz
-
- **Package**
 - PBGA1380 (42.5mm x 42.5mm)

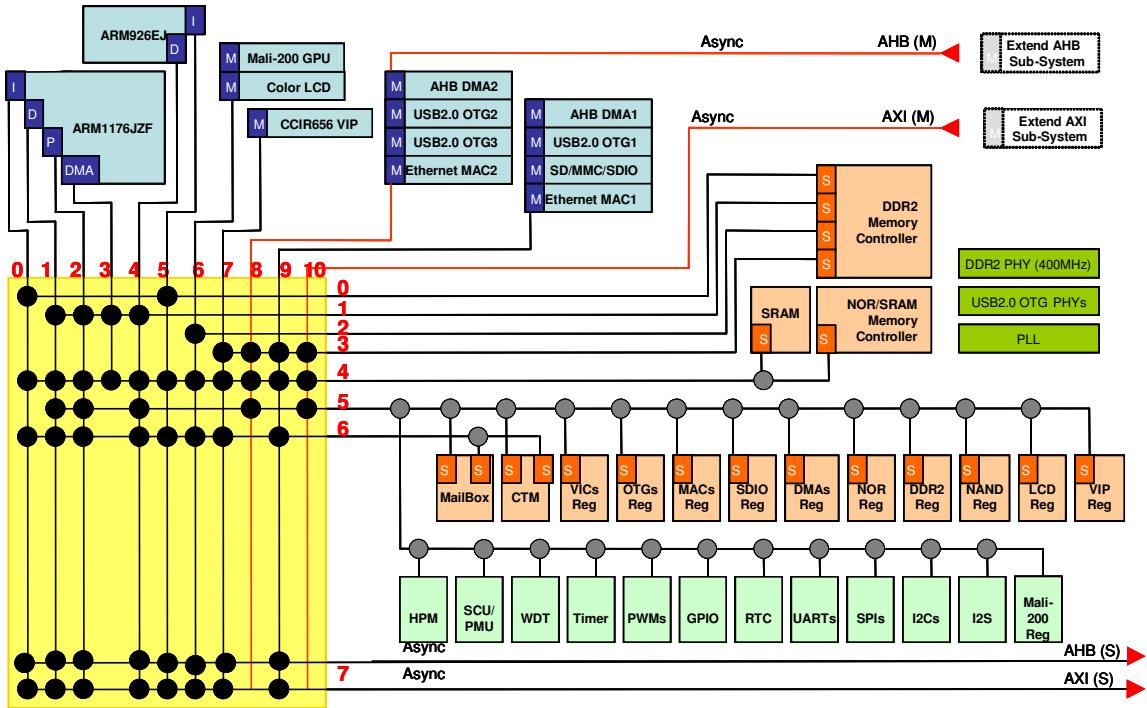
Application Field

- Mobile Internet Device (MID)
- Smartbook (Netbook)
- Thin Client
- Home Multimedia Center (HMC)
- Dual-Core System (ARM926EJ+DSP or ARM1176JZF+DSP)

System Block Diagram



System Architecture



Pin Descriptions

Signal	I/O	Description
System Control		
CPU_SEL[1:0]	I	<p>Active CPU select.</p> <p>00: Internal ARM1176JZF active.</p> <p>01: Internal ARM926EJ active.</p> <p>10: Reserved.</p> <p>11: All internal CPUs inactive.</p>
DEBUG_SEL[1:0]	I	<p>CPU JTEG debug mode section.</p> <p>00: Debug embedded ARM1176JZF or ARM926EJ core only</p> <p>01: Reserved.</p> <p>10: Debug both embedded and external processors.</p> <p>-> Embedded processor -> External processor.</p> <p>11: Debug external processor only.</p>
PLL0_CONFIG[2:0]	I	<p>PLL0 output clock configuration, when CLKPLL0 is 32MHz.</p> <p>PLL0 is the clock source of AXI/AHB/APB bus clock.</p> <p>AXI clock = 1/2 PLL0 clock; AHB clock = 1/4 PLL0 clock; APB clock = 1/8 PLL0 clock.</p> <p>000: 336MHz 001: 400MHz 010: 528MHz 011: 664MHz 100: 800MHz 101: 928MHz 110: 1000MHz 111: 1056MHz</p>
PLL1_CONFIG[2:0]	I	PLL1 output clock configuration, when CLKPLL1 is 32MHz.

		PLL1 is the clock source of ARM1176JZF CPU clock.
	000: 336MHz	
	001: 400MHz	
	010: 528MHz	
	011: 664MHz	
	100: 800MHz	
	101: 928MHz	
	110: 1000MHz	
	111: 1056MHz	
PLL2_CONFIG[2:0]	I	PLL2 output clock configuration, when CLKPLL2 is 32MHz.
		PLL2 is the clock source of ARM926EJ CPU clock.
	000: 336MHz	
	001: 400MHz	
	010: 528MHz	
	011: 664MHz	
	100: 800MHz	
	101: 928MHz	
	110: 1000MHz	
	111: 1056MHz	
CLKEXT_CONFIG[1:0]	I	External ACLK/HCLK bus clock output (CLKEXTOUT) configuration.
	00: Divide by 1	
	01: Divide by 2	
	10: Divide by 4	
	11: Divide by 8	
AHRESET_SEL	I	AXI and AHB bus reset generation mode.
	0: Hardware reset, external bus reset is the same as internal bus one.	
	1: Software reset, for dual-core sequential booting.	
PLL0BP	I	PLL0 bypass mode, active HIGH.
PLL1BP	I	PLL1 bypass mode, active HIGH.

PLL2BP	I	PLL2 bypass mode, active HIGH.
POR_N	I(ST)	External power-on-reset, active LOW.
EXTRESET_N	O	AXI and AHB bus reset for external device, active LOW.
CLKPLL0 (AXI/AHB/APB)	I(ST)	(individual) Clock for PLL0, 32MHz.
CLKPLL1 (ARM1176JZF)	I(ST)	(Individual) Clock for PLL1, 32MHz.
CLKPLL2 (ARM926EJ)	I(ST)	(Individual) Clock for PLL2, 32MHz.
CLKEXTIN	I(ST)	(Individual) Clock for asynchronous AXI and AHB bus extension, 66MHz.
CLKRTC	I(ST)	(Individual) Clock for RTC, 32.768KHz.
CLKUART	I(ST)	(Individual) Clock for UART, 24MHz.
CLKI2S	I(ST)	(Individual) Clock for I2S, 33.8688MHz.
CLKLCD	I(ST)	(Individual) Clock for LCD, 28MHz.
CLKVIP	I(ST)	(Individual) Clock for VIP, 24 MHz.
CLKEXTOUT	O	AXI/AHB bus clock for external device.
CLKOTG0	I(ST)	(Individual) Clock for OTG PHY0, 12MHz.
CLKOTG1	I(ST)	(Shared) Clock for OTG PHY1, 12MHz.
CLKOTG2	I(ST)	(Shared) Clock for OTG PHY2, 12MHz.
CLKSDM	I(ST)	(Individual) Clock for SD/MMC/SDIO, 50MHz.
REGPWR_ON	O	External regulator power on/off. Need to add pull-high resistor to avoid power up fail. 0: Regulator power off. 1: Regulator power on.

Interrupt Interface

NIRQ	O	nIRQ output from built-in VIC, active LOW.
NFIQ	O	nFIQ output from built-in VIC, active LOW.
EXT_INT[5:0]	I	External interrupt input.
MAC0_INT	O	MAC0 interrupt output.

MAC1_INT	O	MAC1 interrupt output.
TIMER0_INT	O	Timer0 interrupt output.
TIMER1_INT	O	Timer1 interrupt output.
TIMER2_INT	O	Timer2 interrupt output.
PWM0_INT	O	PWM0 interrupt output.
PWM1_INT	O	PWM1 interrupt output.
PWM2_INT	O	PWM2 interrupt output.
PWM3_INT	O	PWM3 interrupt output.
RTC_INT	O	RTC interrupt output.
UART0_INT	O	UART0 interrupt output.
UART1_INT	O	UART1 interrupt output.
UART2_INT	O	UART2 interrupt output.
SPI0_INT	O	SPI0 interrupt output.
SPI1_INT	O	SPI1 interrupt output.
I2C0_INT	O	I2C0 interrupt output.
I2C1_INT	O	I2C1 interrupt output.
I2C2_INT	O	I2C2 interrupt output.
I2S_INT	O	IIS interrupt output.
OTG0_INT	O	OTG0 interrupt output.
OTG1_INT	O	OTG1 interrupt output.
OTG2_INT	O	OTG2 interrupt output.
GPIO_INT	O	GPIO interrupt output.
HDMA0_INT	O	HDMA0 interrupt output.
HDMA1_INT	O	HDMA1 interrupt output.
NFC_INT	O	NAND flash controller interrupt output.
VIP_INT	O	VIP interrupt output.
CLCD_INT	O	Color CLD controller interrupt output.
MBOX0_INT	O	Mail box interrupt output, trigger by external 2 nd processor.
MBOX1_INT	O	Mail box interrupt output, trigger by built-in ARM processor.
DDR2_INT	O	DDR2 interrupt output.
SD_INT	O	SD2.0 / SDIO module 0 interrupt output
M200_INT	O	Mali200 interrupt output.
MPG2_INT	O	MaliGP2 interrupt output.

MMU_INT	O	Mali MMU interrupt output.
AXI Interface		
AXI Master Interface		
AWADDR_M[31:0]	I	AXI write address bus.
AWVALID_M	I	AXI write address valid.
AWREADY_M	O	AXI write address ready.
AWLEN_M[3:0]	I	AXI write address burst size.
AWSIZE_M[2:0]	I	AXI write address transfer size.
AWBURST_M[1:0]	I	AXI write address burst type.
AWLOCK_M[1:0]	I	AXI write address lock type.
AWCACHE_M[3:0]	I	AXI write address cache attribute.
AWPROT_M[2:0]	I	AXI write address protection level.
AWID_M[4:0]	I	AXI write address ID.
ARADDR_M[31:0]	I	AXI read address bus.
ARREADY_M	O	AXI read address ready.
ARVALID_M	I	AXI read address valid.
ARLEN_M[3:0]	I	AXI read address burst size.
ARSIZE_M[2:0]	I	AXI read address transfer size.
ARBURST_M[1:0]	I	AXI read address burst type.
ARLOCK_M[1:0]	I	AXI read address lock type.
ARCACHE_M[3:0]	I	AXI read address cache attribute.
ARPROT_M[2:0]	I	AXI read address protection level.
ARID_M[4:0]	I	AXI read address ID.
WLAST_M	I	AXI write data last.
WDATA_M[63:0]	I	AXI write data.
WVALID_M	I	AXI write data valid.
WREADY_M	O	AXI write data ready.
WSTRB_M[7:0]	I	AXI write data strobe.
WID_M[4:0]	I	AXI write data ID.
RDATA_M[63:0]	O	AXI read data.
RVALID_M	O	AXI read data valid.
RREADY_M	I	AXI read data ready.
RID_M[4:0]	O	AXI read data ID.
RRESP_M[1:0]	O	AXI read response.
RLAST_M	O	AXI read last.

BREADY_M	I	AXI write response ready.
BVALID_M	O	AXI write response valid.
BRESP_M[1:0]	O	AXI write response.
BID_M[4:0]	O	AXI write response ID.
AXI Slave Interface		
AWADDR_S[31:0]	O	AXI write address bus.
AWVALID_S	O	AXI write address valid.
AWREADY_S	I	AXI write address ready.
AWLEN_S[3:0]	O	AXI write address burst size.
AWSIZE_S[2:0]	O	AXI write address transfer size.
AWBURST_S[1:0]	O	AXI write address burst type.
AWLOCK_S[1:0]	O	AXI write address lock type.
AWCACHE_S[3:0]	O	AXI write address cache attribute.
AWPROT_S[2:0]	O	AXI write address protection level.
AWID_S[8:0]	O	AXI write address ID.
ARADDR_S[31:0]	O	AXI read address bus.
ARREADY_S	I	AXI read address ready.
ARVALID_S	O	AXI read address valid.
ARLEN_S[3:0]	O	AXI read address burst size.
ARSIZE_S[2:0]	O	AXI read address transfer size.
ARBURST_S[1:0]	O	AXI read address burst type.
ARLOCK_S[1:0]	O	AXI read address lock type.
ARCACHE_S[3:0]	O	AXI read address cache attribute.
ARPROT_S[2:0]	O	AXI read address protection level.
ARID_S[8:0]	O	AXI read address ID.
WLAST_S	O	AXI write data last.
WDATA_S[63:0]	O	AXI write data.
WVALID_S	O	AXI write data valid.
WREADY_S	I	AXI write data ready.
WSTRB_S[7:0]	O	AXI write data strobe.
WID_S[8:0]	O	AXI write data ID.
RDATA_S[63:0]	I	AXI read data.
RVALID_S	I	AXI read data valid.
RREADY_S	O	Read data ready.
RID_S[8:0]	I	Read data ID.

RRESP_S[1:0]	I	Read response.
RLAST_S	I	Read last.
BREADY_S	O	Write response ready.
BVALID_S	I	Write response valid.
BRESP_S[1:0]	I	Write response.
BID_S[8:0]	I	Write response ID.
AHB Interface		
HADDR[31:0]	I/O	AHB address bus.
HTRANS[1:0]	I/O	AHB htrans bus.
HBURST[2:0]	I/O	AHB hburst.
HWRITE	I/O	AHB read/write indicator.
HSIZE[1:0]	I/O	AHB hsize.
HDATA[31:0]	I/O	AHB read/write data bus.
HREADY	I/O	AHB hready.
HRESP[1:0]	I	AHB hresp.
HBUSREQ	I	AHB hbusreq from external AHB master.
HLOCK	O	AHB hlock from external AHB master.
HGRANT	O	AHB hgrant to external AHB master.
HSEL	O	AHB hsel to external AHB slave.
DDR2 Memory Interface		
DDR2_CLK	O	DDR2 clock.
DDR2_CLKN	O	DDR2 inverted clock.
DDR2_ADDR[14:0]	O	DDR2 address bus.
DDR2_DATA[31:0]	I/O	DDR2 data bus.
DDR2_BA[2:0]	O	DDR2 bank address.
DDR2_CASN	O	DDR2 column address strobe, active LOW.
DDR2_RASN	O	DDR2 row address strobe, active LOW.
DDR2_CKE	O	DDR2 clock enable, active HIGH.
DDR2_CSN	O	DDR2 chip select, active LOW.
DDR2_WEN	O	DDR2 write enable, active LOW.
DDR2_DM[3:0]	O	DDR2 data mask, active HIGH.
DDR2_DQS[3:0]	I/O	DDR2 data strobe. Edge-aligned with read data, center-aligned with write data.
DDR2_DQSN[3:0]	I/O	Inverted DDR2 data strobe. DDR2_DQSN are only used

DDR2_ODT	O	DDR2 on-die termination enable, active HIGH.
DDR2_FTUNE[3:0]	O	DDR2 timing strobe window fine-tune. To add 5~15pF capacitor in this pin for timing fine-tune.
OTG 0/1/2 Interface		
OTG0_VBUS	AI/AO	OTG0 VBUS 5V.
OTG0_DP	AI/AO	OTG0 positive output channel.
OTG0_DM	AI/AO	OTG0 negative output channel.
OTG0_ID	AI/AO	OTG0 ID pin of mini-AB receptacle.
OTG0_RREFEXT	AI/AO	OTG0 external resistor connection for current reference.
OTG0_DRVVBUS	O	OTG0 drive enable of 5V VBUS, active HIGH.
OTG1_VBUS	AI/AO	OTG1 VBUS 5V.
OTG1_DP	AI/AO	OTG1 positive output channel.
OTG1_DM	AI/AO	OTG1 negative output channel.
OTG1_ID	AI/AO	OTG1 ID pin of mini-AB receptacle.
OTG1_RREFEXT	AI/AO	OTG1 external resistor connection for current reference.
OTG1_DRVVBUS	O	OTG1 drive enable of 5V VBUS, active HIGH.
OTG2_VBUS	AI/AO	OTG2 VBUS 5V.
OTG2_DP	AI/AO	OTG2 positive output channel.
OTG2_DM	AI/AO	OTG2 negative output channel.
OTG2_ID	AI/AO	OTG2 ID pin of mini-AB receptacle.
OTG2_RREFEXT	AI/AO	OTG2 external resistor connection for current reference.
OTG2_DRVVBUS	O	OTG2 drive enable of 5V VBUS, active HIGH.
MAC0/1 Interface		
MII0_MDC	O	MII0 management data clock.

MII0_MDIO	I/O	MII0 management data
MII0_TXEN	O	MII0 transmit enable.
MII0_TXER	O	MII0 transmit error.
MII0_TXD[3:0]	O	MII0 transmit data.
MII0_COL	I	MII0 collision status.
MII0_TXCLK	I(ST)	MII0 TX clock
MII0_RXCLK	I(ST)	MII0 RX clock
MII0_RXER	I	MII0 receive error.
MII0_RXDV	I	MII0 receive data valid.
MII0_CRS	I	MII0 carrier sense valid.
MII0_RXD[3:0]	I	MII receive data.
MII1_MDC	O	MII1 management data clock.
MII1_MDIO	I/O	MII1 management data
MII1_TXEN	O	MII1 transmit enable.
MII1_TXER	O	MII1 transmit error.
MII1_TXD[3:0]	O	MII1 transmit data.
MII1_COL	I	MII1 collision status.
MII1_TXCLK	I(ST)	MII1 TX clock
MII1_RXCLK	I(ST)	MII1 RX clock
MII1_RXER	I	MII1 receive error.
MII1_RXDV	I	MII1 receive data valid.
MII1_CRS	I	MII1 carrier sense valid.
MII1_RXD[3:0]	I	MII1 receive data.
LCD Interface		
LCD_POWER	O	LCD panel power enables.
LCD_CP	O	LCD panel clock.
LCD_DATA[23:0]	O	LCD panel data.
		24-bpp: for LCD-M0 and LCD-M1
		R[7:0] = LCD_DATA[7:0]
		G[7:0] = LCD_DATA[15:8]
		B[7:0] = LCD_DATA[23:16]
		16-bpp: for LCD-M0
		R[7:3] = LCD_DATA[7:3]
		G[7:2] = LCD_DATA[15:10]
		B[7:3] = LCD_DATA[23:19]

16-bpp: for LCD-M1		
		R[7:3] = LCD_DATA[6:2]
		G[7:2] = LCD_DATA[1],
		LCD_DATA[11:7]
		B[7:3] = LCD_DATA[17:13]
LCD_LP	O	TFT horizontal synchronization pulse.
LCD_FP	O	TFT vertical synchronization pulse.
LCD_AC	O	TFT data enable output.
UART Interface		
UART0_TXD	O	UART0 transmit serial data output.
UART0_RXD	I	UART0 receive serial data input.
UART1_TXD	O	UART1 transmit serial data output.
UART1_RXD	I	UART1 receive serial data input.
UART2_TXD	O	UART2 transmit serial data output.
UART2_RXD	I	UART2 receive serial data input.
UART2_NCTS	I	UART2 clear to send modem status input.
UART2_NRTS	O	UART2 request to send modem status output.
SPI Interface		
SPI0_SCK	I/O	SPI0 serial clock.
SPI0_SS[1:0]	I/O	SPI0 slave select, active LOW.
SPI0_MOSI	I/O	SPI0 serial data output.
SPI0_MISO	I/O	SPI0 serial data input.
SPI1_SCK	I/O	SPI1 serial clock.
SPI1_SS[1:0]	I/O	SPI1 slave select, active LOW.
SPI1_MOSI	I/O	SPI1 serial data output.
SPI1_MISO	I/O	SPI1 serial data input.
I2C Interface		
I2C0_SCL	I/O	I2C0 serial clock.
I2C0_SDA	I/O	I2C0 serial data.
I2C1_SCL	I/O	I2C1 serial clock.
I2C1_SDA	I/O	I2C1 serial data.
I2C2_SCL	I/O	I2C2 serial clock.
I2C2_SDA	I/O	I2C2 serial data.
I2S Interface		

I2S_TXSCLK	I/O	I2S transmitter serial clock.
I2S_RXSCLK	I/O	I2S receiver serial clock input.
I2S_TXLRCK	I/O	I2S transmitter left/right channel select.
I2S_RXLRCK	I/O	I2S receiver left/right channel select.
I2S_SDO	O	I2S serial data output.
I2S_SDI	I	I2S serial data input.
SD Card 0 Interface		
SDC0_CLK	O	SD memory card clock.
SDC0_PWR	O	SD memory card power enable signal, active HIGH.
SDC0_DATA[7:0]	I/O	SD memory card data bus.
SDC0_CMD	I/O	SD memory card command line.
SDC0_WP	I	SD memory card write protect enable, active HIGH.
SDC0_CD	I	SD memory card detect signal, active HIGH.
SD Card 1 Interface		
SDC1_CLK	O	SDC_CLK
SDC1_PWR	O	SD memory card power enable signal, active HIGH.
SDC1_DATA[7:0]	I/O	SDC_DATA[7:0]
SDC1_CMD	I/O	SDC_CMD
SDC1_WP	I	SDC_WP
SDC1_CD	I	SDC_CD
JTAG Interface		
TCK	I(ST)	ICE JTAG test clock for debug.
TRSTN	I(ST)	ICE JTAG test reset for debug.
TDI	I	ICE JTAG test serial data input for debug.
TMS	I	ICE JTAG test mode select for debug.
TDO	O	ICE JTAG test serial data out for debug.
RTCK	O	ICE JTAG return test clock.
CPU1_TDI	O	JTAG test serial data input for external 2 nd processor debug.
CPU1_TDO	I	JTAG test serial data out for external 2 nd processor debug.

CPU1_RTCK	I(ST)	JTAG return test clock for external 2 nd processor debug.
CPU1_DBGREQ	O	Force external 2 nd processor into debug state, active HIGH.
CPU1_DBGACK	I	Acknowledge external 2 nd processor is in debug state, active HIGH.
NAND Flash Interface		
NF_RBN[3:0]	I	NAND-Flash ready/busy indicator, HIGH for ready.
NF_WPN	O	NAND-Flash write protect, active LOW.
NF_WEN	O	NAND-Flash write enable, active LOW.
NF_REN	O	NAND-Flash read enable, active LOW.
NF_CEN[3:0]	O	NAND-Flash chip enable, active LOW.
NF_ALE	O	NAND-Flash address latch enable, active HIGH.
NF_CLE	O	NAND-Flash command latch enable, active HIGH.
NF_IO[7:0]	I/O	NAND-Flash command, address, and data bus.
NOR Flash Interface		
ST_ROMSZ	I	Static memory data bus size. 0: 8-bit boot mode. 1: 16-bit boot mode.
ST_CSN[1:0]	O	Static memory chip selects
ST_OEN	O	Static memory output enable signal indicates memory read
ST_WEN	O	Static memory write enable, indicates memory write
ST_ADDR[26:0]	O	Memory address for static memory.
ST_DQ[15:0]	IO	Static memory data bus.
PWM Interface		
PWM[3:0]	IO	PWM output in reference mode, and input in capture mode.
GPIO Interface		
GPIO[15:0]	IO	General purpose I/O.

VIP Interface

VIP_DATA[7:0]

I

Video data bus.

Power and Ground

VDD1

P

1.0V core power.

(Shut down by REGPWR_ON request)

VDD2

P

3.3V I/O power.

(Shut down by REGPWR_ON request)

VDD3

P

1.8V SSTL-18 I/O power for DDR2.

(Shut down by REGPWR_ON request)

VDD4

P

3.3V analog power for USB2.0 HS OTG PHY0.

(Shut down by REGPWR_ON request)

VDD5

P

2.5V analog power for USB2.0 HS OTG PHY0.

(Shut down by REGPWR_ON request)

VDD6

P

3.3V analog power for USB2.0 HS OTG PHY1.

(Shut down by REGPWR_ON request)

VDD7

P

2.5V analog power for USB2.0 HS OTG PHY1.

(Shut down by REGPWR_ON request)

VDD8

P

3.3V analog power for USB2.0 HS OTG PHY2.

(Shut down by REGPWR_ON request)

VDD9

P

2.5V analog power for USB2.0 HS OTG PHY2.

(Shut down by REGPWR_ON request)

VDD10

P

2.5V analog power for PLL0.

(Shut down by REGPWR_ON request)

VDD11

P

2.5V analog power for PLL1.

(Shut down by REGPWR_ON request)

VDD12

P

2.5V analog power for PLL2.

(Shut down by REGPWR_ON request)

VDD13

P

1.0V core power for RTC.

(Always on power)

VDD14	P	3.3V I/O power for RTC. (Always on power)
VDD15	P	2.5V I/O power for OTG PHY0, PHY1, PHY2 (Shut down by REGPWR_ON request)
VDD16	P	1.0V core power for USB2.0 HS OTG PHY0 (Shut down by REGPWR_ON request)
VDD17	P	1.0V core power for USB2.0 HS OTG PHY1 (Shut down by REGPWR_ON request)
VDD18	P	1.0V core power for USB2.0 HS OTG PHY2 (Shut down by REGPWR_ON request)
Vref	P	0.9V Vref power for DDR2 (Shut down by REGPWR_ON request)
VDD19	G	2.5V/3.3V I/O Power for AXI/AHB extension (Shut down by REGPWR_ON request)
GND1	G	Core ground.
GND2	G	I/O ground.
GND3	G	SSTL-18 I/O ground for DDR2.
GND4	G	Analog ground for USB2.0 HS OTG PHY0.
GND5	G	Analog ground for USB2.0 HS OTG PHY1.
GND6	G	Analog ground for USB2.0 HS OTG PHY2.
GND7	G	Analog ground for PLL0.
GND8	G	Analog ground for PLL1.
GND9	G	Analog ground for PLL2.
GND10	G	Core ground for RTC.
GND11	G	I/O ground for RTC.
GND12	G	I/O ground for USB2.0 OTG PHY0, PHY1, PHY2.

GND13	G	Core ground for USB2.0 HS OTG PHY0
GND14	G	Core ground for USB2.0 HS OTG PHY1
GND15	G	Core ground for USB2.0 HS OTG PHY2
GND16	G	2.5V/3.3V IO Ground for AXI/AHB extension

Note:

1. I/O stands for input/output
2. AI/AO standard for analog input/output
3. ST stands for Schmitt trigger
4. P stands for power
5. G stands for ground

System Address Map

The 32-bit address bus can address up to 4GB of memory. This space is sub-divided into a numbers of memory banks and control registers. L6021 uses little-endian only for memory and registers access.

L6021 provides 2 banks of NOR Flash memory and 1 bank of DDR2 memory. It also provides address decode remap function. It cans speed-up whole system performance. The detail memory map is as follow.

Memory System Block Diagram

	Remap=0	Remap=1	Remap=2
0xFFFF_FFFF	Reserved	Reserved	Reserved
0x6000_0000			
0x5FFF_FFFF	Special Function	Special Function	Special Function
0x5000_0000	Register	Register	Register
0x4FFF_FFFF	DDR2	DDR2	DDR2
0x4000_0000			
0x3FFF_FFFF	External AXI	External AXI	External AXI
0x3000_0000			
0x2FFF_FFFF	External AHB	External AHB	External AHB
0x2000_0000			
0x1FFF_FFFF	Reserved	Reserved	Reserved
0x1802_0000			
0x1801_FFFF	Internal SRAM	Internal SRAM	Internal SRAM
0x1800_0000			
0x17FF_FFFF	Flash1	Flash1	Flash1
0x1400_0000			
0x13FF_FFFF	Flash0	Flash0	Flash0
0x1000_0000			
0x0FFF_FFFF	Flash0	DDR2	Internal SRAM
0x0000_0000			

Device Specific Memory Map

Address		Size (MB)	Description	Note
0x0000_0000	0x0FFF_FFFF	256	Remap0: Flash0 Remap1: DDR2 Remap2: Internal SRAM	Mirrored Region
0x1000_0000	0x13FF_FFFF	64	Flash0	
0x1400_0000	0x17FF_FFFF	64	Flash1	
0x1800_0000	0x1801_FFFF	0.128	Internal SRAM	
0x1802_0000	0x1FFF_FFFF	-	Reserved	
0x2000_0000	0x2FFF_FFFF	256	External AHB	
0x3000_0000	0x3FFF_FFFF	256	External AXI	
0x4000_0000	0x4FFF_FFFF	256	DDR2	

AHB Specific Function Registers

Address		Size (KB)	Description	Note
0x5000_0000	0x5001_FFFF	128	DDR2 Controller	
0x5002_0000	0x5003_FFFF	128	NOR Flash Controller	
0x5004_0000	0x5005_FFFF	128	NAND Flash Controller	
0x5006_0000	0x5007_FFFF	128	DMA0	
0x5008_0000	0x5009_FFFF	128	DMA1	
0x500A_0000	0x500B_FFFF	128	Mailbox for ARM CPU	
0x500C_0000	0x500D_FFFF	128	Mailbox for 2nd Processor	
0x500E_0000	0x500F_FFFF	128	CTM for ARM CPU	
0x5010_0000	0x5011_FFFF	128	CTM for 2nd Processor	
0x5012_0000	0x5013_FFFF	128	USB OTG	

			Controller 0
0x5014_0000	0x5015_FFFF	128	USB OTG
			Controller 1
0x5016_0000	0x5017_FFFF	128	USB OTG
			Controller 2
0x5018_0000	0x5019_FFFF	128	MAC Controller 0
0x501A_0000	0x501B_FFFF	128	MAC Controller 1
0x501C_0000	0x501D_FFFF	128	LCD Controller
0x501E_0000	0x501F_FFFF	128	VIP Controller
0x5020_0000	0x5021_FFFF	128	SD2.0/MMC/SDIO
0x5022_0000	0x5023_FFFF	128	Interrupt To CPU
			Controller0
0x5024_0000	0x5025_FFFF	128	Interrupt Cascaded
			Controller1
0x5026_0000	0x50FF_FFFF	-	Reserved

APB Specific Function Registers

Address		Size (KB)	Description	Note
0x5100_0000	0x5100_0FFF	4	Watch Dog Timer	
0x5100_1000	0x5100_1FFF	4	GP Timers	
0x5100_2000	0x5100_2FFF	4	SCU	
0x5100_3000	0x5100_3FFF	4	PWM Timers	
0x5100_4000	0x5100_4FFF	4	RTC	
0x5100_5000	0x5100_5FFF	4	GPIO	
0x5100_6000	0x5100_6FFF	4	UART0	
0x5100_7000	0x5100_7FFF	4	UART1	
0x5100_8000	0x5100_8FFF	4	UART2	
0x5100_9000	0x5100_9FFF	4	SPI0	
0x5100_A000	0x5100_AFFF	4	SPI1	
0x5100_B000	0x5100_BFFF	4	I2C0	
0x5100_C000	0x5100_CFFF	4	I2C1	
0x5100_D000	0x5100_DFFF	4	I2C2	
0x5100_E000	0x5100_EFFF	4	I2S	

0x5100_F000	0x5100_FFFF	4	PMU	
0x5101_0000	0x5101_3FFF	16	Mali-200 GPU (Mali200/MaliGP2/MaliMMU)	
0x5101_4000	0x5101_4FFF	4	HPM0	400MHz
0x5101_5000	0x5101_5FFF	4	HPM1	400MHz
0x5101_6000	0x5101_6FFF	4	HPM2	400MHz
0x5101_7000	0x5101_7FFF	4	HPM3	400MHz
0x5101_8000	0x5101_8FFF	4	HPM4	400MHz
0x5101_9FFF	0x5101_FFFF	-	Reserved	
0x5102_0000	0x5102_0FFF	4	HPM5	200MHz
0x5102_1000	0x5102_1FFF	4	HPM6	200MHz
0x5102_2000	0x5FFF_FFFF	-	Reserved	

Interrupt Sources

L6021 supports 40 interrupt sources as shown in the below table

Channel #	Interrupt Source Description
39	Mali-200 MMU interrupt
38	Mali-200 MaliGP2 interrupt
37	Mali-200 Mali200 interrupt
36	External 5 interrupt
35	External 4 interrupt
34	External 3 interrupt
33	External 2 interrupt
32	External 1 interrupt
31	External 0 interrupt
30	I2C 2 interrupt
29	I2C 1 interrupt
28	I2C 0 interrupt
27	SD/MMC/SDIO interrupt
26	I2S interrupt
25	SPI 1 interrupt
24	SPI 0 interrupt
23	UART 2 interrupt
22	UART 1 interrupt

21	UART 0 interrupt
20	GPIO interrupt
19	RTC interrupt
18	PWM 3 interrupt
17	PWM 2 interrupt
16	PWM 1 interrupt
15	PWM 0 interrupt
14	Timer 2 interrupt
13	Timer 1 interrupt
12	Timer 0 interrupt
11	VIP controller interrupt
10	LCD controller interrupt
9	MAC controller 1 interrupt
8	MAC controller 0 interrupt
7	USB OTG controller 2 interrupt
6	USB OTG controller 1 interrupt
5	USB OTG controller 0 interrupt
4	DDR2 memory controller interrupt
3	Mailbox interrupt triggered by external 2 nd processor
2	DMA1 interrupt
1	DMA0 interrupt
0	NAND Flash controller interrupt

Chapter 2 System Controller

Overview

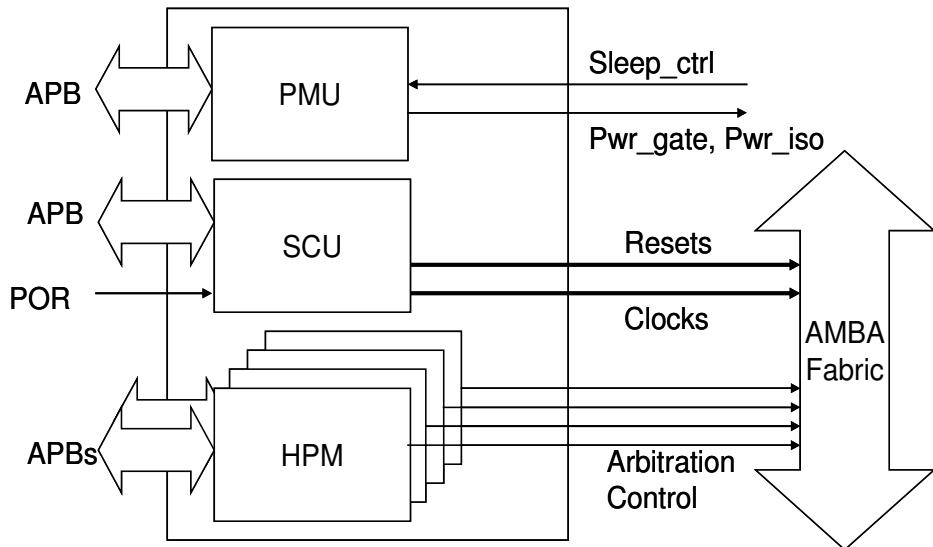
System controller is for generating all of the internal and system clocks and resets signals. Besides, it also provides a memory remap function after boot for ARM processor, power mode management, and AMBA bus fabric matrix control. This makes the system have better performance in memory usage, power management and bus traffic arrangement.

Key Features

- PLL0/1/2 configuration.
- System clock and reset generation.
- Memory space remap for ARM series processor.
- System power mode control. (PMU)
- AMBA Bus fabric matrix control. (HPM)
 - Fixed Round-Robin (RR) scheme
 - Programmable RR scheme
 - Programmable scheme that provides prioritized groups of Least Recently Granted (LRG) arbitration
 - Programmable Quality of Service (QoS) scheme

Architecture

Block Diagram



Block Descriptions

System Control Unit (SCU)

System clock and reset generation and remap control

Power Management Unit (PMU)

Power state control includes sleep and wakeup mechanism.

High-Performance Matrix (HPMs)

An arbitration mechanism that you can configure for each MI, implementing:

- A fixed Round-Robin (RR) scheme
- A programmable RR scheme
- A programmable scheme that provides prioritized groups of Least Recently Granted (LRG) arbitration.
- A programmable Quality of Service (QoS) scheme

This HPM functionality is performed by ARM's primecell PL301 IP. There is more detailed function description in "High Performance Matrix Technical Summary" from ARM.

Registers

This section describes the control/status registers of the design.

Registers Summary

System Control Unit (SCU)

Name	Offset	Size	Reset Value	Description
SCU_LPID	0x0000	W	0x6491000A	Leopard chip ID
SCU_REMAP	0x0004	W	0x00000000	Decoder remap control register
SCU_ERSTCTRL	0x0008	W	0x00000003	Software reset for external AXI and AHB bus
SCU_IRSTCTRL	0x000C	W	0x00000000	Software reset for internal blocks

Power Management Unit (PMU)

Name	Offset	Size	Reset Value (by POR_N)	Description
PMU_PWRMD	0x0000	W	0x00000000	Power mode
PMU_PWRCFG	0x0004	W	0x00000000	Power management configuration
PMU_WKUPCNT	0x0008	W	0x00000001	Wakeup reset counter
PMU_WKUPSTAT	0x000C	W	0x00000000	Wakeup trigger source
PMU_ALONRST	0x0010	W	0x00000001	Software reset for always-on blocks
PMU_INFORM0	0x0014	W	0x00000000	User define register
PMU_INFORM1	0x0018	W	0x00000000	User define register
PMU_PLL_RSTCNT	0x0024	W	0x00000000	PLL reset counter register

High-Performance Matrix (HPMx) (x=0~6)

Name	Offset	Size	Reset Value (by POR_N)	Description
HPMx_QOSTIDE	0x0400	W	0x00000000	QoS tidemark for MI 0
HPMx_QOSACC	0x0404	W	0x00000000	QoS access control for MI 0
HPMx_ARCAV	0x0408	W	0x00000000	AR channel arbitration value for MI 0

HPMx_AWCAV	0x040C	W	0x00000000	AW channel arbitration value for MI 0
------------	--------	---	------------	---------------------------------------

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

SCU_LDID

Address: Operational Base + offset (0x00)

Leopard chip ID

Bit	Attr	Reset Value	Description
31:0	R	0x6491_000A	L6021 chip ID

SCU_REMAP

Address: Operational Base + offset (0x04)

Decoder remap control register

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Decoder remap function for 0x0000_0000 to 0x0FFF_FFFF memory space. 0x00: Flash0. (when system booting) 0x01: DDR2. 0x02: Internal SRAM. 0x03: Reserved.

SCU_ERSTCTRL

Address: Operational Base + offset (0x08)

External AXI and AHB bus reset control register, when **AHRESET_SEL** pin setting is HIGH.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1	RW	0x0	Software reset generation for external AXI bus. 0: De-assert ARESET_N signal. 1: Assert ARESET_N signal.
0	RW	0x0	Software reset generation for external AHB bus.

			0: De-assert HRESET_N signal. 1: Assert HRESET_N signal.
--	--	--	---

SCU_IRSTCTRL

Address: Operational Base + offset (0x0C)

Internal software reset control register for internal blocks.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1	RW	0x0	Software reset generation for CTM block. 0: De-assert software signal. 1: Assert software signal.
0	RW	0x0	Software reset generation for mailbox block. 0: De-assert software signal. 1: Assert software signal.

PMU_PWRMD

Address: Operational Base + offset (0x00)

This register contains the power mode control bit.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0000	SLEEP Mode Control. When write “0x2bed” to this bit, then the system enters into SLEEP mode.

PMU_PWRCFG

Address: Operational Base + offset (0x04)

This register contains the power management configuration bit.

Bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0x0	Mask external interrupt 5 wake-up source. 0: No mask 1: Masked
5	RW	0x0	Mask external interrupt 4 wake-up source. 0: No mask 1: Masked
4	RW	0x0	Mask external interrupt 3 wake-up source.

			0: No mask 1: Masked
3	RW	0x0	Mask external interrupt 2 wake-up source. 0: No mask 1: Masked
2	RW	0x0	Mask external interrupt 1 wake-up source. 0: No mask 1: Masked
1	RW	0x0	Mask external interrupt 0 wake-up source. 0: No mask 1: Masked
0	RW	0x0	Mask RTC alarm interrupt wake-up source. 0: No mask 1: Masked

PMU_WKUPCNT

Address: Operational Base + offset (0x08)

This register contains the value of reset duration counter. These values must be bigger than 0.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0001	Force reset signal active during the external power source settle down when a wake-up procedure from SLEEP mode is being started. (Count by 32KHz clock)

PMU_WKUPSTAT

Address: Operational Base + offset (0x0C)

This register is contains the trigger source for exiting SLEEP mode.

Bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0x0	Wake-up by external interrupt 5. This bit is cleared by write “1”.
5	RW	0x0	Wake-up by external interrupt 4. This bit is cleared by write “1”.
4	RW	0x0	Wake-up by external interrupt 3. This bit is cleared by write “1”.
3	RW	0x0	Wake-up by external interrupt 2. This bit is cleared by write “1”.
2	RW	0x0	Wake-up by external interrupt 1.

			This bit is cleared by write “1”.
1	RW	0x0	Wake-up by external interrupt 0. This bit is cleared by write “1”.
0	RW	0x0	Wake-up by RTC alarm. This bit is cleared by write “1”.

PMU_ALONRST

Address: Operational Base + offset (0x10)

This register contains the power mode control bit.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x1	Software reset generation for RTC block. 0: De-assert software signal. 1: Assert software signal.

PMU_INFORM0

Address: Operational Base + offset (0x14)

User defined information register.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	User defined information register.

PMU_INFORM1

Address: Operational Base + offset (0x18)

User defined information register.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	User defined information register.

PMU_PLL_RSTCNT

Address: Operational Base + offset (0x24)

PLL reset counter register.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	PLL reset counter register.

HPMx_QOSTIDE (x=0~6)

Address: Operational Base + offset (0x400)

QoS tidemark for MI 0 register.

If a value is written to this register that is larger than the combined acceptance capability of the attached slave, then the QoS scheme never becomes active for this MI. If a value of 0 is written to this register, then the QoS scheme is turned off for this MI. This behavior ensures that it is impossible to block all transactions completely by accidental mis-programming

Bit	Attr	Reset Value	Description
31:0	RW	0x0	QoS tidemark Register.

HPMx_QOSACC (x=0~6)

Address: Operational Base + offset (0x404)

QoS access control for MI 0 register.

A 1 in any bit of this register indicates that the SI corresponding to the bit position is permitted to use the reserved slots of the connected combined acceptance capability of the slaves.

The maximum value that you can write to this register is $2^{<\text{total number of SIs}>} - 1$. Changes to these values occur on the first possible arbitration time after they are written.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	QoS access control Register.

HPMx_ARCAV (x=0~6)

Address: Operational Base + offset (0x408)

AR channel arbitration value for MI 0 register.

The arbitration schemes for the AR channel and AW channel are set when you configure the HPM, and they control the arbitration scheme for these channels when the HPM exits from reset. However, the HPM enables you to change the AR channel and AW channel arbitration schemes by using the APB SI on the HPM to write to the arbitration control registers.

Bit	Attr	Reset Value	Description
31:24	RW	-	W: Slave interface number for write. R: Reserved.

23:16	RW	-	Reserved (Set to zero).
15:8	RW	0x0	Interface priority value for the arbitration scheme. The range of values permitted is from 0 to 255 where: 0: Highest priority 255: Lowest priority.
7:0	RW	-	W: Set to zero. R: Slave interface number for read.

HPMx_AWCAV (x=0~6)

Address: Operational Base + offset (0x40C)

AW channel arbitration value for MI 0 register.

The arbitration schemes for the AR channel and AW channel are set when you configure the HPM, and they control the arbitration scheme for these channels when the HPM exits from reset. However, the HPM enables you to change the AR channel and AW channel arbitration schemes by using the APB SI on the HPM to write to the arbitration control registers.

Bit	Attr	Reset Value	Description
31:24	RW	-	W: Slave interface number for write. R: Reserved.
23:16	RW	-	Reserved (Set to zero).
15:8	RW	0x0	Interface priority value for the arbitration scheme. The range of values permitted is from 0 to 255 where: 0: Highest priority 255: Lowest priority.
7:0	RW	-	W: Set to zero. R: Slave interface number for read.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

PLL0 Configuration (for AXI/AHB/APB System Clocks)

PLL_CON FIG	PLLOUT (MHz)	AXI (MHz)	AHB (MHz)	APB (MHz)	FIN (MHz)	DIVF[7:0]	DIVR[5:0]	DIVQ[2:0]	RANGE[2:0]
000	332	166	83	21	32	82	1	2	1

001	400	200	100	25	32	99	1	2	1
010	528	264	132	33	32	65	1	1	1
011	664	332	166	41.5	32	82	1	1	1
100	800	400	200	50	32	99	1	1	1
101	928	469	235	59	32	115	1	1	1
110	1000	500	250	62.5	32	124	1	1	1
111	1056	528	264	66.5	32	65	1	0	1

PLL1 Configuration (for ARM1176JZF CPU Clock)

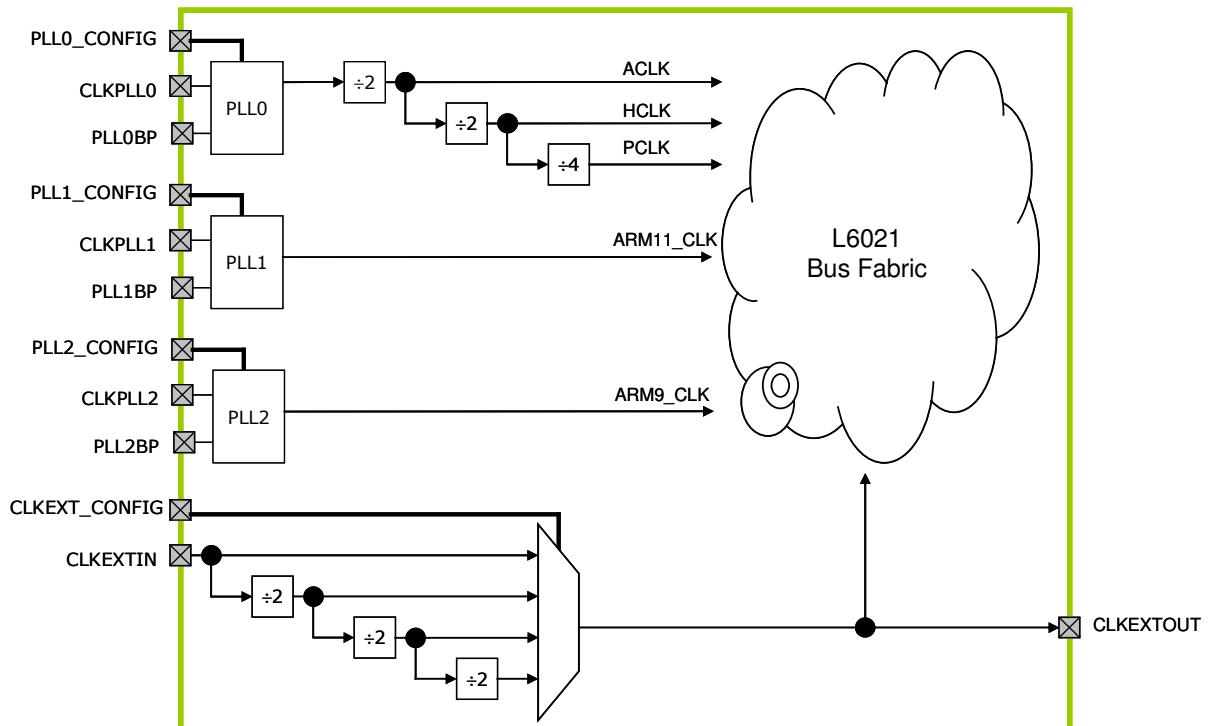
PLL_CON FIG	PLLOUT (MHz)	ARM1176JZF (MHz)	FIN (MHz)	DIVF[7:0]	DIVR[5:0]	DIVQ[2:0]	RANGE[2:0]
000	332	332	32	82	1	2	1
001	400	400	32	99	1	2	1
010	528	528	32	65	1	1	1
011	664	664	32	82	1	1	1
100	800	800	32	99	1	1	1
101	928	928	32	115	1	1	1
110	1000	1000	32	124	1	1	1
111	1056	1056	32	65	1	0	1

PLL2 Configuration (for ARM926EJ CPU Clock)

PLL_CON FIG	PLLOUT (MHz)	ARM1176JZF (MHz)	FIN (MHz)	DIVF[7:0]	DIVR[5:0]	DIVQ[2:0]	RANGE[2:0]
000	332	332	32	82	1	2	1
001	400	400	32	99	1	2	1
010	528	528	32	65	1	1	1
011	664	664	32	82	1	1	1
100	800	800	32	99	1	1	1
101	928	928	32	115	1	1	1
110	1000	1000	32	124	1	1	1
111	1056	1056	32	65	1	0	1

System Clock Generation (CPU/AXI/AHB/APB)

There are four system clocks in L6021. They provide the clock source for ARM1176JZF, ARM926EJ, internal AMBA (AXI/AHB/APB) bus and external AMBA (AXI/AHB) bus. Below is the detailed clock scheme.



System Reset Generation

The system reset source is from external POR and internal wakeup reset from sleep power mode. The reset generator will generate internal signals including AMBA bus resets and peripheral reset after receive the external reset trigger.

Chapter 3 DDR2 SDRAM Controller

Overview

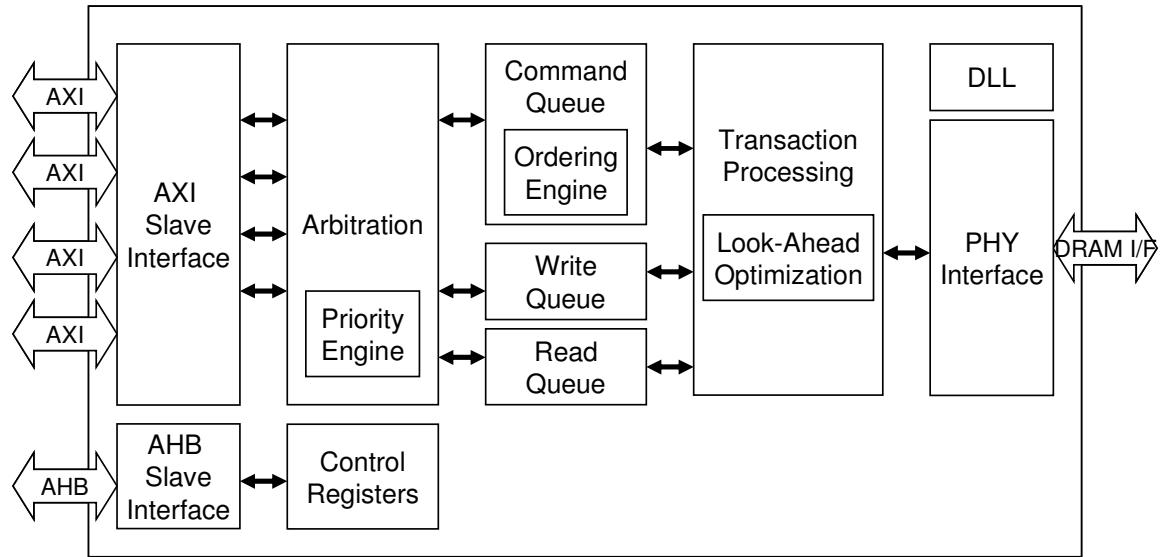
The DDR2 memory controller has one AHB and four AXI compatible bus for programming its configuration registers and for access to SDRAM. CPU could program SDRAM interface timing registers through AHB interface before system data access. For system memory access, there are four AXI ports to provide maximum four-layer AXI bus data access throughput in parallel.

Key Features

- Fully pipelined command, read and write data interfaces to the memory controller.
- Advanced bank look-ahead features for high memory throughput.
- Front-end interface to 4 AXI ports
- A programmable register interface to control memory device parameters and protocols.
- Full initialization of memory on memory controller reset.
- Programmable memory datapath size of full memory data width or half memory data width.

Architecture

Block Diagram



Block Descriptions

AHB Control Register Port

The control registers AHB slave port. It provides CPU to programming SDRAM interface timing and checks SDRAM status.

AXI Data Access Port 1,2,3,4

The AXI data access port is used for AMBA component memory data access. There are four AXI data access port to provide maximum 4-layer internal AXI bus in parallel to enhance data throughput.

Arbitration

The Arbiter is responsible for arbitrating requests from the ports and sending requests to the memory controller. Each transaction received at the Arbiter logic has an associated priority, which works with each port's arbitration logic to determine how ports issue requests to [memory](#) controller. The memory controller supports the Bandwidth Allocation/Priority Round-Robin arbitration scheme.

Command Queue

The memory controller contains a command queue that accepts commands from the Arbiter. This

command queue uses a placement algorithm to determine the order that commands will execute in the memory controller. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at the time. Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. In addition, the placement logic attempts to maximize efficiency of the memory controller through command grouping and bank splitting. Once placed into the command queue, the relative order of commands is constant.

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
DDR2_CTL_00	0x0000	W	0x0000000000	DDR2 control register 00
DDR2_CTL_01	0x0004	W	0x0000000000	DDR2 control register 01
DDR2_CTL_02	0x0008	W	0x0000000000	DDR2 control register 02
DDR2_CTL_03	0x000C	W	0x0000000000	DDR2 control register 03
DDR2_CTL_04	0x0010	W	0x0000000000	DDR2 control register 04
DDR2_CTL_05	0x0014	W	0x0000000000	DDR2 control register 05
DDR2_CTL_06	0x0018	W	0x0000000000	DDR2 control register 06
DDR2_CTL_07	0x001C	W	0x0000000000	DDR2 control register 07
DDR2_CTL_08	0x0020	W	0x0000000000	DDR2 control register 08
DDR2_CTL_09	0x0024	W	0x0000000000	DDR2 control register 09
DDR2_CTL_10	0x0028	W	0x0000000000	DDR2 control register 10
DDR2_CTL_11	0x002C	W	0x0000000000	DDR2 control register 11
DDR2_CTL_12	0x0030	W	0x0000000000	DDR2 control register 12
DDR2_CTL_13	0x0034	W	0x0000000000	DDR2 control register 13
DDR2_CTL_14	0x0038	W	0x0000000000	DDR2 control register 14
DDR2_CTL_15	0x003C	W	0x0000000000	DDR2 control register 15
DDR2_CTL_16	0x0040	W	0x0000000000	DDR2 control register 16
DDR2_CTL_17	0x0044	W	0x0000000000	DDR2 control register 17
DDR2_CTL_18	0x0048	W	0x0000000000	DDR2 control register 18
DDR2_CTL_19	0x004C	W	0x0000000000	DDR2 control register 19
DDR2_CTL_20	0x0050	W	0x0000000000	DDR2 control register 20

DDR2_CTL_21	0x0054	W	0x00000000	DDR2 control register 21
DDR2_CTL_22	0x0058	W	0x00000000	DDR2 control register 22
DDR2_CTL_23	0x005C	W	0x00000000	DDR2 control register 23
DDR2_CTL_24	0x0060	W	0x00000000	DDR2 control register 24
DDR2_CTL_25	0x0064	W	0x00000000	DDR2 control register 25
DDR2_CTL_26	0x0068	W	0x00000000	DDR2 control register 26
DDR2_CTL_27	0x006C	W	0x00000000	DDR2 control register 27
DDR2_CTL_28	0x0070	W	0x00000000	DDR2 control register 28
DDR2_CTL_29	0x0074	W	0x00000000	DDR2 control register 29
DDR2_CTL_30	0x0078	W	0x00000000	DDR2 control register 30
DDR2_CTL_31	0x007C	W	0x00000000	DDR2 control register 31
DDR2_CTL_32	0x0080	W	0x00000000	DDR2 control register 32
DDR2_CTL_33	0x0084	W	0x00000000	DDR2 control register 33
DDR2_CTL_34	0x0088	W	0x00000000	DDR2 control register 34
DDR2_CTL_35	0x008C	W	0x00000000	DDR2 control register 35
DDR2_CTL_36	0x0090	W	0x00000000	DDR2 control register 36
DDR2_CTL_37	0x0094	W	0x00000000	DDR2 control register 37
DDR2_CTL_38	0x0098	W	0x00000000	DDR2 control register 38
DDR2_CTL_39	0x009C	W	0x00000000	DDR2 control register 39
DDR2_CTL_40	0x00A0	W	0x00000000	DDR2 control register 40
DDR2_CTL_41	0x00A4	W	0x00000000	DDR2 control register 41
DDR2_CTL_42	0x00A8	W	0x00000000	DDR2 control register 42
DDR2_CTL_43	0x00AC	W	0x00000000	DDR2 control register 43
DDR2_CTL_44	0x00B0	W	0x00000000	DDR2 control register 44
DDR2_CTL_45	0x00B4	W	0x00000000	DDR2 control register 45
DDR2_CTL_46	0x00B8	W	0x00000000	DDR2 control register 46
DDR2_CTL_47	0x00BC	W	0x00000000	DDR2 control register 47
DDR2_CTL_48	0x00C0	W	0x00000000	DDR2 control register 48
DDR2_CTL_49	0x00C4	W	0x00000000	DDR2 control register 49
DDR2_CTL_50	0x00C8	W	0x00000000	DDR2 control register 50
DDR2_CTL_52	0x00D0	W	0x00000000	DDR2 control register 52
DDR2_CTL_53	0x00D4	W	0x00000000	DDR2 control register 53
DDR2_CTL_54	0x00D8	W	0x00000000	DDR2 control register 54

DDR2_CTL_55	0x00DC	W	0x00000000	DDR2 control register 55
DDR2_CTL_56	0x00E0	W	0x00000000	DDR2 control register 56
DDR2_CTL_57	0x00E4	W	0x00000000	DDR2 control register 57
DDR2_CTL_58	0x00E8	W	0x00000000	DDR2 control register 58
DDR2_CTL_59	0x00EC	W	0x00000000	DDR2 control register 59
DDR2_CTL_60	0x00F0	W	0x00000000	DDR2 control register 60
DDR2_CTL_61	0x00F4	W	0x00000000	DDR2 control register 61
DDR2_CTL_62	0x00F8	W	0x00000000	DDR2 control register 62
DDR2_CTL_63	0x00FC	W	0x00000000	DDR2 control register 63
DDR2_CTL_64	0x0100	W	0x00000000	DDR2 control register 64
DDR2_CTL_65	0x0104	W	0x00000000	DDR2 control register 65
DDR2_CTL_66	0x0108	W	0x00000000	DDR2 control register 66
DDR2_CTL_67	0x010C	W	0x00000000	DDR2 control register 67
DDR2_CTL_68	0x0110	W	0x00000000	DDR2 control register 68
DDR2_CTL_69	0x0114	W	0x00000000	DDR2 control register 69
DDR2_CTL_70	0x0118	W	0x00000000	DDR2 control register 70
DDR2_CTL_71	0x011C	W	0x00000000	DDR2 control register 71
DDR2_CTL_72	0x0120	W	0x00000000	DDR2 control register 72
DDR2_CTL_73	0x0124	W	0x00000000	DDR2 control register 73
DDR2_CTL_74	0x0128	W	0x00000000	DDR2 control register 74
DDR2_CTL_75	0x012C	W	0x00000000	DDR2 control register 75
DDR2_CTL_76	0x0130	W	0x00000000	DDR2 control register 76
DDR2_CTL_77	0x0134	W	0x00000000	DDR2 control register 77
DDR2_CTL_78	0x0138	W	0x00000000	DDR2 control register 78
DDR2_CTL_79	0x013C	W	0x00000000	DDR2 control register 79
DDR2_CTL_80	0x0140	W	0x00000000	DDR2 control register 80
DDR2_CTL_81	0x0144	W	0x00000000	DDR2 control register 81
DDR2_CTL_82	0x0148	W	0x00000000	DDR2 control register 82
DDR2_CTL_83	0x014C	W	0x00000000	DDR2 control register 83
DDR2_CTL_84	0x0150	W	0x00000000	DDR2 control register 84
DDR2_CTL_85	0x0154	W	0x00000000	DDR2 control register 85
DDR2_CTL_86	0x0158	W	0x00000000	DDR2 control register 86
DDR2_CTL_87	0x015C	W	0x00000000	DDR2 control register 87

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

DDR2_CTL_00

Address: Operational Base + offset (0x000)

DDR2 control register 00

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	W	0x0	AREFRESH; Trigger auto-refresh at boundary specified by AUTO_REFRESH_MODE.
23:17	-	-	Reserved.
16	RW	0x0	AP; Enable auto precharge mode of controller.
15:9	-	-	Reserved.
8	RW	0x0	ADDR_CMP_EN; Enable address collision detection for cmd queue placement logic.
7:1	-	-	Reserved.
0	RW	0x0	ACTIVE_AGING; Enable command aging in the command queue.

DDR2_CTL_01

Address: Operational Base + offset (0x004)

DDR2 control register 01

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	W	0x0	AXI2_BDW_OVFLOW; Port 2 behavior when bandwidth maximized.
23:17	-	-	Reserved.
16	RW	0x0	AXI1_BDW_OVFLOW; Port 1 behavior when bandwidth maximized.
15:9	-	-	Reserved.
8	RW	0x0	AXI0_BDW_OVFLOW;

			Port 0 behavior when bandwidth maximized.
7:1	-	-	Reserved.
0	RW	0x0	AUTO_REFRESH_MODE; Define auto-refresh to occur at next burst or next cmd boundary.

DDR2_CTL_02

Address: Operational Base + offset (0x008)

DDR2 control register 02

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	W	0x0	BIST_DATA_CHECK; Enable data checking with BIST operation.
23:17	-	-	Reserved.
16	RW	0x0	BIST_ADDR_CHECK; Enable address checking with BIST operation.
15:9	-	-	Reserved.
8	RW	0x0	BANK_SPLIT_EN; Enable bank splitting for cmd queue placement logic.
7:1	-	-	Reserved.
0	RW	0x0	AXI3_BDW_OVERFLOW; Port 3 behavior when bandwidth maximized.

DDR2_CTL_03

Address: Operational Base + offset (0x00C)

DDR2 control register 03

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	W	0x0	CONCURRENTAP_WR_ONLY; Limit concurrent auto-precharge by waiting for the write recovery time, tWR, before issuing a read.
23:17	-	-	Reserved.
16	RW	0x0	CONCURRENTAP; Allow controller to issue cmds to other banks while a bank

			is in auto pre-charge.
15:9	-	-	Reserved.
8	R	0x0	CKE_STATUS; Register access to cke_status signal.
7:1	-	-	Reserved.
0	W	0x0	BIST_GO; Initiate a BIST operation.

DDR2_CTL_04

Address: Operational Base + offset (0x010)

DDR2 control register 04

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	DRAM_CLK_DISABLE; Set value for the dfi_dram_clk_disable signal.
23:17	-	-	Reserved.
16	RW	0x0	DQS_N_EN; Set DQS pin as single-ended or differential.
15:9	-	-	Reserved.
8	RW	0x0	DLL_BYPASS_MODE; Enable the DLL bypass feature of the controller.
7:1	-	-	Reserved.
0	R	0x0	DLLLOCKREG; Status of DLL lock coming out of master delay.

DDR2_CTL_05

Address: Operational Base + offset (0x014)

DDR2 control register 05

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	FAST_WRITE; Define when write cmds are issued to DRAM devices.
23:17	-	-	Reserved.
16	RW	0x0	ENABLE_QUICK_SREFRESH;

			Allow user to interrupt memory initialization to enter self-refresh mode.
15:9	-	-	Reserved.
8	RW	0x0	EIGHT_BANK_MODE; Number of banks on the DRAM(s).
7:1	-	-	Reserved.
0	RW	0x0	DRIVE_DQ_DQS; Sets DQ/DQS output enable behavior when controller is idle.

DDR2_CTL_06

Address: Operational Base + offset (0x018)

DDR2 control register 06

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	LOWPOWER_REFRESH_ENABLE; Enable refreshes while in low power mode.
23:17	-	-	Reserved.
16	RW	0x0	INTRPTWRITEA; Allow the controller to interrupt a combined write with auto pre-charge cmd with another write cmd.
15:9	-	-	Reserved.
8	RW	0x0	INTRPTREADA; Allow the controller to interrupt a combined read with auto pre-charge cmd with another read cmd.
7:1	-	-	Reserved.
0	RW	0x0	INTRPTAPBURST; Allow the controller to interrupt an auto pre-charge cmd with another cmd.

DDR2_CTL_07

Address: Operational Base + offset (0x01C)

DDR2 control register 07

Bit	Attr	Reset Value	Description

31:25	-	-	Reserved.
24	RW	0x0	PLACEMENT_EN; Enable placement logic for cmd queue.
23:17	-	-	Reserved.
16	RW	0x0	ODT_ALT_EN; Enable use of non-DFI odt_alt signal.
15:9	-	-	Reserved.
8	RW	0x0	NO_CMD_INIT; Disable DRAM cmds until TDLL has expired during initialization.
7:1	-	-	Reserved.
0	R	0x0	MAX_CS_REG; Maximum number of chip selects available.

DDR2_CTL_08

Address: Operational Base + offset (0x020)

DDR2 control register 08

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	REDUC; Enable the half datapath feature of the controller.
23:17	-	-	Reserved.
16	RW	0x0	PWRUP_SREFRESH_EXIT; Allow powerup via self-refresh instead of full memory initialization.
15:9	-	-	Reserved.
8	RW	0x0	PRIORITY_EN; Enable priority for cmd queue placement logic.
7:1	-	-	Reserved.
0	RW	0x0	POWER_DOWN; Disable clock enable and set DRAMs in power-down state.

DDR2_CTL_09

Address: Operational Base + offset (0x024)

DDR2 control register 09

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	RW_SAME_EN; Enable read/write grouping for cmd queue placement logic.
23:17	-	-	Reserved.
16	RW	0x0	RESYNC_DLL_PER_AREF_EN; Enables automatic DLL resyncs after every refresh.
15:9	-	-	Reserved.
8	RW	0x0	RESYNC_DLL; Initiate a DLL resync. WRITEONLY
7:1	-	-	Reserved.
0	RW	0x0	REG_DIMM_ENABLE; Enable registered DIMM operation of the controller.

DDR2_CTL_10

Address: Operational Base + offset (0x028)

DDR2 control register 10

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x0	SWAP_PORT_RW_SAME_EN; No meaning for this MC.
23:17	-	-	Reserved.
16	RW	0x0	SWAP_EN; Enable command swapping logic in execution unit.
15:9	-	-	Reserved.
8	RW	0x0	START; Initiate cmd processing in the controller.
7:1	-	-	Reserved.
0	RW	0x0	SREFRESH; Place DRAMs in self-refresh mode.

DDR2_CTL_11

Address: Operational Base + offset (0x02C)

DDR2 control register 11

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	W	0x0	WRITE_MODEREG; Write mode register data to the DRAMs.
23:17	-	-	Reserved.
16	RW	0x0	WRITEINTERP; Allow controller to interrupt a write burst to the DRAMs with a read cmd.
15:9	-	-	Reserved.
8	RW	0x0	TREF_ENABLE; Issue auto-refresh cmds to the DRAMs every TREF cycles.
7:1	-	-	Reserved.
0	RW	0x0	TRAS_LOCKOUT; Allow the controller to execute auto pre-charge cmds before TRAS_MIN expires.

DDR2_CTL_12

Address: Operational Base + offset (0x030)

DDR2 control register 12

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	AXI1_FIFO_TYPE_REG; Clock domain relativity between AXI port 1 and memory controller core.
23:18	-	-	Reserved.
17:16	RW	0x0	AXI0_W_PRIORITY; Priority of write cmds from AXI port 0.
15:10	-	-	Reserved.
9:8	RW	0x0	AXI0_R_PRIORITY; Priority of read cmds from AXI port 0.
7:2	-	-	Reserved.

1:0	RW	0x0	AXI0_FIFO_TYPE_REG; Clock domain relativity between AXI port 0 and memory controller core.
-----	----	-----	---

DDR2_CTL_13

Address: Operational Base + offset (0x034)

DDR2 control register 13

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	AXI2_R_PRIORITY; Priority of read cmds from AXI port 2.
23:18	-	-	Reserved.
17:16	RW	0x0	AXI2_FIFO_TYPE_REG; Clock domain relativity between AXI port 2 and memory controller core.
15:10	-	-	Reserved.
9:8	RW	0x0	AXI1_W_PRIORITY; Priority of write cmds from AXI port 1.
7:2	-	-	Reserved.
1:0	RW	0x0	AXI1_R_PRIORITY; Priority of read cmds from AXI port 1.

DDR2_CTL_14

Address: Operational Base + offset (0x038)

DDR2 control register 14

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	RW	0x0	AXI3_W_PRIORITY; Priority of write cmds from AXI port 3.
23:18	-	-	Reserved.
17:16	RW	0x0	AXI3_R_PRIORITY; Priority of read cmds from AXI port 3.
15:10	-	-	Reserved.
9:8	RW	0x0	AXI3_FIFO_TYPE_REG;

			Clock domain relativity between AXI port 3 and memory controller core.
7:2	-	-	Reserved.
1:0	RW	0x0	AXI2_W_PRIORITY; Priority of write cmds from AXI port 2.

DDR2_CTL_15

Address: Operational Base + offset (0x03C)

DDR2 control register 15

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:24	R	0x0	DLL_OBS_REG_0_2; Reports status of the TSTCLK1 and TSTCLK2 outputs for data slice 2.
23:18	-	-	Reserved.
17:16	R	0x0	DLL_OBS_REG_0_1; Reports status of the TSTCLK1 and TSTCLK2 outputs for data slice 1
15:10	-	-	Reserved.
9:8	R	0x0	DLL_OBS_REG_0_0; Reports status of the TSTCLK1 and TSTCLK2 outputs for data slice 0.
7:2	-	-	Reserved.
1:0	R	0x0	BIST_RESULT; BIST operation status (pass/fail).

DDR2_CTL_16

Address: Operational Base + offset (0x040)

DDR2 control register 16

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	CASLAT; Encoded CAS latency sent to DRAMs during initialization.

23:19	-	-	Reserved.
18:16	RW	0x0	ARB_CMD_Q_THRESHOLD; Threshold for cmd queue fullness related to overflow.
15:11	-	-	Reserved.
10:8	RW	0x0	ADDR_PINS; Difference between number of addr pins available and number being used.
7:2	-	-	Reserved.
1:0	R	0x0	DLL_OBS_REG_0_3; Reports status of the TSTCLK1 and TSTCLK2 outputs for data slice 3.

DDR2_CTL_17

Address: Operational Base + offset (0x044)

DDR2 control register 17

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	Q_FULLNESS; Quantity that determines cmd queue full.
23:19	-	-	Reserved.
18:16	R	0x0	PORT_DATA_ERROR_TYPE; Type of error and access type that caused the PORT data error.
15:11	-	-	Reserved.
10:8	RW	0x0	COLUMN_SIZE; Difference between number of column pins available and number being used.
7:3	-	-	Reserved.
2:0	RW	0x0	CKE_DELAY; Additional cycles to delay CKE for status reporting.

DDR2_CTL_18

Address: Operational Base + offset (0x048)

DDR2 control register 18

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	TCKE; Minimum CKE pulse width.
23:19	-	-	Reserved.
18:16	RW	0x0	TBST_INT_INTERVAL; DRAM burst interrupt interval in cycles.
15:11	-	-	Reserved.
10:8	RW	0x0	R2W_SAMECS_DLY; Additional delay to insert between reads and writes to the same chip select.
7:3	-	-	Reserved.
2:0	RW	0x0	R2R_SAMECS_DLY; Additional delay to insert between reads and reads to the same chip select.

DDR2_CTL_19

Address: Operational Base + offset (0x04C)

DDR2 control register 19

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x0	W2R_SAMECS_DLY; Additional delay to insert between writes and reads to the same chip select.
23:19	-	-	Reserved.
18:16	RW	0x0	TRTP; DRAM TRTP parameter in cycles.
15:11	-	-	Reserved.
10:8	RW	0x0	TRRD; DRAM TRRD parameter in cycles.
7:3	-	-	Reserved.
2:0	RW	0x0	TDFI_DRAM_CLK_DISABLE; Delay from DFI clock disable to memory clock disable.

DDR2_CTL_20

Address: Operational Base + offset (0x050)

DDR2 control register 20

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	APREBIT; Location of the auto pre-charge bit in the DRAM address.
23:20	-	-	Reserved.
19:16	RW	0x0	AGE_COUNT; Initial value of master aging-rate counter for cmd aging.
15:12	-	-	Reserved.
11:8	RW	0x0	ADD_ODT_CLK_DIFFTYPE_SAMECS; Additional delay to insert between different transaction types to the same chip select to meet ODT timing requirements.
7:3	-	-	Reserved.
2:0	RW	0x0	W2W_SAMECS_DLY; Additional delay to insert between writes and writes to the same chip select.

DDR2_CTL_21

Address: Operational Base + offset (0x054)

DDR2 control register 21

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	CKSRX; Clock stable delay on self-refresh exit.
23:20	-	-	Reserved.
19:16	RW	0x0	CKSRE; Clock hold delay on self-refresh entry.
15:12	-	-	Reserved.
11:8	RW	0x0	CASLAT_LIN_GATE; Adjusts data capture gate open by half cycles.
7:4	-	-	Reserved.

3:0	RW	0x0	CASLAT_LIN; Sets latency from read cmd send to data receive from/to controller.
-----	----	-----	--

DDR2_CTL_22

Address: Operational Base + offset (0x058)

DDR2 control register 22

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	R	0x0	MAX_COL_REG; Maximum width of column address in DRAMs.
23:20	-	-	Reserved.
19:16	RW	0x0	INITAREF; Number of auto-refresh cmds to execute during DRAM initialization.
15:12	-	-	Reserved.
11:8	RW	0x0	DRAM_CLASS; Defines the mode of operation of the controller.
7:4	-	-	Reserved.
3:0	RW	0x0	COMMAND_AGE_COUNT; Initial value of individual cmd aging counters for cmd aging.

DDR2_CTL_23

Address: Operational Base + offset (0x05C)

DDR2 control register 23

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	R	0x0	TDFI_CTRLUPD_MIN; Holds the DFI tCTRLUPD_MIN timing parameter.
23:20	-	-	Reserved.
19:16	RW	0x0	RDLAT_ADJ; Adjustment value for PHY read timing.
15:12	-	-	Reserved.

11:8	R	0x0	PORT_CMD_ERROR_TYPE; Type of error and access type that caused the PORT cmd error.
7:4	-	-	Reserved.
3:0	R	0x0	MAX_ROW_REG; Maximum width of memory address bus.

DDR2_CTL_24

Address: Operational Base + offset (0x060)

DDR2 control register 24

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	R	0x0	TDFI_PHY_WRLAT; Holds the calculated DFI tPHY_WRLAT timing parameter.
23:20	-	-	Reserved.
19:16	RW	0x0	TDFI_PHY_RDLAT; Holds the tPHY_RDLAT timing parameter.
15:12	-	-	Reserved.
11:8	RW	0x0	TDFI_DRAM_CLK_ENABLE; Delay from DFI clock enable to memory clock enable.
7:4	-	-	Reserved.
3:0	RW	0x0	TDFI_CTRL_DELAY; Delay from DFI command to memory command.

DDR2_CTL_25

Address: Operational Base + offset (0x064)

DDR2 control register 25

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	TRP; DRAM TRP parameter in cycles.
23:20	-	-	Reserved.
19:16	RW	0x0	TDFI_RDDATA_EN_BASE;

			Sets DFI base value for the tRDDATA_EN timing parameter.
15:12	-	-	Reserved.
11:8	R	0x0	TDFI_RDDATA_EN; Holds the calculated DFI tRDDATA_EN timing parameter.
7:4	-	-	Reserved.
3:0	RW	0x0	TDFI_PHY_WRLAT_BASE; Sets DFI base value for the tPHY_WRLAT timing parameter.

DDR2_CTL_26

Address: Operational Base + offset (0x068)

DDR2 control register 26

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	RW	0x0	WRLAT_ADJ; Adjustment value for PHY writes timing.
23:20	-	-	Reserved.
19:16	RW	0x0	WRLAT; DRAM WRLAT parameter in cycles.
15:12	-	-	Reserved.
11:8	RW	0x0	TWTR; DRAM TWTR parameter in cycles.
7:4	-	-	Reserved.
3:0	RW	0x0	TRP_AB; DRAM TRP All Bank parameter in cycles.

DDR2_CTL_27

Address: Operational Base + offset (0x06C)

DDR2 control register 27

Bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:24	RW	0x0	OCD_ADJUST_PUP_CS_0; OCD pull-up adjust setting for DRAMs for chip select 0.

23:21	-	-	Reserved.
20:16	RW	0x0	OCD_ADJUST_PDN_CS_0; OCD pull-down adjust setting for DRAMs for chip select 0.
15:13	-	-	Reserved.
12:8	RW	0x0	LOWPOWER_CONTROL; Controls entry into the low power modes.
7:5	-	-	Reserved.
4:0	RW	0x0	LOWPOWER_AUTO_ENABLE; Enables automatic entry into the low power mode on idle.

DDR2_CTL_28

Address: Operational Base + offset (0x070)

DDR2 control register 28

Bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28:24	RW	0x0	TMRD; DRAM TMRD parameter in cycles.
23:21	-	-	Reserved.
20:16	RW	0x0	TDAL; DRAM TDAL parameter in cycles.
15:13	-	-	Reserved.
12:8	RW	0x0	TCKESR; Minimum CKE low pulse width during a self-refresh.
7:5	-	-	Reserved.
4:0	RW	0x0	TCCD; DRAM CAS-to-CAS parameter in cycles.

DDR2_CTL_29

Address: Operational Base + offset (0x074)

DDR2 control register 29

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:24	RW	0x0	TFAW;

			DRAM TFAW parameter in cycles.
23:22	-	-	Reserved.
21:16	R	0x0	OUT_OF_RANGE_TYPE; Type of cmd that caused an Out-of-Range interrupt.
15:14	-	-	Reserved.
13:8	RW	0x0	ADDR_SPACE; Sets the number of address bits to check during BIST operation.
7:5	-	-	Reserved.
4:0	RW	0x0	TWR_INT; DRAM TWR parameter in cycles.

DDR2_CTL_30

Address: Operational Base + offset (0x078)

DDR2 control register 30

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	RW	0x0	AXI1_BDW; Maximum bandwidth percentage for port 1. Range: 0x0-0x64
23	-	-	Reserved.
22:16	R	0x0	AXI0_CURRENT_BDW; Current bandwidth usage percentage for port 0. Range: 0x0-0x64
15	-	-	Reserved.
14:8	RW	0x0	AXI0_BDW; Maximum bandwidth percentage for port 0. Range: 0x0-0x64
7:6	-	-	Reserved.
5:0	RW	0x0	TRC; DRAM TRC parameter in cycles.

DDR2_CTL_31

Address: Operational Base + offset (0x07C)

DDR2 control register 31

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	RW	0x0	AXI3_BDW; Maximum bandwidth percentage for port 3. Range: 0x0-0x64
23	-	-	Reserved.
22:16	R	0x0	AXI2_CURRENT_BDW; Current bandwidth usage percentage for port 2. Range: 0x0-0x64
15	-	-	Reserved.
14:8	RW	0x0	AXI2_BDW; Maximum bandwidth percentage for port 2. Range: 0x0-0x64
7	-	-	Reserved.
6:0	R	0x0	AXI1_CURRENT_BDW; Current bandwidth usage percentage for port 1. Range: 0x0-0x64

DDR2_CTL_32

Address: Operational Base + offset (0x080)

DDR2 control register 32

Bit	Attr	Reset Value	Description
31:24	R	0x0	DLL_LOCK; Number of delay elements in master DLL lock.
23:16	RW	0x0	DFT_CTRL_REG; Enables the PHY testing mode.
15	-	-	Reserved.
14:8	R	0x0	OUT_OF_RANGE_LENGTH; Length of cmd that caused an Out of Range interrupt.
7	-	-	Reserved.
6:0	R	0x0	AXI3_CURRENT_BDW; Current bandwidth usage percentage for port 3. Range: 0x0-0x64

DDR2_CTL_33

Address: Operational Base + offset (0x084)

DDR2 control register 33

Bit	Attr	Reset Value	Description
31:24	RW	0x0	TRCD_INT; DRAM TRCD parameter in cycles.
23:16	RW	0x0	TRAS_MIN; DRAM TRAS_MIN parameter in cycles.
15:8	RW	0x0	TMOD; Number of clock cycles after MRS command and before any other command.
7:0	RW	0x0	DLL_RST_ADJ_DLY; Minimum number of cycles after setting master delay in DLL until reset is released.

DDR2_CTL_34

Address: Operational Base + offset (0x088)

DDR2 control register 34

Bit	Attr	Reset Value	Description
31:18	-	-	Reserved.
17:8	W	0x0	INT_ACK; Clear mask of the INT_STATUS parameter.
7:0	RW	0x0	TRFC; DRAM TRFC parameter in cycles.

DDR2_CTL_35

Address: Operational Base + offset (0x08C)

DDR2 control register 35

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	R	0x0	INT_STATUS; Status of interrupt features in the controller.
15:11	-	-	Reserved.

10:0	RW	0x0	INT_MASK; Mask for controller_int signals from the INT_STATUS parameter.
------	----	-----	---

DDR2_CTL_36

Address: Operational Base + offset (0x090)

DDR2 control register 36

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:16	R	0x0	PORT_CMD_ERROR_ID; Source ID of cmd that caused the PORT cmd error.
15:11	-	-	Reserved.
10:0	R	0x0	OUT_OF_RANGE_SOURCE_ID; Source ID of cmd that caused an Out-of-Range interrupt.

DDR2_CTL_37

Address: Operational Base + offset (0x094)

DDR2 control register 37

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	TDFI_CTRLUPD_MAX; Holds the DFI tCTRLUPD_MAX timing parameter.
15:11	-	-	Reserved.
10:0	R	0x0	PORT_DATA_ERROR_ID; Source ID of cmd that caused the PORT data error.

DDR2_CTL_38

Address: Operational Base + offset (0x098)

DDR2 control register 38

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	TDFI_PHYUPD_TYPE0; Holds the DFI tPHYUPD_TYPE0 timing parameter.
15:14	-	-	Reserved.
13:0	RW	0x0	TDFI_PHYUPD_RESP;

			Holds the DFI tPHYUPD_RESP timing parameter.
--	--	--	--

DDR2_CTL_39

Address: Operational Base + offset (0x09C)

DDR2 control register 39

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	TDFI_PHYUPD_TYPE2; Holds the DFI tPHYUPD_TYPE2 timing parameter.
15:14	-	-	Reserved.
13:0	RW	0x0	TDFI_PHYUPD_TYPE1; Holds the DFI tPHYUPD_TYPE1 timing parameter.

DDR2_CTL_40

Address: Operational Base + offset (0x0A0)

DDR2 control register 40

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	RW	0x0	TREF; DRAM TREF parameter in cycles.
15:14	-	-	Reserved.
13:0	RW	0x0	TDFI_PHYUPD_TYPE3; Holds the DFI tPHYUPD_TYPE3 timing parameter.

DDR2_CTL_41

Address: Operational Base + offset (0x0A4)

DDR2 control register 41

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:16	RW	0x0	MR1_DATA_0; Data to program into memory mode register 1 for chip select 0.
15	-	-	Reserved.
14:0	RW	0x0	MR0_DATA_0;

			MRS data to program to memory mode register 0 for chip select 0.
--	--	--	--

DDR2_CTL_42

Address: Operational Base + offset (0x0A8)

DDR2 control register 42

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:16	RW	0x0	MR3_DATA_0; Data to program into memory mode register 3 for chip select 0.
15	-	-	Reserved.
14:0	RW	0x0	MR2_DATA_0; Data to program into memory mode register 2 for chip select 0.

DDR2_CTL_43

Address: Operational Base + offset (0x0AC)

DDR2 control register 43

Bit	Attr	Reset Value	Description
31:16	RW	0x0	AXI1_EN_SIZE_LT_WIDTH_INSTR; Allow narrow instructions from AXI port 1 requestors with bit enabled.
15:0	RW	0x0	AXI0_EN_SIZE_LT_WIDTH_INSTR; Allow narrow instructions from AXI port 0 requestors with bit enabled.

DDR2_CTL_44

Address: Operational Base + offset (0x0B0)

DDR2 control register 44

Bit	Attr	Reset Value	Description
31:16	RW	0x0	AXI3_EN_SIZE_LT_WIDTH_INSTR; Allow narrow instructions from AXI port 3 requestors with bit enabled.

15:0	RW	0x0	AXI2_EN_SIZE_LT_WIDTH_INSTR; Allow narrow instructions from AXI port 2 requestors with bit enabled.
------	----	-----	--

DDR2_CTL_45

Address: Operational Base + offset (0x0B4)

DDR2 control register 45

Bit	Attr	Reset Value	Description
31:16	RW	0x0	LOWPOWER_EXTERNAL_CNT; Counts idle cycles to self-refresh with memory clock gating.
15:0	RW	0x0	DLL_RST_DELAY; Minimum number of cycles required for DLL reset.

DDR2_CTL_46

Address: Operational Base + offset (0x0B8)

DDR2 control register 46

Bit	Attr	Reset Value	Description
31:16	RW	0x0	LOWPOWER_POWER_DOWN_CNT; Counts idle cycles to memory power-down.
15:0	RW	0x0	LOWPOWER_INTERNAL_CNT; Counts idle cycles to self-refresh with memory and controller clk gating.

DDR2_CTL_47

Address: Operational Base + offset (0x0BC)

DDR2 control register 47

Bit	Attr	Reset Value	Description
31:16	RW	0x0	LOWPOWER_SELF_REFRESH_CNT; Counts idle cycles to memory self-refresh.
15:0	RW	0x0	LOWPOWER_REFRESH_HOLD; Re-Sync counter for DLL in Clock Gate Mode.

DDR2_CTL_48

Address: Operational Base + offset (0x0C0)

DDR2 control register 48

Bit	Attr	Reset Value	Description
31:16	RW	0x0	TDLL; DRAM TDLL parameter in cycles.
15:0	RW	0x0	TCPD; DRAM TCPD parameter in cycles.

DDR2_CTL_49

Address: Operational Base + offset (0x0C4)

DDR2 control register 49

Bit	Attr	Reset Value	Description
31:16	RW	0x0	TRAS_MAX; DRAM TRAS_MAX parameter in cycles.
15:0	RW	0x0	TPDEX; DRAM TPDEX parameter in cycles.

DDR2_CTL_50

Address: Operational Base + offset (0x0C8)

DDR2 control register 50

Bit	Attr	Reset Value	Description
31:16	RW	0x0	TXSR; DRAM TXSR parameter in cycles.
15:0	RW	0x0	TXSNR; DRAM TXSNR parameter in cycles.

DDR2_CTL_52

Address: Operational Base + offset (0x0D0)

DDR2 control register 52

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	RW	0x0	TINIT; DRAM TINIT parameter in cycles.

DDR2_CTL_53

Address: Operational Base + offset (0x0D4)

DDR2 control register 53

Bit	Attr	Reset Value	Description
31:0	RW	0x0	BIST_DATA_MASK; Mask applied to data for BIST error checking.

DDR2_CTL_54

Address: Operational Base + offset (0x0D8)

DDR2 control register 54

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_0_0; Controls the DLL adjust module for data slice 0.

DDR2_CTL_55

Address: Operational Base + offset (0x0DC)

DDR2 control register 55

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_0_1; Controls the DLL adjust module for data slice 1.

DDR2_CTL_56

Address: Operational Base + offset (0x0E0)

DDR2 control register 56

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_0_2; Controls the DLL adjust module for data slice 2.

DDR2_CTL_57

Address: Operational Base + offset (0x0E4)

DDR2 control register 57

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_0_3; Controls the DLL adjust module for data slice 3.

DDR2_CTL_58

Address: Operational Base + offset (0x0E8)

DDR2 control register 58

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_0; Controls the TSTCTRL bits for data slice 0.

DDR2_CTL_59

Address: Operational Base + offset (0x0EC)

DDR2 control register 59

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_1; Controls the TSTCTRL bits for data slice 1.

DDR2_CTL_60

Address: Operational Base + offset (0x0F0)

DDR2 control register 60

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_2; Controls the TSTCTRL bits for data slice 2.

DDR2_CTL_61

Address: Operational Base + offset (0x0F4)

DDR2 control register 61

Bit	Attr	Reset Value	Description
31:0	RW	0x0	DLL_CTRL_REG_1_3; Controls the TSTCTRL bits for data slice 3.

DDR2_CTL_62

Address: Operational Base + offset (0x0F8)

DDR2 control register 62

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PAD_CTRL_REG_0;

			Controls pad termination, type and IDDQ settings.
--	--	--	---

DDR2_CTL_63

Address: Operational Base + offset (0x0FC)

DDR2 control register 63

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_0; Controls pad output enable times and other PHY parameters for data slice 0.

DDR2_CTL_64

Address: Operational Base + offset (0x100)

DDR2 control register 64

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_1; Controls pad output enable times and other PHY parameters for data slice 1.

DDR2_CTL_65

Address: Operational Base + offset (0x104)

DDR2 control register 65

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_2; Controls pad output enable times and other PHY parameters for data slice 2.

DDR2_CTL_66

Address: Operational Base + offset (0x108)

DDR2 control register 66

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_0_3; Controls pad output enable times and other PHY parameters for data slice 3.

DDR2_CTL_67

Address: Operational Base + offset (0x10C)

DDR2 control register 67

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_0; Controls pad termination and loopback for data slice 0.

DDR2_CTL_68

Address: Operational Base + offset (0x110)

DDR2 control register 68

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_1; Controls pad termination and loopback for data slice 1.

DDR2_CTL_69

Address: Operational Base + offset (0x114)

DDR2 control register 69

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_2; Controls pad termination and loopback for data slice 2.

DDR2_CTL_70

Address: Operational Base + offset (0x118)

DDR2 control register 70

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_1_3; Controls pad termination and loopback for data slice 3.

DDR2_CTL_71

Address: Operational Base + offset (0x11C)

DDR2 control register 71

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PHY_CTRL_REG_2; Selects the dfi_rddata_valid delay.

DDR2_CTL_72

Address: Operational Base + offset (0x120)

DDR2 control register 72

Bit	Attr	Reset Value	Description
31:0	R	0x0	PHY_OBS_REG_0_0; Controls loopback status, data and masking info for slice 0.

DDR2_CTL_73

Address: Operational Base + offset (0x124)

DDR2 control register 73

Bit	Attr	Reset Value	Description
31:0	R	0x0	PHY_OBS_REG_0_1; Controls loopback status, data and masking info for slice 1.

DDR2_CTL_74

Address: Operational Base + offset (0x128)

DDR2 control register 74

Bit	Attr	Reset Value	Description
31:0	R	0x0	PHY_OBS_REG_0_2; Controls loopback status, data and masking info for slice 2.

DDR2_CTL_75

Address: Operational Base + offset (0x12C)

DDR2 control register 75

Bit	Attr	Reset Value	Description
31:0	R	0x0	PHY_OBS_REG_0_3; Controls loopback status, data and masking info for slice 3.

DDR2_CTL_76

Address: Operational Base + offset (0x130)

DDR2 control register 76

Bit	Attr	Reset Value	Description
31:0	R	0x0	BIST_FAIL_ADDR [31:0]; Address of BIST error.

DDR2_CTL_77

Address: Operational Base + offset (0x134)

DDR2 control register 77

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0x0	BIST_FAIL_ADDR [32]; Address of BIST error.

DDR2_CTL_78

Address: Operational Base + offset (0x138)

DDR2 control register 78

Bit	Attr	Reset Value	Description
31:0	RW	0x0	BIST_START_ADDRESS [31:0]; Start BIST checking at this address.

DDR2_CTL_79

Address: Operational Base + offset (0x13C)

DDR2 control register 79

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	BIST_START_ADDRESS [32]; Start BIST checking at this address.

DDR2_CTL_80

Address: Operational Base + offset (0x140)

DDR2 control register 80

Bit	Attr	Reset Value	Description
31:0	R	0x0	OUT_OF_RANGE_ADDR [31:0];

			Address of cmd that caused an Out-of-Range interrupt.
--	--	--	---

DDR2_CTL_81

Address: Operational Base + offset (0x144)

DDR2 control register 81

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0x0	OUT_OF_RANGE_ADDR [32]; Address of cmd that caused an Out-of-Range interrupt.

DDR2_CTL_82

Address: Operational Base + offset (0x148)

DDR2 control register 82

Bit	Attr	Reset Value	Description
31:0	R	0x0	PORT_CMD_ERROR_ADDR [31:0]; Address of port that caused the PORT cmd error.

DDR2_CTL_83

Address: Operational Base + offset (0x14C)

DDR2 control register 83

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0x0	PORT_CMD_ERROR_ADDR [32]; Address of port that caused the PORT cmd error.

DDR2_CTL_84

Address: Operational Base + offset (0x150)

DDR2 control register 84

Bit	Attr	Reset Value	Description
31:0	R	0x0	BIST_EXP_DATA [31:0]; Expected data on BIST error.

DDR2_CTL_85

Address: Operational Base + offset (0x154)

DDR2 control register 85

Bit	Attr	Reset Value	Description
31:0	R	0x0	BIST_EXP_DATA [63:32]; Expected data on BIST error.

DDR2_CTL_86

Address: Operational Base + offset (0x158)

DDR2 control register 86

Bit	Attr	Reset Value	Description
31:0	R	0x0	BIST_FAIL_DATA [31:0]; Actual data on BIST error.

DDR2_CTL_87

Address: Operational Base + offset (0x15C)

DDR2 control register 87

Bit	Attr	Reset Value	Description
31:0	R	0x0	BIST_FAIL_DATA [63:32]; Actual data on BIST error.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only**Functional Description****DDR2 Initialization Sequence**

On power-on reset, software must initialize the DRAM controller and each of the DDR2 connected to the DRAM controller. Refer to the DDR2 data sheet for the start up procedure. Example sequences are given below:

- Clear the `rst_n` signal by driving it to 'b0. All programmable registers will be cleared.
- Set the `rst_n` signal synchronously with the memory controller clock by driving the signal to 'b1.
- Issue write register commands to configure the DRAM protocols and the settings for the DCC. Keep the start parameter de-asserted during this initialization step.

- Assert the start parameter. This triggers the memory controller to execute the initialization sequence using the parameters written into the registers.

Memory Mapping for Address Space

The maximum allowable address space and mapping into the DRAM devices for the Memory Controller is shown in below figure. This map corresponds to a memory device with 15 row bits and 13 column bits.

32	18	17	15	14	2	1	0
Row		Bank		Column		Datapath	

The `addr_pins` and `column_size` parameters can each range from the maximum configured for the memory controller to seven bits smaller than the maximum configured. This allows the Memory Controller to function with a wide variety of memory device sizes.

The settings for the `addr_pins` and `column_size` parameters control how the address map is used to decode the user address to the DRAM chip selects and row and column addresses. The `eight_bank_mode` parameter controls the address when eight bank mode is supported. It is assumed that the values in these parameters never exceed the maximum values configured. Using the example shown in above figure, if the memory controller is wired to devices with 12 row pins and 11 column bits, the maximum accessible memory space would be reduced. The accessible memory space for this configuration is 256 MB.

The address map for this configuration is shown in below figure. Note that address bits 28 through 33 are listed as ‘don’t care’ bits. These bits are ignored when the memory controller generates the address to the DRAM devices, but they are used to verify that the address lies within the usable address range of the memory controller. Therefore, the user should drive these bits to ‘b0 to avoid the memory controller interpreting the command as being out-of-range and setting one or both of the out-of-range interrupt bits.

33	28	17	16	15	13	12	2	1	0
Don’t Care		Row		Bank		Column		Datapath	

Multiple-Port Arbiter

The Arbiter is responsible for arbitrating requests from the ports and sending requests to the

memory controller core. Each transaction received at the Arbiter logic has an associated priority, which works with each port's arbitration logic to determine how ports issue requests to the memory controller core. This memory controller supports the Bandwidth Allocation/Priority Round-Robin arbitration scheme. The Arbiter logic routes read data from the memory controller core to the appropriate port. The requesting port is assumed able to receive the data. Write data from each port is connected directly to its own write data interface in the memory controller core, allowing the ports to independently pass write data to the memory controller core buffers.

Round-Robin Arbitration

Round-robin operation is a simple form of arbitration which offers each port an opportunity to issue a command. This scheme uses a counter that rotates through the port numbers, incrementing every time a port request is granted. If the port that the counter is referencing has an active request, and the memory controller core command queue is not full, then this request will be sent to the memory controller core. If there is not an active request for that port, then the port will be skipped and the next port will be checked. The counter will increment by one whenever any request has been processed, regardless of which port's request was arbitrated. Round-robin operation ensures that each port's requests can be successfully arbitrated into the memory controller core every 4 cycles. No port will ever be locked out, and any port can have its requests serviced on every cycle as long as all other ports are quiet and the command queue is not full.

An example of the round-robin scheme is shown in below table. Cycles 0, 2 and 6 show the system behavior when the command queue is full. Cycle 8 shows the system behavior when the port addressed by the arbitration counter does not have an active request. All other cycles show normal behavior.

Cycle	Port Addressed by the Arbitration Counter	Ports Requesting				Command Queue Full?	Winner of Arbitration	Value of Counter at Next Cycle
		Port 0	Port 1	Port 2	Port 3			
0	0	Y	Y	Y	Y	Yes	None	0
1	0	Y	Y	Y	Y	No	P0	1
2	1		Y	Y	Y	Yes	None	1
3	1	Y	Y	Y	Y	No	P1	2
4	2	Y		Y	Y	No	P2	3
5	3	Y			Y	No	P3	0

6	0	Y		Y		Yes	None	0
7	0	Y		Y		No	P0	1
8	1			Y		No	P2	2
9	2			Y	Y	No	P2	3
10	3	Y			Y	No	P3	0

AXI Port Priority

For AXI ports, the priority is associated with a port and each port has separate priority parameter for reads and writes. These values are stored into the programmable parameters `axiY_r_priority` and `axiY_w_priority` (where Y represents the port number) at controller initialization. Internally, the ports are organized into priority groups based on their priority settings. All ports within a priority group are treated equally for arbitration unless a port has exceeded its allocated bandwidth. The priority value is also used by the placement logic inside the memory controller core when filling the command queue. A priority value of 0 is highest priority, and a priority value of (decimal) 3 is the lowest priority in the memory controller. The user may program at priority level 0; however, it is best to reserve this priority value so that the placement queue can elevate to this level through aging.

AXI Port Bandwidth

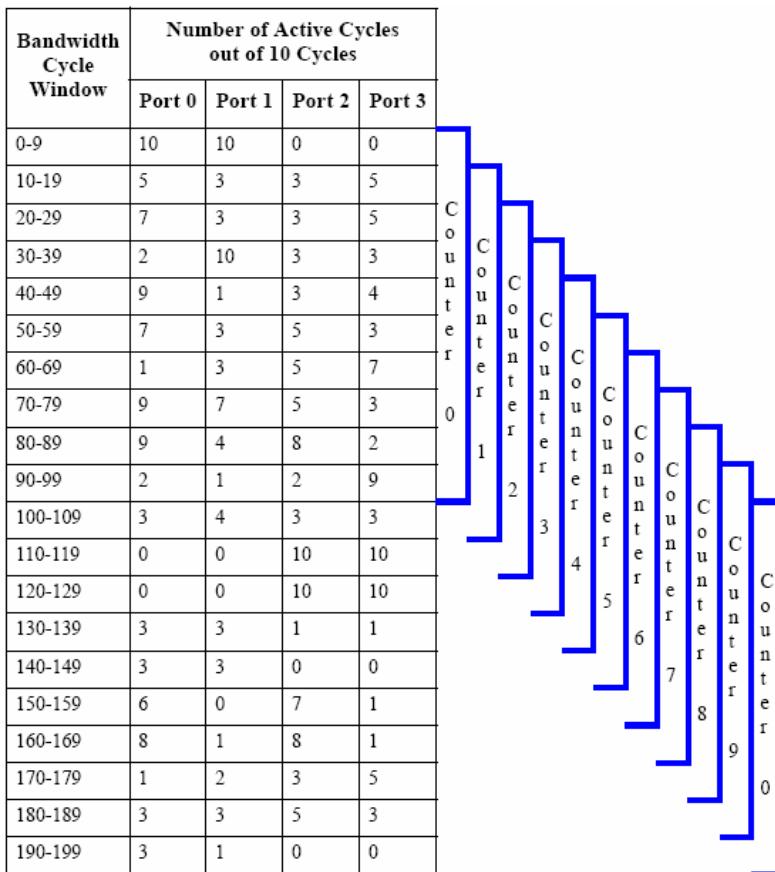
Each port has an associated bandwidth limit that sets the maximum percentage of the memory controller core bandwidth that the port is allowed to use. Once this level is reached, the Arbiter will no longer accept requests from this port until the bandwidth usage drops below the threshold. This scheme allows for the bandwidth to be shared between the ports. If required, an overflow option allows the port to continue to receive requests after the bandwidth limit has been reached. The bandwidth limits are stored in the programmable parameters `axiY_bdw` for each port Y at memory controller initialization. The `axiY_current_bdw` parameters are used to track the actual bandwidth utilized as computed by the bandwidth calculation module inside the Arbiter.

Port bandwidth is computed by counting the number of cycles that the memory controller core is busy actively processing that port's request in each 100 cycle period, referred to as the statistics window. In the memory controller, 10 counters are used for this computation. The counters track the number of active cycles in each statistics window, generating a moving average bandwidth value for each port. This is the actual bandwidth utilized value saved in the current bandwidth parameters (`axiY_current_bdw`). The values in the current bandwidth parameters are updated every 10 cycles with the actual bandwidth used in the last 100 cycles.

The memory controller core is defined as actively processing for a port if any of the following

situations occur:

- The memory controller core is ready to transfer write data from the port to memory, but the data has not arrived from the port.
- The memory controller core is holding the port's command and is ready to transfer to memory, but is waiting to open a bank, pre-charge a bank, or some other memory-related action.
- The memory controller core is actively transferring data from the port to memory.
- The memory controller core is ready to transfer read data from memory to the port but the port is busy and unable to accept the data. Note that if all ports are assigned 100% bandwidth, then bandwidth usage will not factor and arbitration will be purely based on priority.



For this system, the bandwidth will be monitored over each 100 cycles. The bandwidth calculation parameters will be updated every 10 cycles with the bandwidth usage of the last 100 cycles as shown graphically in above figure. The bandwidth totals are shown in below table.

Note that in the system, these values would be stored in the axiY_current_bdw parameters after each calculation.

Counter Number	Cycles Counting	Calculated Usage			
		Port 0	Port 1	Port 2	Port 3
0	0 ~ 99	61/100=61%	45/100=45%	37/100=37%	41/100=41%
1	10 ~ 109	54/100=54%	39/100=39%	40/100=40%	44/100=44%
2	20 ~ 119	49/100=49%	36/100=36%	47/100=47%	49/100=49%
3	30 ~ 129	42/100=42%	33/100=33%	54/100=54%	54/100=54%
4	40 ~ 139	43/100=43%	26/100=26%	52/100=52%	52/100=52%
5	50 ~ 149	37/100=37%	28/100=28%	49/100=49%	48/100=48%
6	60 ~ 159	36/100=36%	25/100=25%	51/100=51%	46/100=46%
7	70 ~ 169	43/100=43%	23/100=23%	54/100=54%	40/100=40%
8	80 ~ 179	35/100=35%	18/100=18%	52/100=52%	42/100=42%
9	90 ~ 189	29/100=29%	17/100=17%	49/100=49%	43/100=43%
0	100 ~ 199	30/100=30%	17/100=17%	47/100=47%	34/100=34%

Port Bandwidth Hold-Off

When the bandwidth used by a port exceeds its specified limit, that port is held off from subsequent arbitration decisions for a period of time known as the statistics reporting time. This causes a period of inactivity from that port, allowing the actual bandwidth used for that port to fall below the threshold. Since the bandwidth used is updated every ten cycles, the minimum hold off period for this system is ten cycles. This scheme is designed to constrain individual ports (especially ports programmed at higher priority) from overtaking all available bandwidth and locking out other ports. However, this can have its drawbacks. Consider a situation where only one port has been actively requesting and has therefore used all of its available bandwidth. The bandwidth hold-off will prevent additional requests from being accepted, even though no other ports are requesting. The memory controller core command queue will sit empty for several cycles while the hold-off is cleared. This is obviously wasted memory controller bandwidth and is detrimental to overall system performance. Memory controller has incorporated a bandwidth hold-off override function for such a situation in the axiY_bdw_ovflow parameters.

A port will be allowed to exceed its allocated bandwidth when all of these conditions are true:

1. The bandwidth overflow parameter (axiY_bdw_ovflow) is set to 'b1' for port Y.
2. No other port, whose bandwidth has not been exceeded, is requesting at the same priority level.
3. The command queue has less than the number of entries specified in the arb_cmd_q_threshold

parameter.

This last condition is a preventative measure to maintain latency requirements for ports programmed at higher priority. When a port is allowed to exceed bandwidth, it may fill the command queue with transactions. If this occurs, and a higher priority port starts requesting, then there will be no room in the command queue for the new requests. This means that the higher priority port will actually be held off for potentially several cycles. In this situation, even though the memory controller core bandwidth is being utilized well, the latency requirements of the higher priority port are not being met. The arb_cmd_q_threshold parameter is used to limit the bandwidth overflow and prevent this condition. It ensures that a certain number of slots remain available in the command queue for other ports. As a result, bandwidth overflow will be allowed as long as there are less than arb_cmd_q_threshold number of entries in the command queue.

Command Queue and Placement Logics

The memory controller core contains a command queue that accepts commands from the Arbiter. This command queue uses a placement algorithm to determine the order that commands will execute in the memory controller core. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at the time. Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. In addition, the placement logic attempts to maximize efficiency of the memory controller core through command grouping and bank splitting. Once placed into the command queue, the relative order of commands is constant. Many of the rules used in placement may be individually enabled/disabled. In addition, the queue may be disabled by clearing the placement_en parameter, resulting in an in-line queue that services requests in the order they are received. If the placement_en parameter is cleared to 'b0, the placement algorithm will be ignored.

Rules of the Placement Algorithm

The factors affecting command placement all work together to identify where a new command fits into the execution order. They are listed in order of importance.

1. Address Collision/Data Coherency Violation

The order in which read and write commands are processed in the memory controller is critical to proper system behavior. While reads and writes to different addresses are independent and may be re-ordered without affecting system performance, reads and writes that access the same

address are significantly related. If the port requests a read after a write to the same address, then repositioning the read before the write would return the original data, not the changed data. Similarly, if the read was requested ahead of the write but accidentally positioned after the write, then the read would return the new data, not the original data prior to being overwritten. These are significant data coherency mistakes. To avoid address collisions, reads or writes that access the same bank and row as a command already in the command queue will be inserted into the command queue after the original command, even if the new command is of a higher priority. This factor may be enabled/disabled through the `addr_cmp_en` parameter and should only be disabled if the system can guarantee coherency of reads and writes.

2. Source ID Collision

Each port is assigned a specific source ID that is a combination of the port and thread ID information, and identifies the source uniquely. This allows the memory controller to map data from/ to the correct source/destination. Note that a source ID does contain port identification information which means that the rules for placement are dependent on the requesting port.

There will not be source ID collisions between ports. In general, read commands from the same source ID will be placed in the command queue in order. Therefore, a read command with the same source ID as a read command already in the command queue will be processed after the original read command. All write commands from a port, even with different source IDs, will be executed in order. The behavior of commands of different types from the same source ID is dependent on the user configuration. For this memory controller, the placement of new read/write commands that collide in terms of source ID with existing entries in the command queue will only depend on other commands of the same type, not on different types. This means that, if there are no address conflicts, a read command could be executed ahead of a write command with the same source ID, and likewise a write command could be executed ahead of a read command with the same source ID. This feature will always be enabled.

3. Write Buffer Collision

Incoming write requests in the command queue are allocated to one of the 4 write buffers of the memory controller core automatically based on availability. New write commands will be designated to any available buffer. However, back-to-back write requests from a particular source ID will be allocated to the same write buffer as the previous command. Since the memory controller core must pull data out of the buffers in the order it was stored, if a write command is linked to a buffer that is associated with another command in the queue, then the new command will be placed in the command queue after that command, regardless of priority. This feature will

always be enabled.

4. Priority

Priorities are used to distinguish important commands from less important commands. Each command is given a priority based on the command type through the programmable parameters axiY_r_priority and axiY_w_priority (where Y represents the port number). A priority value of 0 is the highest priority, and a priority value of 3 is the lowest priority for this memory controller. The placement algorithm will attempt to place higher priority commands ahead of lower priority commands, as long as they have no source ID, write buffer or address collisions. Higher priority commands will be placed lower in the command queue if they access the same address, are from the same requestor or use the same buffer as lower priority commands already in the command queue. This feature is enabled through the priority_en parameter.

5. Bank Splitting

Before accesses can be made to two different rows within the same bank, the first active row must be closed (pre-charged) and the new row must be opened (activated). Both activities require some timing overhead; therefore, for optimization, the placement queue will attempt to insert the new command into the command queue such that commands to other banks may execute during this timing overhead. The placement of the new commands will still follow priority, source ID, write buffer and address collision rules. The placement logic will also attempt to optimize the memory controller core by inserting a command to the same bank as an existing command in the command queue immediately after the original command. This reduces the overall timing overhead by potentially eliminating one pre-charging/ activating cycle. This placement will only be possible if there are no priority, source ID, write buffer or address collisions or conflicts with other commands in the command queue. All bank splitting features are enabled through the bank_split_en parameter.

6. Read/Write Grouping

The memory suffers a small timing overhead when switching from read to write mode. For efficiency, the placement queue will attempt to place a new read command sequentially with other read commands in the command queue, or a new write command sequentially with other write commands in the command queue. Grouping will only be possible if no priority, source ID, write buffer or address collision rules are violated. This feature is enabled through the rw_same_en parameter.

Command Execution Order After Placement

Once a command has been placed in the command queue, its order relative to the other commands in the queue at that time is fixed. While this provides simplicity in the algorithm, there are drawbacks. For this reason, the memory controller offers two options that affect commands once they have been placed in the command queue.

1. Command Aging

Since commands can be inserted ahead of existing commands in the command queue, the situation could occur where a low priority command remains at the bottom of the queue indefinitely. To avoid such a lockout condition, aging counters have been included in the placement logic that measure the number of cycles that each command has been waiting. If command aging is enabled through the active_aging parameter, then if an aging counter hits its maximum, the priority of the associated command will be decremented by one (lower priority commands are executed first). This increases the likelihood that this command will move to the top of the command queue and be executed. Note that this command does not move relative positions in the command queue when it ages; the new priority will be considered when placing new commands into the command queue. Aging is controlled through a master aging counter and command aging counters associated with each command in the command queue. The age_count and command_age_count parameters hold the initial values for each of these counters, respectively. When the master counter counts down the age_count value, a signal is sent to the command aging counters to decrement. When the command aging counters have completely decremented, then the priority of the associated command is decremented by one number and the counter is reset. Therefore, a command does not age by a priority level until the total elapsed cycles has reached the product of the age_count and command_age_count values. The maximum number of cycles that any command can wait in the command queue until reaching the top priority level is the product of the age_count value, the command_age_count value, and the number of priority levels in the system.

2. High-Priority Command Swapping

Commands are assigned priority values to ensure that critical commands are executed more quickly in the memory controller than less important commands. Therefore, it is desirable that high-priority commands pass into the memory controller core as soon as possible. The placement algorithm takes priority into account when determining the order of commands, but still allows a scenario in which a high-priority command sits waiting at the top of the command queue while another command, perhaps of a lower priority, is in process. The high-priority command

swapping feature allows this new high-priority command to be executed more quickly. If the user has enabled the swapping function through the swap_en parameter, then the entry at the top of the command queue will be compared with the current command in progress. If the command queue's top entry is of a higher priority (not the same priority), and it does not have an address, source ID or write buffer conflict with the current command being executed, then the original command will be interrupted. For this memory controller, an additional check is performed before a read command is interrupted. If the read command in progress and the read command at the top of the command queue are from the same port, then the executing command will only be interrupted if the swap_port_rw_same_en parameter is set to 'b1. If this parameter is cleared to 'b0, a read command from the same port as a read command in progress, even with a higher priority and without any conflicts, would remain at the top of the command queue while the current command completes.

Note: All write commands from a single port, even with different source IDs, will be executed in order. Therefore, two write commands from the same port will never be swapped regardless of the settings of the swap_en and swap_port_rw_same_en parameters.

Note: Priorities are assigned to read commands based on the settings in the axiY_r_priority parameters. While all read commands from a port are assigned the same priority when placed in the command queue, their priorities may change over time through command aging. While uncommon, it is possible that a higher-priority read command may be at the top of the command queue while a lower-priority read command is executing. The behavior of the system in this scenario is based on the value of the swap_en and swap_port_rw_same_en parameters. Below table describes "Swapping Behavior":

Active Command Priority	New Command Priority	Originating Ports for Commands	Conflicts	Action
Higher	Lower	Same or Different	Yes or No	Current Command continues
Lower	Higher	Same	Yes	Current Command continues
Lower	Higher	Same	No	Will swap IF swap_en = 1 & swap_port_rw_same_en = 1
Lower	Higher	Different	No	Will swap IF swap_en = 1

If the command is to be interrupted, it will be halted after completing the current burst, stored and placed at the top of the queue, and the new command will be executed. As long as the command queue is not full, new commands may continue to be inserted into the command queue based on the placement rules, even at the head of the queue ahead of the interrupted command. The top entry in the command queue will be executed next. Whenever the interrupted command is resumed, it will start from the point at which it was interrupted. Note that priority 0 commands will never be interrupted, so the user should set any commands that should not be interrupted to priority 0. If supported by the port interfaces, setting the swap_port_rw_same_en parameter will enable interleaving.

Chapter 4 Static Memory Controller

Overview

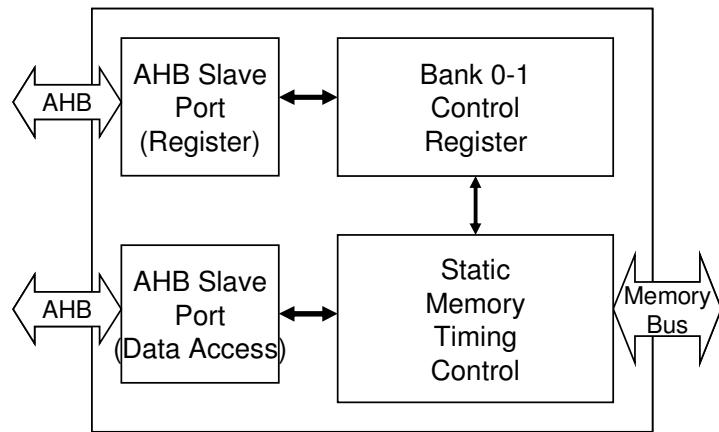
The memory controller provides a common AHB interface to external ROM, SRAM, Flash memories. The access timing of each memory type can be configurable through registers

Key Features

- Support static memory device including SRAM, ROM and NOR-Flash.
- 8, 16-bit wide static memory device support.
- Support Word, Halfword and Byte access.
- Static memory features include:
 - Programmable wait states
 - Bus turnaround delay
 - Output enable and write enable delays

Architecture

Block Diagram



Block Descriptions

The Static memory controller consist of one AHB control register port used for memory configuration and one AHB data access port used for AMBA component memory data access.

The Static Memory Controller generates static memory signals that support SRAM, ROM and NOR-Flash.

AHB Control Register Port

The control registers AHB slave port. It provides CPU or MCU to change the Static memory topology, size, type and timing parameters.

AHB Data Access Port

The AHB data access port is used for AMBA component memory data access. It can connect to AHB bus for one layer AHB architecture.

Memory Interface

The Static Memory Controller generates static memory signals that support SRAM, ROM and NOR-Flash.

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
ST0_TCEWD	0x8000	W	0x0000000F	The width of write CE
ST0_TCE2WE	0x8004	W	0x0000000F	The low of CE to the low of WE
ST0_TWEWD	0x8008	W	0x0000000F	The width of WE
ST0_TWE2CE	0x800C	W	0x0000000F	The high of WE to the high of CE
ST0_TCEWDR	0x8010	W	0x00000000	The width of read CE
ST0_TCE2RD	0x8014	W	0x0000000F	The low of CE to the low of RD
ST0_TRDWD	0x8018	W	0x00000015	The width of RD
ST0_TRD2CE	0x801C	W	0x00000000	The high of RD to the high of CE
ST0_BASIC	0x8020	W	0x00000000	The flash width size and write protect
ST1_TCEWD	0x9000	W	0x0000000F	The width of write CE
ST1_TCE2WE	0x9004	W	0x0000000F	The low of CE to the low of WE
ST1_TWEWD	0x9008	W	0x0000000F	The width of WE
ST1_TWE2CE	0x900C	W	0x0000000F	The high of WE to the high of CE

ST1_TCEWDR	0x9010	W	0x00000000	The width of read CE
ST1_TCE2RD	0x9014	W	0x0000000F	The low of CE to the low of RD
ST1_TRDWD	0x9018	W	0x0000000F	The width of RD
ST1_TRD2CE	0x901C	W	0x00000000	The high of RD to the high of CE
ST1_BASIC	0x9020	W	0x00000000	The flash width size and write protect

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

STx_TCEWD (x=0,1)

Address: Operational Base + offset (0x8000, 0x9000)

Static memory timing control register for write CE width

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0xF	Static memory timing control register for write CE width

STx_TCE2WE (x=0,1)

Address: Operational Base + offset (0x8004, 0x9004)

Static memory timing control register for low of CE to low of WE

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0xF	Static memory timing control register for low of CE to low of WE

STx_TWEWD (x=0,1)

Address: Operational Base + offset (0x8008, 0x9008)

Static memory timing control register for WE width

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0xF	Static memory timing control register for WE width

STx_TWE2CE (x=0,1)

Address: Operational Base + offset (0x800C, 0x900C)

Static memory timing control register for high of WE to high of CE

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0xF	Static memory timing control register for high of WE to high of CE

STx_TCEWDR (x=0,1)

Address: Operational Base + offset (0x8010, 0x9010)

Static memory timing control register for read CE width

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Static memory timing control register for read CE width

STx_TCE2RD (x=0,1)

Address: Operational Base + offset (0x8014, 0x9014)

Static memory timing control register for low of CE to low of RD

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0xF	Static memory timing control register for low of CE to low of RD

STx_TRDWD (x=0,1)

Address: Operational Base + offset (0x8018, 0x9018)

Static memory timing control register for RD width

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	X=0, 0x15 X=1, 0xF	Static memory timing control register for RD width

STx_TRD2CE (x=0,1)

Address: Operational Base + offset (0x801C, 0x901C)

Static memory timing control register for high of RD to high of CE

Bit	Attr	Reset Value	Description

31:8	-	-	Reserved.
7:0	RW	0x0	Static memory timing control register for high of RD to high of CE

STx_BASIC (x=0,1)

Address: Operational Base + offset (0x8020, 0x9020)

Static memory basic setting

Bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5	RW	0x0	Write protect bit0: Writable 1: Write protect
4:2	-	-	Reserved.
1:0	RW	0x0	Static memory data size 0x0 : 8 bits 0x1 : 16 bits 0x2 : 32 bits 0x3 : Reserved. (for x = 0, these two bits are reserved. Use has to use the hardware input signal “ST_ROMSZ” to configure static memory data size 0x0: 8bits 0x1:16bits)

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

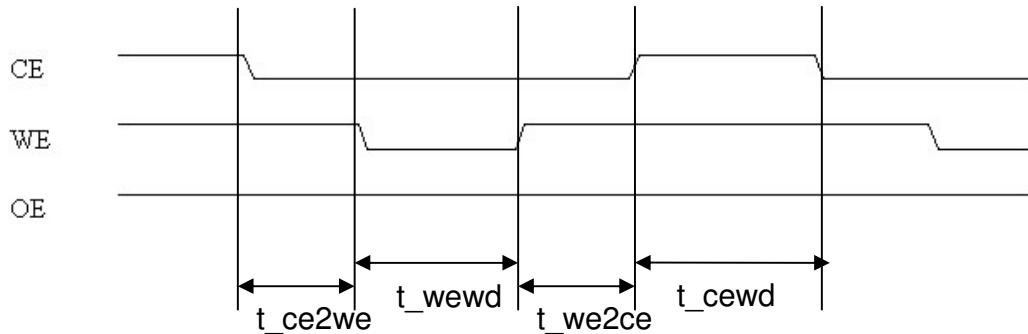
AHB read/ write from Flash/ROM

The FLASH/ROM/SRAM memory devices are asynchronous devices that do not use the clock input. In the following timing diagram, the clock signal is provided as a reference to the internal operation of the core.

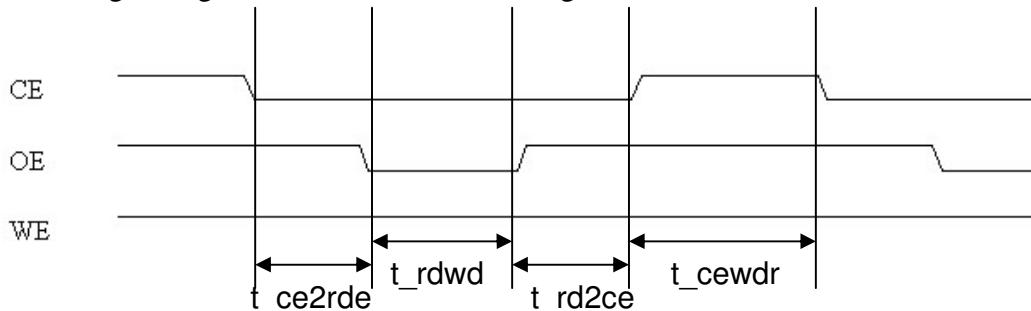
Memory access is started by the address (ADDR) and chip select signal. Once asserted, the memory will provide the read/write data after the access time. The memory manufacturer in unit of nanosecond provides access time. The memory controller can be programmed to provide access time based on clock cycle. For example, if system controller is running at 100MHz (cycle time: 10ns) and access time of the memory is 40ns (including clock to output delay and data

input setup time), the number of memory read/write access delay should be 4. User must make sure that the clock to output delay of all the address and control signals and the data input setup time to the core are include in the access time computation.

The writing timing is described in the following:



The reading timing is described in the following:



The static memory controller uses the CE, OE and WE to control the static memory (ex. Flash, Static RAM).

The above diagram shows four consecutive read to the memory. In this example, the memory is 16-bit wide and the user interface is 32-bit. Each 32-bit read by request by the user is executed as 2 consecutive 16-bit read to the memory. The memory controller increments the address for each read. After all four read accesses are completed; the memory controller returns one 32-bit word to the requestor.

Chapter 5 NAND Flash Controller

Overview

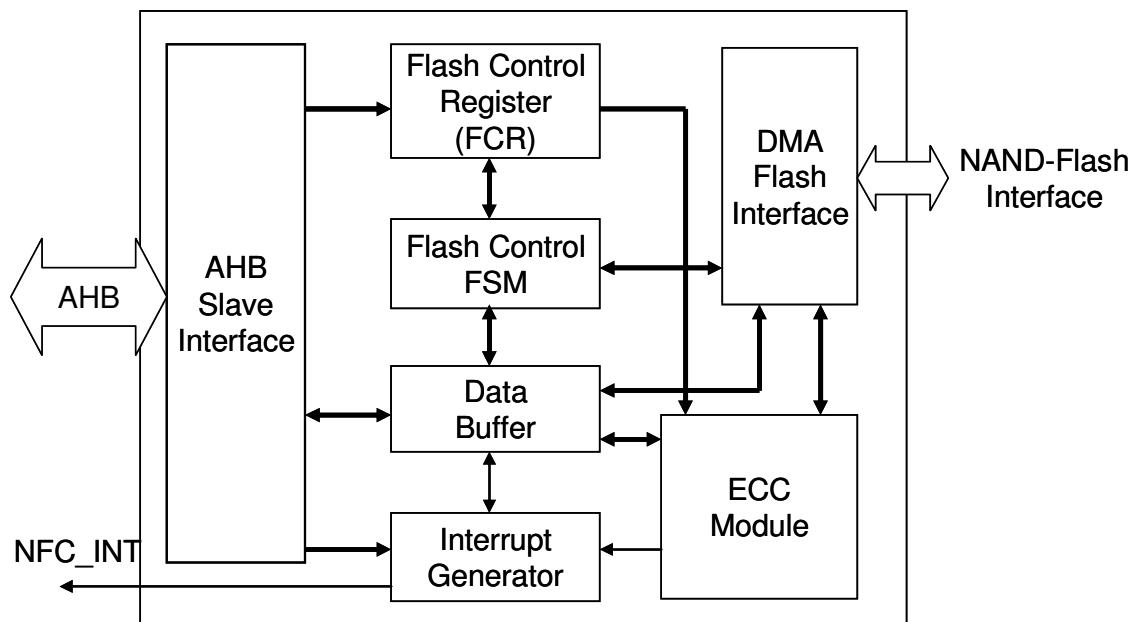
The NAND Flash controller provides a hardware solution to access the NAND type Flash memory. For reusability and performance, the flash controller includes AHB slave port, flash control register (FCR), controller FSM , hardware ECC module , flash DMA , and internal buffer and Boot Rom .

Key Features

- Configurable access time
- Configurable command mode
- Hardware ECC module (6 symbol correct/512 Bytes)
- Internal SRAM buffer to enhance performance (Buff size is 2k bytes)

Architecture

Block Diagram



Block Descriptions

AHB Slave Interface

This block is the interface between AMBA AHB and NAND Flash Controller Slave port.

Flash Controller Register

Accept the command from host microprocessor and control DMA Flash I/F operation.

Buffer

Built-in 512x32 data buffer.

DMA Flash I/F

Controlled by Flash Controller Register and generate signal to NAND Flash Device

ECC Module

Error correcting code module, generate error correcting code in data writer operation and detect error data in data read operation.

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
NFC_BUF	0x8000 ~0x87FF	W	0x00000000	2K-Byte internal buffer.
NFC_CONFIG	0xC000	W	0x001FFF80	Flash type configuration register.
NFC_CMD1	0xC004	W	0x00000190	First Command input register.
NFC_CMD2	0xC008	W	0x00000000	Second Command input register.
NFC_SCMD	0xC00C	W	0x00000000	Read Status Command input register.

NFC_ADDR1	0xC010	W	0x00000100	1 st Cycle address input register.
NFC_ADDR2	0xC014	W	0x00000000	2 nd Cycle address input register.
NFC_ADDR3	0xC018	W	0x00000000	3 rd Cycle address input register.
NFC_ADDR4	0xC01C	W	0x00000000	4 th Cycle address input register.
NFC_ADDR5	0xC020	W	0x00000000	5 th Cycle address input register.
NFC_DATA	0xC024	W	0x00000000	Access Data register.
NFC_BUF_SADD R	0xC028	W	0x00000000	Internal buffer start address register.
NFC_BUF_CNT	0xC02C	W	0x00000003	Internal buffer data count register.
NFC_DMA_SET	0xC034	W	0x00000000	DMA operation setting register.
NFC_CEWP	0xC038	W	0x00000001	Chip enable/Write protect register.
NFC_CTRL	0xC03C	W	0x00000000	NAND Flash Controller control register.
NFC_RESET	0xC040	W	0x00000000	NAND Flash Controller reset register.
NFC_STATE	0xC044	W	0x00000000	NAND Flash Controller status register.
NFC_INTS	0xC04C	W	0x00000000	NAND Flash Controller interrupt status register.
NFC_GPIO	0xC050	W	0x00000000	GPIO control register
NFC_1st_ECC_1	0xC058	W	0x00000000	1st 512 byte ECC_1 code register
NFC_1st_ECC_2	0xC05C	W	0x00000000	1st 512 byte ECC_2 code register
NFC_1st_ECC_3	0xC060	W	0x00000000	1st 512 byte ECC_3 code register
NFC_1st_ECC_4	0xC064	W	0x00000000	1st 512 byte ECC_4 code register
NFC_2nd_ECC_1	0xC068	W	0x00000000	2nd 512 byte ECC_1 code register. (Big page only)
NFC_2nd_ECC_2	0xC06C	W	0x00000000	2nd 512 byte ECC_2 code register. (Big page only)

NFC_2nd_ECC_3	0xC070	W	0x00000000	2nd 512 byte ECC_3 code register. (Big page only)
NFC_2nd_ECC_4	0xC074	W	0x00000000	2nd 512 byte ECC_4 code register. (Big page only)
NFC_3rd_ECC_1	0xC078	W	0x00000000	3rd 512 byte ECC_1 code register. (Big page only)
NFC_3rd_ECC_2	0xC07C	W	0x00000000	3rd 512 byte ECC_2 code register. (Big page only)
NFC_3rd_ECC_3	0xC080	W	0x00000000	3rd 512 byte ECC_3 code register. (Big page only)
NFC_3rd_ECC_4	0xC084	W	0x00000000	3rd 512 byte ECC_4 code register. (Big page only)
NFC_4th_ECC_1	0xC088	W	0x00000000	4th 512 byte ECC_1 code register. (Big page only)
NFC_4th_ECC_2	0xC08C	W	0x00000000	4th 512 byte ECC_2 code register. (Big page only)
NFC_4th_ECC_3	0xC090	W	0x00000000	4th 512 byte ECC_3 code register. (Big page only)
NFC_4th_ECC_4	0xC094	W	0x00000000	4th 512 byte ECC_4 code register. (Big page only)
NFC_1SYND1	0xC098	W	0x00000000	1 st 512 byte Syndrome_1 code register. (Big page only)
NFC_1SYND2	0xC09C	W	0x00000000	1 st 512 byte Syndrome_2 code register.
NFC_1SYND3	0xC0A0	W	0x00000000	1 st 512 byte Syndrome_3 code register.
NFC_1SYND4	0xC0A4	W	0x00000000	1 st 512 byte Syndrome_4 code register.
NFC_2SYND1	0xC0A8	W	0x00000000	2 nd 512 byte Syndrome_1 code register.
NFC_2SYND2	0xC0AC	W	0x00000000	2 nd 512 byte Syndrome_2 code register. (Big page only)
NFC_2SYND3	0xC0B0	W	0x00000000	2 nd 512 byte Syndrome_3 code register. (Big page only)

NFC_2SYND4	0xC0B4	W	0x00000000	2 nd 512 byte Syndrome_4 code register. (Big page only)
NFC_3SYND1	0xC0B8	W	0x00000000	3 rd 512 byte Syndrome_1 code register. (Big page only)
NFC_3SYND2	0xC0BC	W	0x00000000	3 rd 512 byte Syndrome_2 code register. (Big page only)
NFC_3SYND3	0xC0C0	W	0x00000000	3 rd 512 byte Syndrome_3 code register. (Big page only)
NFC_3SYND4	0xC0C4	W	0x00000000	3 rd 512 byte Syndrome_4 code register. (Big page only)
NFC_4SYND1	0xC0C8	W	0x00000000	4 th 512 byte Syndrome_1 code register. (Big page only)
NFC_4SYND2	0xC0CC	W	0x00000000	4 th 512 byte Syndrome_2 code register. (Big page only)
NFC_4SYND3	0xC0D0	W	0x00000000	4 th 512 byte Syndrome_3 code register. (Big page only)
NFC_4SYND4	0xC0D4	W	0x00000000	4 th 512 byte Syndrome_4 code register. (Big page only)

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

NFC_CONFIG

NAND Flash type configure register. This register defines flash read/write cycle AC timing and support flash memory type, support NAND-flash device number.

Bit	Attr	Reset Value	Description												
31:21	-	-	Reserved.												
20:17	RW	0xF	<p>tWC: Write Cycle time.</p> <table> <tr> <td>0000 : 1 Cycle</td> <td>1000 : 9 Cycle</td> </tr> <tr> <td>0001 : 2 Cycle</td> <td>1001 : 10 Cycle</td> </tr> <tr> <td>0010 : 3 Cycle</td> <td>1010 : 11 Cycle</td> </tr> <tr> <td>0011 : 4 Cycle</td> <td>1011 : 12 Cycle</td> </tr> <tr> <td>0100 : 5 Cycle</td> <td>1100 : 13 Cycle</td> </tr> <tr> <td>0101 : 6 Cycle</td> <td>1101 : 14 Cycle</td> </tr> </table>	0000 : 1 Cycle	1000 : 9 Cycle	0001 : 2 Cycle	1001 : 10 Cycle	0010 : 3 Cycle	1010 : 11 Cycle	0011 : 4 Cycle	1011 : 12 Cycle	0100 : 5 Cycle	1100 : 13 Cycle	0101 : 6 Cycle	1101 : 14 Cycle
0000 : 1 Cycle	1000 : 9 Cycle														
0001 : 2 Cycle	1001 : 10 Cycle														
0010 : 3 Cycle	1010 : 11 Cycle														
0011 : 4 Cycle	1011 : 12 Cycle														
0100 : 5 Cycle	1100 : 13 Cycle														
0101 : 6 Cycle	1101 : 14 Cycle														

			0110 : 7 Cycle 0111 : 8 Cycle	1110 : 15 Cycle 1111 : 16 Cycle
16:14	RW	0X7	tWP: WE Pulse time , duration of write pulse. 000 : 1 Cycle 001 : 2 Cycle 010 : 3 Cycle 011 : 4 Cycle	100 : 5 Cycle 101 : 6 Cycle 110 : 7 Cycle 111 : 8 Cycle
13:10	RW	0xF	tRC: Read Cycle time. 0000 : 1 Cycle 0001 : 2 Cycle 0010 : 3 Cycle 0011 : 4 Cycle 0100 : 5 Cycle 0101 : 6 Cycle 0110 : 7 Cycle 0111 : 8 Cycle	1000 : 9 Cycle 1001 : 10 Cycle 1010 : 11 Cycle 1011 : 12 Cycle 1100 : 13 Cycle 1101 : 14 Cycle 1110 : 15 Cycle 1111 : 16 Cycle
9:7	RW	0X7	tRP: RE Pulse time, duration of read pulse. 000 : 1 Cycle 001 : 2 Cycle 010 : 3 Cycle 011 : 4 Cycle	100 : 5 Cycle 101 : 6 Cycle 110 : 7 Cycle 111 : 8 Cycle
6:5	-	-	Reserved.	
4:3	RW	0x0	NAND-Flash memory page size. 00 : 512 Bytes 01 : 2K Bytes	10 : Reserved 11 : Reserved
2:0	-	-	Reserved.	

Note1: Read/Write Cycle time (tRC/tWC) and RE/WE (tRP/tWP) Pulse time must reference specification of NAND Flash device.

Note2: RE Pulse time (tRP) must more than RE access time (tREA).

NFC_CMD1

NAND Flash first command input register. This register defines first command for NAND-Flash memory.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.

8	RW	0x1	Command valid Bit (set by S/W ,clear by H/W) 0: Command is not valid 1: Command is valid
7:0	RW	0x90	NAND Flash memory 1 st Command Input Port

NFC_CMD2

NAND Flash second command input register. This register defines second command for NAND-Flash memory.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	RW	0x0	Command valid Bit (set by S/W ,clear by H/W) 0: Command not valid 1: Command is valid
7:0	RW	0x0	NAND Flash memory 2 nd Command input port

NFC_SCMD

NAND Flash read status command input register. This register defines command for NANA-Flash memory Read Status operation.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	RW	0x0	Command valid Bit (set by S/W ,clear by H/W) 0: Command not valid 1: Command is valid
7:0	RW	0x0	NAND Flash memory Read Status Command input port

NFC_ADDR1

NAND Flash address input register. This register defines 1st cycle address for operation.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x1	Address valid Bit (set by S/W ,clear by H/W) 0: Address not valid 1: Address is valid
7:0	RW	0x0	NAND Flash memory address input port

NFC_ADDR2

NAND Flash address input register. This register defines 2nd cycle address for operation.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	Address valid Bit (set by S/W ,clear by H/W) 0: Address not valid. 1: Address is valid.
7:0	RW	0x0	NAND Flash memory address input port

NFC_ADDR3

NAND Flash address input register. This register defines 3rd cycle address for operation.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	Address valid Bit (set by S/W ,clear by H/W) 0: Address not valid. 1: Address is valid.
7:0	RW	0x0	NAND Flash memory address input port

NFC_ADDR4

NAND Flash address input register. This register defines 4th cycle address for operation.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	Address valid Bit (set by S/W ,clear by H/W) 0: Address not valid. 1: Address is valid.
7:0	RW	0x0	NAND Flash memory address input port

NFC_ADDR5

NAND Flash address input register. This register defines 5th cycle address for operation.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	Address valid Bit (set by S/W ,clear by H/W) 0: Address not valid. 1: Address is valid.
7:0	RW	0x0	NAND Flash memory address input port

NFC_DATA

Access data register. This register offers a data port for S/W direct access flash memory.

Only READ-ID operation is used.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Access data port.

NFC_BUF_SADDR

Internal buffer start address register. This register defines start address for internal SRAM buffer read/write.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8:0	W	0x0	Internal buffer start address.

NFC_BUF_CNT

Internal buffer data cont register.

Bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	W	0x0	Data access with/without ECC code. 0: Access without ECC code 1: Access with ECC code
15:0	W	0x03	Internal buffer read/write data number.

NFC_DMA_SET

DMA operation setting register. This register defines flash controller DMA operation mode and data path.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	RW	0x0	Endian mode set 0: Little endian 1: Big endian
10	RW	0x0	ECC test mode 0: Disable ECC codec test mode, data path is normal 1: Enable ECC codec test mode, data path not to

NAND-Flash			
9	RW	0x0	Buffer read/write enable register. 0: Disable 1: Enable
8	RW	0x0	Second command input control register 0: No second command 1: Second command
7	RW	0x0	Auto status check enable register 0: Disable 1: Enable
6	RW	0x0	Access data path select register 0: Access data from/to internal buffer. (normal access) 1: Access data from/to data register. (Read-ID operation only)
5:4	RW	0x0	I/O Operation Type 01: Read operation 10: Write operation Other: Reserved.
3:1	-	-	Reserved.
0	RW	0x0	Device wait R/B enable register. 1: Wait R/B 0: Not Wait R/B

NFC_CEWP

Flash Chip Enable/Write Protect. This register defines which NAND-Flash device has been selected and which device has been protected.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7	RW	0x0	NAND Flash device 3 Write Protect enable register. 0: Disable Write Protect 1: Enable Write Protect
6	RW	0x0	NAND Flash device 2 Write Protect enable register. 0: Disable Write Protect 1: Enable Write Protect
5	RW	0x0	NAND Flash device 1 Write Protect enable register. 0: Disable Write Protect 1: Enable Write Protect
4	RW	0x0	NAND Flash device 0 Write Protect enable register. 0: Disable Write Protect 1: Enable Write Protect
3	RW	0x0	NAND Flash device 3 Chip Enable 0: Chip Disable 1: Chip Enable
2	RW	0x0	NAND Flash device 2 Chip Enable 0: Chip Disable 1: Chip Enable
1	RW	0x0	NAND Flash device 1 Chip Enable 0: Chip Disable 1: Chip Enable
0	RW	0x1	NAND Flash device 0 Chip Enable 0: Chip Disable 1: Chip Enable

NFC_CTRL

Control register of NAND Flash controller control register. This register defines NAND Flash controller operation enable.

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2	RW	0x0	Flash DMA enable 0: Disable flash DMA 1: Enable flash DMA
1	RW	0x0	Access data enable 0: Disable to Read/Write Data 1: Enable to Read/Write Data
0	RW	0x0	Flash Command/Address input enable 0: Disable flash send command with/without address 1: Enable flash send command with/without address

NFC_RESET

NAND Flash controller reset register.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	W	0x0	NAND Flash controller reset enable 0: Disable Reset 1: Enable Reset

NFC_STATE

NAND Flash device status register. This register recorded R/B pin signal of every device, if R/B =0 expresses that device is busy, if R/B =1 expresses device is ready.

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3	R	0x0	NAND Flash Device 3 Ready/Busy 0: Device 3 R/B = 1'b0 1: Device 3 R/B = 1'b1
2	R	0x0	NAND Flash Device 2 Ready/Busy 0: Device 2 R/B = 1'b0 1: Device 2 R/B = 1'b1
1	R	0x0	NAND Flash Device 1 Ready/Busy 0: Device 1 R/B = 1'b0 1: Device 1 R/B = 1'b1
0	R	0x0	NAND Flash Device 0 Ready/Busy 0: Device 0 R/B = 1'b0 1: Device 0 R/B = 1'b1

NFC_INTS

NAND Flash controller interrupt status register.

Bit	Attr	Reset Value	Description
31:12	RW	0x0	Reserved
11:8	RW	0x0	Syndrome error location
7	RW	0x0	Block data is ERASE
6	RW	0x0	Block Erase Done
5	RW	0x0	Block Erase Fail

4	RW	0x0	Read Data Done
3	RW	0x0	Syndrome error
2	RW	0x0	Time Out
1	RW	0x0	Write Data Done
0	RW	0x0	Write Data Error

NFC_GPIO

GPIO control register. This register define the direction of general purpose input/output(NFC_GPIO) pin .

Bit #	Access	Reset	Description
31:1	RW	0x0	Reserved
0	RW	0x0	General Purpose Input/Output

NFC_1st_ECC_1

1st 512byte ECC code register 1 . The register NFC_1st_ECC_1~4 are recorded 1st 512 bytes ECC code. There registers are read only.

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	1 st 512 byte ECC_3 Code
19:10	R	0x0	1 st 512 byte ECC_2 Code
9:0	R	0x0	1 st 512 byte ECC_1 Code

NFC_1st_ECC_2

1st 512byte ECC code register 2 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	1 st 512 byte ECC_6 Code
19:10	R	0x0	1 st 512 byte ECC_5 Code
9:0	R	0x0	1 st 512 byte ECC_4 Code

NFC_1st_ECC_3

1st 512byte ECC code register 3 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	1 st 512 byte ECC _9 Code
19:10	R	0x0	1 st 512 byte ECC _8 Code
9:0	R	0x0	1 st 512 byte ECC _7 Code

NFC_1st_ECC_4

1st 512byte ECC code register 4 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	1 st 512 byte ECC _12 Code
19:10	R	0x0	1 st 512 byte ECC _11 Code
9:0	R	0x0	1 st 512 byte ECC _10 Code

NFC_2nd_ECC_1 (Big page only)

2nd 512byte ECC code register 1 . The register NFC_2nd_ ECC _1~4 are recorded 2nd 512 bytes ECC code. These registers are read only.

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	2 nd 512 byte ECC _3 Code
19:10	R	0x0	2 nd 512 byte ECC _2 Code
9:0	R	0x0	2 nd 512 byte ECC _1 Code

NFC_2nd_ECC_2 (Big page only)

2nd 512byte ECC code register 2 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	2 nd 512 byte ECC _6 Code
19:10	R	0x0	2 nd 512 byte ECC _5 Code

9:0	R	0x0	2 nd 512 byte ECC _4 Code
-----	---	-----	--------------------------------------

NFC_2nd_ECC_3 (Big page only)

2nd 512byte ECC code register 3 .

Bit #	Acces s	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	2 nd 512 byte ECC _9 Code
19:10	R	0x0	2 nd 512 byte ECC _8 Code
9:0	R	0x0	2 nd 512 byte ECC _7 Code

NFC_2nd_ECC_4 (Big page only)

2nd 512byte ECC code register 4 .

Bit #	Acces s	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	2 nd 512 byte ECC _12 Code
19:10	R	0x0	2 nd 512 byte ECC _11 Code
9:0	R	0x0	2 nd 512 byte ECC _10 Code

NFC_3rd_ECC_1 (Big page only)

3rd 512byte ECC code register 1 . The register NFC_3rd_ ECC _1~4 are recorded 3rd 512 bytes ECC code. There registers are read only.

Bit #	Acces s	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	3 rd 512 byte ECC _3 Code
19:10	R	0x0	3 rd 512 byte ECC _2 Code
9:0	R	0x0	3 rd 512 byte ECC _1 Code

NFC_3rd_ECC_2 (Big page only)

3rd 512byte ECC code register 2 .

Bit #	Acces	Reset	Description

	s		
31:30	R	0x0	Reserved
29:20	R	0x0	3 rd 512 byte ECC _6 Code
19:10	R	0x0	3 rd 512 byte ECC _5 Code
9:0	R	0x0	3 rd 512 byte ECC _4 Code

NFC_3rd_ECC_3 (Big page only)

3rd 512byte ECC code register 3 .

Bit #	Acces s	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	3 rd 512 byte ECC _9 Code
19:10	R	0x0	3 rd 512 byte ECC _8 Code
9:0	R	0x0	3 rd 512 byte ECC _7 Code

NFC_3rd_ECC_4 (Big page only)

3rd 512byte ECC code register 4 .

Bit #	Acces s	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	3 rd 512 byte ECC _12 Code
19:10	R	0x0	3 rd 512 byte ECC _11 Code
9:0	R	0x0	3 rd 512 byte ECC _10 Code

NFC_4th_ECC_1 (Big page only)

4th 512byte ECC code register 1 . The register NFC_4th _ ECC _1~4 are recorded 4th 512 bytes ECC code. There registers are read only.

Bit #	Acces s	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	4 th 512 byte ECC _3 Code
19:10	R	0x0	4 th 512 byte ECC _2 Code
9:0	R	0x0	4 th 512 byte ECC _1 Code

NFC_4th_ECC_2 (Big page only)

4th 512byte ECC code register 2 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	4 th 512 byte ECC _6 Code
19:10	R	0x0	4 th 512 byte ECC _5 Code
9:0	R	0x0	4 th 512 byte ECC _4 Code

NFC_4th_ECC_3 (Big page only)

4th 512byte ECC code register 3 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	4 th 512 byte ECC _9 Code
19:10	R	0x0	4 th 512 byte ECC _8 Code
9:0	R	0x0	4 th 512 byte ECC _7 Code

NFC_4th_ECC_4 (Big page only)

4th 512byte ECC code register 4 .

Bit #	Access	Reset	Description
31:30	R	0x0	Reserved
29:20	R	0x0	4 th 512 byte ECC _12 Code
19:10	R	0x0	4 th 512 byte ECC _11 Code
9:0	R	0x0	4 th 512 byte ECC _10 Code

NFC_1SYND1

1st 512byte Syndrome code register 1.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	1 st 512 byte Syndrome_3 Code.
19:10	R	0x0	1 st 512 byte Syndrome_2 Code.

9:0	R	0x0	1 st 512 byte Syndrome_1 Code.
-----	---	-----	---

NFC_1SYND2

1st 512byte Syndrome code register 2.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	1 st 512 byte Syndrome_6 Code.
19:10	R	0x0	1 st 512 byte Syndrome_5 Code.
9:0	R	0x0	1 st 512 byte Syndrome_4 Code.

NFC_1SYND3

1st 512byte Syndrome code register 3.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	1 st 512 byte Syndrome_9 Code.
19:10	R	0x0	1 st 512 byte Syndrome_8 Code.
9:0	R	0x0	1 st 512 byte Syndrome_7 Code.

NFC_1SYND4

1st 512byte Syndrome code register 4.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	1 st 512 byte Syndrome_12 Code.
19:10	R	0x0	1 st 512 byte Syndrome_11 Code.
9:0	R	0x0	1 st 512 byte Syndrome_10 Code.

NFC_2SYND1 (Big page only)

2nd 512byte Syndrome code register 1.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	2 nd 512 byte Syndrome_3 Code.
19:10	R	0x0	2 nd 512 byte Syndrome_2 Code.
9:0	R	0x0	2 nd 512 byte Syndrome_1 Code.

NFC_2SYND2 (Big page only)

2nd 512byte Syndrome code register 2.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	2 nd 512 byte Syndrome_6 Code.
19:10	R	0x0	2 nd 512 byte Syndrome_5 Code.
9:0	R	0x0	2 nd 512 byte Syndrome_4 Code.

NFC_2SYND3 (Big page only)

2nd 512byte Syndrome code register 3.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	2 nd 512 byte Syndrome_9 Code.
19:10	R	0x0	2 nd 512 byte Syndrome_8 Code.
9:0	R	0x0	2 nd 512 byte Syndrome_7 Code.

NFC_2SYND4 (Big page only)

2nd 512byte Syndrome code register 4.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	2 nd 512 byte Syndrome_12 Code.
19:10	R	0x0	2 nd 512 byte Syndrome_11 Code.
9:0	R	0x0	2 nd 512 byte Syndrome_10 Code.

NFC_3SYND1 (Big page only)

3rd 512byte Syndrome code register 1.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	3 rd 512 byte Syndrome_3 Code.
19:10	R	0x0	3 rd 512 byte Syndrome_2 Code.
9:0	R	0x0	3 rd 512 byte Syndrome_1 Code.

NFC_3SYND2 (Big page only)

3rd 512byte Syndrome code register 2.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	3 rd 512 byte Syndrome_6 Code.
19:10	R	0x0	3 rd 512 byte Syndrome_5 Code.
9:0	R	0x0	3 rd 512 byte Syndrome_4 Code.

NFC_3SYND3 (Big page only)

3rd 512byte Syndrome code register 3.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	3 rd 512 byte Syndrome_9 Code.
19:10	R	0x0	3 rd 512 byte Syndrome_8 Code.
9:0	R	0x0	3 rd 512 byte Syndrome_7 Code.

NFC_3SYND4 (Big page only)

3rd 512byte Syndrome code register 4.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	3 rd 512 byte Syndrome_12 Code.
19:10	R	0x0	3 rd 512 byte Syndrome_11 Code.
9:0	R	0x0	3 rd 512 byte Syndrome_10 Code.

NFC_4SYND1 (Big page only)

4th 512byte Syndrome code register 1.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	4 th 512 byte Syndrome_3 Code.
19:10	R	0x0	4 th 512 byte Syndrome_2 Code.
9:0	R	0x0	4 th 512 byte Syndrome_1 Code.

NFC_4SYND2 (Big page only)

4th 512byte Syndrome code register 2.

Bit	Attr	Reset Value	Description

31:30	-	-	Reserved.
29:20	R	0x0	4 th 512 byte Syndrome_6 Code.
19:10	R	0x0	4 th 512 byte Syndrome_5 Code.
9:0	R	0x0	4 th 512 byte Syndrome_4 Code.

NFC_4SYND3 (Big page only)

4th 512byte Syndrome code register 3.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	4 th 512 byte Syndrome_9 Code.
19:10	R	0x0	4 th 512 byte Syndrome_8 Code.
9:0	R	0x0	4 th 512 byte Syndrome_7 Code.

NFC_4SYND4 (Big page only)

4th 512byte Syndrome code register 4.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:20	R	0x0	4 th 512 byte Syndrome_12 Code.
19:10	R	0x0	4 th 512 byte Syndrome_11 Code.
9:0	R	0x0	4 th 512 byte Syndrome_10 Code.

Notes: Attr: RW – Read/writable, R – Read only, W – Write only

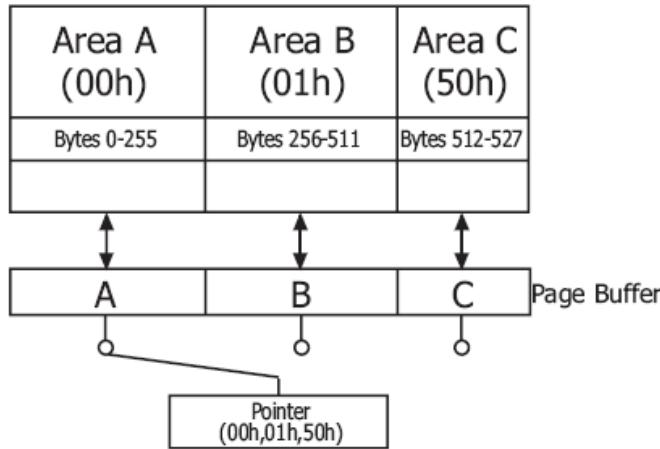
Functional Description

■ Pointer control of Page-Program operation

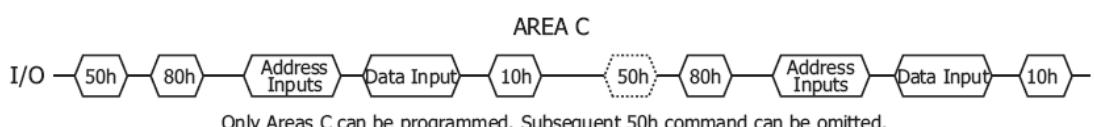
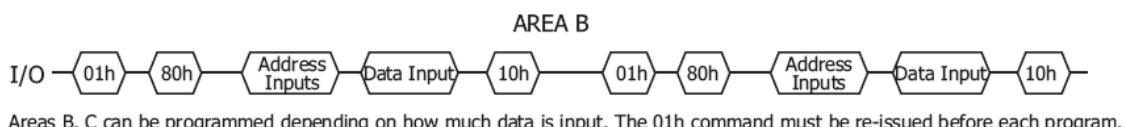
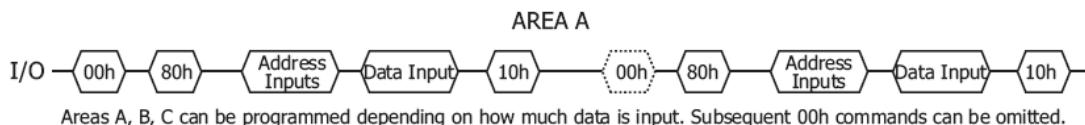
Small page NAND-Flash has three address pointer commands as a substitute for the two most significant column addresses. '00h'

command sets the pointer to 'A' area (0~255byte), '01h' command sets the pointer to 'B' area (256~511byte), and '50h' command sets

the pointer to 'C' area (512~527byte). The pointer operation as below.



With these commands, the starting column address can be set to any of a whole page (0~527byte). '00h' or '50h' is sustained until another address pointer command is inputted. '01h' command, however, is effective only for one operation. After any operation of Read, Program, Erase, Reset, Power_Up is executed once with '01h' command, the address pointer returns to 'A' area by itself. To program data starting from 'A' or 'C' area, '00h' or '50h' command must be inputted before '80h' command is written. A complete read operation prior to '80h' command is not necessary. To program data starting from 'B' area, '01h' command must be inputted right before '80h' command is written. The command input sequence of Page-Program operation as below.



■ NAND-Flash Controller Data Access Sequence

Small Page (512bytes) NAND-Flash



- Date: NAND Controller IP access data 512bytes
- ECC: ECC code for access data
- RE: reserved for S/W use
-

Large Page (2048bytes) NAND-Flash

- 
- Date: NAND Controller IP access data 512bytes
- ECC: ECC code for access data
- RE: reserved for S/W use

NAND Flash Reset Sequence

- 0. Write REST command to NFC_CMD1 register, and set NFC_CMD1[8]='1' to express this is valid command.
- 1. Set NFC_CTRL[2] ='1' (flash DMA enable) to enable NAND Flash controller send REST command to flash device.
- 2. Wait NFC_STATE register bit_x='1' (selected device R/B='1'), that device has finished RESET operation.
-

Page-Program Operation (Small Page Device)

When write data to flash device, host must enable controller to execute Program operation.

Program operation flow as below:

Write data into internal buffer.

0. Set NFC_CONFIG register to define read/write cycle timing and flash device basic setting.
1. Write Pointer command to NFC_CMD1 register and set valid bit='1' to express command is valid.
2. Set NFC_DMA_SET [9:0] ='021', NFC_BUF_CNT = '0', NFC_CTRL bit [2:0] ='101' to start Point-control operation.
3. Wait interrupt signal is asserted, indicate complete Pointer-control operation.
4. Write Program command to NFC_CMD1, NFC_CMD2 and NFC_SCMD register, write start address to NFC_ADDR1~5, and set valid bit='1' to express command and address are valid.
5. If normal Program operation, set NFC_DMA_SET[9:0] ='3a1' to enable Buffer

Read/Write, Second command input, Auto Status Check, DMA operation is Write,Access Data Path is internal buffer and Device 0 Wait R/B.

6. Set *NFC_BUF_CNT* to define access data length is one page with ECC. Set *NFC_buff_saddr* to define the start address of internal buffer. Set *NFC_CEWP* to select flash device.
 7. Set *NFC_CTRL* bit [2:0] = '111' to enable NAND Flash Controller begins data transmission from internal buffer to flash device.
 8. Note: If not page access, must set *NFC_BUF_CNT[16]* = '0' to disable ECC protection.
-

Page-Program Operation (Large Page Device)

When write data to flash device, host must enable controller to execute Program operation.

Program operation flow as below:

Write data into internal buffer.

0. Set *NFC_CONFIG* register to define rear/write cycle timing and flash device basic setting.
 1. Write Program command to *NFC_CMD1*, *NFC_CMD2* and *NFC_SCMD* register, write start address to *NFC_ADDR1~5*, and set valid bit='1' to express command and address are valid.
 2. If normal Program operation, set *NFC_DMA_SET[9:0]* = '3a1' to enable Buffer Read/Write, Second command input, Auto Status Check, DMA operation is Write,Access Data Path is internal buffer and Device 0 Wait R/B.
 3. Set *NFC_BUF_CNT* to define access data length is one page with ECC. Set *NFC_BUF_SADDR* to define the start address of internal buffer. Set *NFC_CEWP* to select flash device.
 4. Set *NFC_CTRL[2:0]* = '111' to enable NAND Flash Controller begins data transmission from internal buffer to flash device.
 5. Note: If not page access, must set *NFC_BUF_CNT[16]* = '0' to disable ECC protection.
-

Page-Read

When read data from flash device, host must enable controller to execute Read operation. Read operation flow as below:

0. Set *NFC_CONFIG* register to define rear/write cycle timing and flash device basic setting.
1. Write Read command to *NFC_CMD1* and *NFC_CMD2*, write start address to *NFC_ADDR1~5*, and set valid bit='1' to express command and address are valid. Set *NFC_DMA_SET[4]* = '0' determine DMA operation mode is Read.

2. If normal Read operation, set *NFC_DMA_SET[9:0]* = '311' to enable Buffer Read/Write, Second command input, DMA operation is Read, Access Data Path is internal buffer and Device 0 Wait R/B.
3. Set *NFC_BUF_CNT* to define access data length is one page with ECC. Set *NFC_BUF_SADDR* to define the start address of internal buffer. Set *fNFC_CEWP* to select flash device.
4. Set *NFC_CTRL[2:0]* = '111' to enable NAND Flash Controller begins data transmission from device to internal buffer.
5. Note: If not page access, must set *NFC_BUF_CNT[16]* = '0' to disable ECC protection.

Read for Copy-Back

When read data from flash device, host must enable controller to execute Read for Copy-Back operation. The operation flow as below:

0. Set *NFC_CONFIG* register to define rear/write cycle timing and flash device basic setting
1. Write Copy-Back Read command to *NFC_CMD1* and *NFC_CMD2*, write start address to *NFC_ADDR1~5*, and set valid bit='1' to express command and address are valid.
2. If normal Read operation, set *NFC_DMA_SET[9:0]* = '311' to enable Buffer Read/Write, Second command input, DMA operation is Read, Access Data Path is internal buffer and Device 0 Wait R/B.
3. Set *NFC_CEWP* to select flash device.
4. Set *NFC_CTRL[2:0]* = '111' to enable NAND Flash Controller begin Read for Copy-Back operation.

Program for Copy-Back

When read data from flash device, host must enable controller to execute Program for Copy-Back operation. The operation flow as below:

0. Set *NFC_CONFIG* register to define rear/write cycle timing and flash device basic setting
1. Write Copy-Back Program command to *NFC_CMD1* and *NFC_CMD2* and *NFC_SCMD*, write start address to *NFC_ADDR1~5*, and set valid bit='1' to express command and address are valid.
2. If normal Read operation, set *NFC_DMA_SET[9:0]* = '1a1' to enable Second command input, Auto Status Check, DMA operation is Write, Access Data Path is internal buffer and Device 0 Wait R/B.
3. Set *NFC_BUF_CNT* to define access data length. Set *NFC_BUF_SADDR* to define the

start address of internal buffer. Set NFC_CEWP to select flash device.

4. *Set NFC_CTRL[2:0]='111' to enable NAND Flash Controller begin Program for Copy-Back operation*

■

Block Erase

Want invoke Block Erase operation there are three steps, host must:

0. *Write Block Erase command to NFC_CMD1 and NFC_CMD2, write start address to NFC_ADDR1~3 register, and set valid bit='1' to express command and address are valid.*
1. *Set NFC_DMA_SET[9:0] ='181' to enable Second command input, Auto Status Check Device 0 Wait R/B.*
2. *Set NFC_CTRL[2:0] ='101' to enable NAND Flash Controller begins block erase operation.*
3. *Wait selected device R/B='1', then NAND Flash Controller automatic read status from flash device status register.*

■

Read-ID

To invoke Red ID from flash memory device host must:

0. *Write Read ID command to NFC_CMD1, and valid bit='1' to express command is valid.*
1. *Set NFC_CTRL[0] ='1' (first command/address input enable) to enable NAND Flash Controller send Read ID command to flash device.*
2. *Set NFC_DMA_SET[9:0] ='050' to enable Access Data Path is register data port and DMA operation is Read.*
3. *Set NFC_BUF_CNT[16:0] ='3H' to define access data length. Set NFC_CE_WP to select flash device.*
4. *Set NFC_CTRL[2:0] ='111' to enable NAND Flash Controller begin READ-ID operation*

Chapter 6 Vectored Interrupt Controller

Overview

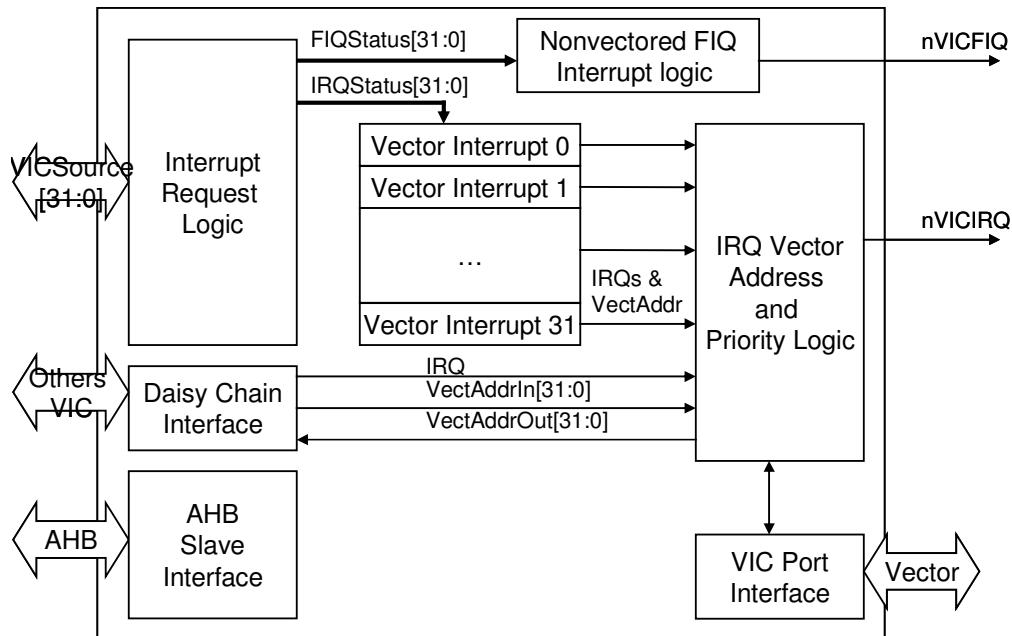
The vectored interrupt controller is composed of 2 VIC's (Vectored Interrupt Controller, ARM PrimeCell PL192). Two VIC's are daisy-chained to support up to 40 interrupt sources.

Key Features

- Support for 40 vectored IRQ interrupts per VIC
- Fixed hardware interrupt priority levels
- Programmable interrupt priority levels
- Hardware interrupt priority level masking
- Programmable interrupt priority level masking
- IRQ and FIQ generation
- Software interrupt generation
- Test registers
- Raw interrupt status
- Interrupt request status

Architecture

Block Diagram



Block Descriptions

Interrupt Request Logic

The interrupt request logic receives interrupt requests from peripherals and combines them with software interrupt requests. It then masks out the interrupt requests which are not enabled, and routes the enable interrupt to either IRQStatus or FIQStatus.

Non-vectored FIQ Interrupt Logic

The non-vectored FIQ interrupt logic generates FIQ interrupt signal by combining the FIQ interrupt requests in the interrupt controller and any requests from daisy-chained interrupt controllers.

Vectored IRQ Interrupt Logic

There are 32 vectored interrupt blocks. The vectored interrupt blocks receive the IRQ interrupt requests and set VectIRQx if the following are true:

- The interrupt is active
- The interrupt's programmed priority level is not masked

- The interrupt is currently the highest priority requesting interrupt

Each vectored interrupt block also provide a VectAddr[31:0] output for use in the interrupt priority block

Interrupt Priority Logic

The interrupt priority block prioritizes the interrupt requests using the programmed and hardware priority levels in the following order:

- Vectored interrupt requests
- External interrupt request (From Daisy chain)

The highest priority request generates an IRQ interrupt if the interrupt is not currently being serviced.

AHB Slave Interface

AHB slave interface provides CPU to programming register and checking status.

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
VIC_IRQSTATUS	0x0000	W	0x00000000	IRQ status register
VIC_FIQSTATUS	0x0004	W	0x00000000	FIQ status register
VIC_RAWINTR	0x0008	W	-	Raw interrupt status register
VIC_INTSELECT	0x000C	W	0x00000000	Interrupt select register
VIC_INTENABLE	0x0010	W	0x00000000	Interrupt enable register
VIC_INTENCLEAR	0x0014	W	-	Interrupt enable clear register
VIC_SOFTINT	0x0018	W	0x00000000	Software interrupt register
VIC_SOFTINTCLEAR	0x001C	W	-	Software interrupt clear register
VIC_PROTECTION	0x0020	W	0x00000000	Protection enable register
VIC_SWPRIORTYMASK	0x0024	W	0x0000FFFF	Software priority mask register

VIC_PRIORITYDAIS Y	0x0028	W	0x0000000F	Vector priority register for Daisy chain
VIC_VECTADDR0	0x0100	W	0x00000000	Vector address 0 register
VIC_VECTADDR1	0x0104	W	0x00000000	Vector address 1 register
VIC_VECTADDR2	0x0108	W	0x00000000	Vector address 2 register
VIC_VECTADDR3	0x010C	W	0x00000000	Vector address 3 register
VIC_VECTADDR4	0x0110	W	0x00000000	Vector address 4 register
VIC_VECTADDR5	0x0114	W	0x00000000	Vector address 5 register
VIC_VECTADDR6	0x0118	W	0x00000000	Vector address 6 register
VIC_VECTADDR7	0x011C	W	0x00000000	Vector address 7 register
VIC_VECTADDR8	0x0120	W	0x00000000	Vector address 8 register
VIC_VECTADDR9	0x0124	W	0x00000000	Vector address 9 register
VIC_VECTADDR10	0x0128	W	0x00000000	Vector address 10 register
VIC_VECTADDR11	0x012C	W	0x00000000	Vector address 11 register
VIC_VECTADDR12	0x0130	W	0x00000000	Vector address 12 register
VIC_VECTADDR13	0x0134	W	0x00000000	Vector address 13 register
VIC_VECTADDR14	0x0138	W	0x00000000	Vector address 14 register
VIC_VECTADDR15	0x013C	W	0x00000000	Vector address 15 register
VIC_VECTADDR16	0x0140	W	0x00000000	Vector address 16 register
VIC_VECTADDR17	0x0144	W	0x00000000	Vector address 17 register
VIC_VECTADDR18	0x0148	W	0x00000000	Vector address 18 register
VIC_VECTADDR19	0x014C	W	0x00000000	Vector address 19 register
VIC_VECTADDR20	0x0150	W	0x00000000	Vector address 20 register
VIC_VECTADDR21	0x0154	W	0x00000000	Vector address 21 register
VIC_VECTADDR22	0x0158	W	0x00000000	Vector address 22 register
VIC_VECTADDR23	0x015C	W	0x00000000	Vector address 23 register
VIC_VECTADDR24	0x0160	W	0x00000000	Vector address 24 register
VIC_VECTADDR25	0x0164	W	0x00000000	Vector address 25 register
VIC_VECTADDR26	0x0168	W	0x00000000	Vector address 26 register
VIC_VECTADDR27	0x016C	W	0x00000000	Vector address 27 register
VIC_VECTADDR28	0x0170	W	0x00000000	Vector address 28 register
VIC_VECTADDR29	0x0174	W	0x00000000	Vector address 29 register
VIC_VECTADDR30	0x0178	W	0x00000000	Vector address 30 register
VIC_VECTADDR31	0x017C	W	0x00000000	Vector address 31 register

VIC_VECTPRIORITY 0	0x0200	W	0x00000000	Vector priority 0 register
VIC_VECTPRIORITY 1	0x0204	W	0x00000000	Vector priority 1 register
VIC_VECTPRIORITY 2	0x0208	W	0x00000000	Vector priority 2 register
VIC_VECTPRIORITY 3	0x020C	W	0x00000000	Vector priority 3 register
VIC_VECTPRIORITY 4	0x0210	W	0x00000000	Vector priority 4 register
VIC_VECTPRIORITY 5	0x0214	W	0x00000000	Vector priority 5 register
VIC_VECTPRIORITY 6	0x0218	W	0x00000000	Vector priority 6 register
VIC_VECTPRIORITY 7	0x021C	W	0x00000000	Vector priority 7 register
VIC_VECTPRIORITY 8	0x0220	W	0x00000000	Vector priority 8 register
VIC_VECTPRIORITY 9	0x0224	W	0x00000000	Vector priority 9 register
VIC_VECTPRIORITY 10	0x0228	W	0x00000000	Vector priority 10 register
VIC_VECTPRIORITY 11	0x022C	W	0x00000000	Vector priority 11 register
VIC_VECTPRIORITY 12	0x0230	W	0x00000000	Vector priority 12 register
VIC_VECTPRIORITY 13	0x0234	W	0x00000000	Vector priority 13 register
VIC_VECTPRIORITY 14	0x0238	W	0x00000000	Vector priority 14 register
VIC_VECTPRIORITY 15	0x023C	W	0x00000000	Vector priority 15 register
VIC_VECTPRIORITY 16	0x0240	W	0x00000000	Vector priority 16 register

VIC_VECTPRIORITY 17	0x0244	W	0x00000000	Vector priority 17 register
VIC_VECTPRIORITY 18	0x0248	W	0x00000000	Vector priority 18 register
VIC_VECTPRIORITY 19	0x024C	W	0x00000000	Vector priority 19 register
VIC_VECTPRIORITY 20	0x0250	W	0x00000000	Vector priority 20 register
VIC_VECTPRIORITY 21	0x0254	W	0x00000000	Vector priority 21 register
VIC_VECTPRIORITY 22	0x0258	W	0x00000000	Vector priority 22 register
VIC_VECTPRIORITY 23	0x025C	W	0x00000000	Vector priority 23 register
VIC_VECTPRIORITY 24	0x0260	W	0x00000000	Vector priority 24 register
VIC_VECTPRIORITY 25	0x0264	W	0x00000000	Vector priority 25 register
VIC_VECTPRIORITY 26	0x0268	W	0x00000000	Vector priority 26 register
VIC_VECTPRIORITY 27	0x026C	W	0x00000000	Vector priority 27 register
VIC_VECTPRIORITY 28	0x0270	W	0x00000000	Vector priority 28 register
VIC_VECTPRIORITY 29	0x0274	W	0x00000000	Vector priority 29 register
VIC_VECTPRIORITY 30	0x0278	W	0x00000000	Vector priority 30 register
VIC_VECTPRIORITY 31	0x027C	W	0x00000000	Vector priority 31 register
VIC_ADDRESS	0x0F00	W	0x00000000	Vector address register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

VIC_IRQSTATUS

Address: Operational Base + offset (0x000)

IRQ status register

Bit	Attr	Reset Value	Description
31:0	R	0x0	<p>Show the status of the interrupts after masking by the VICINTENABLE and VICINTSELECT Registers:</p> <p>0 = interrupt is inactive.</p> <p>1 = interrupt is active.</p> <p>There is one bit of the register for each interrupt source.</p>

VIC_FIQSTATUS

Address: Operational Base + offset (0x004)

FIQ status register

Bit	Attr	Reset Value	Description
31:0	R	0x0	<p>Show the status of the FIQ interrupts after masking by the VICINTENABLE and VICINTSELECT Registers:</p> <p>0 = interrupt is inactive.</p> <p>1 = interrupt is active.</p> <p>There is one bit of the register for each interrupt source.</p>

VIC_RAWINTR

Address: Operational Base + offset (0x008)

FIQ status register

Bit	Attr	Reset Value	Description
31:0	R	0xX	<p>Show the status of the interrupts before masking by Enable Registers:</p> <p>0 = interrupt is inactive before masking.</p> <p>1 = interrupt is active before masking.</p> <p>Because this register provides a direct view of the raw interrupt inputs, the reset value is unknown. There is one bit of the register for each interrupt source</p>

VIC_INTSELECT

Address: Operational Base + offset (0x00C)

Interrupt select register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>Selects type of interrupt for interrupt request:</p> <p>0 = IRQ interrupt.</p> <p>1 = FIQ interrupt.</p> <p>There is one bit of the register for each interrupt source.</p>

VIC_INTENABLE

Address: Operational Base + offset (0x010)

Interrupt enable register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>Enables the interrupt request lines, which allow the interrupts to reach the processor.</p> <p>Read:</p> <p>0 = interrupt disabled</p> <p>1 = Interrupt enabled</p> <p>The interrupt enable can only be set using this register. The VIC_INTENCLEAR Register must be used to disable the interrupt enable.</p> <p>Write:</p> <p>0 = no effect</p> <p>1 = interrupt enabled.</p> <p>On reset, all interrupts are disabled.</p> <p>There is one bit of the register for each interrupt source.</p>

VIC_INTENCLEAR

Address: Operational Base + offset (0x014)

Interrupt enable clear register

Bit	Attr	Reset Value	Description
31:0	W	0x0	<p>Clears corresponding bits in the VICINTENABLE Register:</p> <p>0 = no effect</p> <p>1 = interrupt disabled in VIC_INTENABLE Register.</p>

			There is one bit of the register for each interrupt source.
--	--	--	---

VIC_SOFTINT

Address: Operational Base + offset (0x018)

Software interrupt register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	<p>Setting a bit HIGH generates a software interrupt for the selected source before interrupt masking.</p> <p>Read:</p> <ul style="list-style-type: none"> 0 = software interrupt inactive 1 = software interrupt active <p>Write:</p> <ul style="list-style-type: none"> 0 = no effect 1 = software interrupt enabled <p>There is one bit of the register for each interrupt source.</p>

VIC_SOFTINTCLEAR

Address: Operational Base + offset (0x01C)

Software interrupt clear register

Bit	Attr	Reset Value	Description
31:0	W	0x0	<p>Clears corresponding bits in the VIC_SOFTINT Register:</p> <ul style="list-style-type: none"> 0 = no effect 1 = software interrupt disabled in the VIC_SOFTINT Register. <p>There is one bit of the register for each interrupt source.</p>

VIC_PROTECTION

Address: Operational Base + offset (0x020)

Protection enable register

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	<p>Enables or disables protected register access:</p> <ul style="list-style-type: none"> 0 = protection mode disabled. 1 = protection mode enabled.

			When enabled, only privileged mode accesses (reads and writes) can access the interrupt controller registers, that is, when HPROT[1] is set HIGH for the current transfer. When disabled, both user mode and privileged mode can access the registers. This register can only be accessed in privileged mode, even when protection mode is disabled.
--	--	--	---

VIC_SWPRIORITYMASK

Address: Operational Base + offset (0x024)

Software priority mask register

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x1	Controls software masking of the 16 interrupt priority levels: 0 = interrupt priority level is masked 1 = interrupt priority level is not masked. Each bit of the register is applied to each of the 16 interrupt priority levels.

VIC_PRIORITYDAISY

Address: Operational Base + offset (0x028)

Vector priority register for Daisy chain

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x0	Selects vectored interrupt priority level. You can select any of the 16 vectored interrupt priority levels by programming the register with the hexadecimal value of the priority level required, from 0-15.

VIC_VECTADDRx (x=0~31)

Address: Operational Base + offset (0x100~0x017C)

Vector address x register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Contains ISR vector addresses.

VIC_VECTPRIORITx (x=0~31)

Address: Operational Base + offset (0x200~0x027C)

Vector priority x register

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x0	Selects vectored interrupt priority level. You can select any of the 16 vectored interrupt priority levels by programming the register with the hexadecimal value of the priority level required, from 0-15.

VIC_ADDRESS

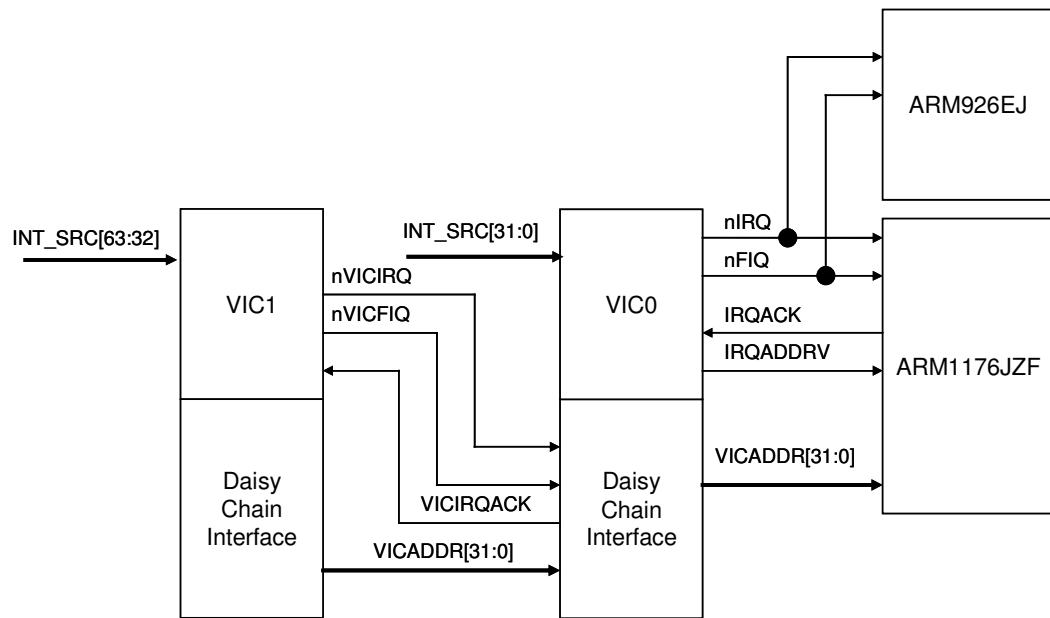
Address: Operational Base + offset (0xF00)

Vector address register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Contains the address of the currently active ISR. A read of this register returns the address of the ISR and sets the current interrupt as being serviced. A read must only be performed while there is an active interrupt. A write of any value to this register clears the interrupt. A write must only be performed at the end of an interrupt service routine.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only**Functional Description****Daisy-Chained Interrupt Scheme**

There are 40 interrupt sources provided in L6021. It is implemented by two cascaded Vector Interrupt Controller (VIC) through daisy-chain interface. This interrupt scheme provides ARM1176JZF and ARM926EJ CPU cores for IRQ and FIQ interrupt. Besides, it also provides ARM1176JZF vector interrupts to improve interrupt service latency handling.



Chapter 7 Dual-Core Communication Unit

Overview

The dual-core communication unit provides handshake mechanism in message delivery and software debug for dual-core system execution. It consists of one mailbox and one cross-trigger module. The mailbox provides inter-processor an effective communication channel with the interrupt mechanism. The cross-trigger module provides dual-core system debugging capability through one JTAG port of ICE.

Key Features

Mailbox

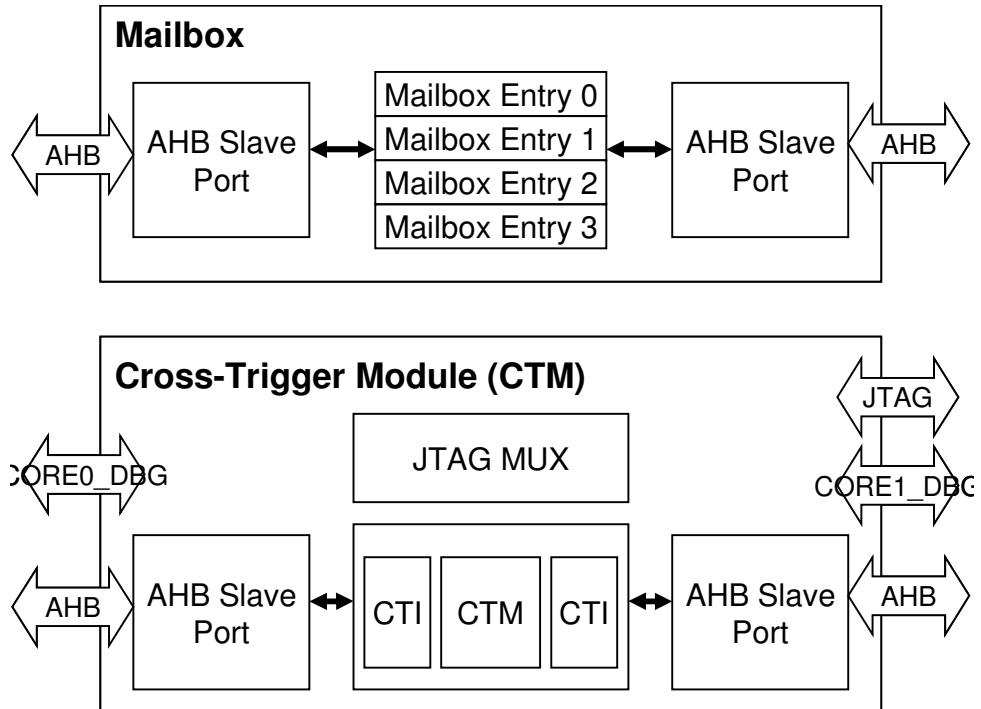
- Separate two AHB interfaces for dual-core access
- Each mailbox element includes one data word register, one command word register, and one flag bit that can represent one interrupt
- Output 1 interrupt to Host interrupt controller
- Output 1 interrupt to Client interrupt controller

Cross-Trigger Module (CTM)

- Separate two AHB interfaces for dual-core access
- Support debugging one JTAG port through daisy chain
- Support synchronization control stop from ICE command

Architecture

Block Diagram



Block Descriptions

Mailbox

The mailbox provides 2 AHB slave ports that connect to host CPU AHB bus and Client CPU AHB bus respectively. It can be programmed independently by processors from their private AHB bus. Each mailbox element contains one data word register, one command word register, and one flag bit that generates one interrupt source. The mailbox have 4 Host-to-Client mailbox elements that can present 4 interrupt sources, and generate 1 interrupt to Client interrupt controller and 4 Client-to-Host mailbox elements that can present 4 interrupt sources, and generate 1 interrupt to Client interrupt controller. It operates synchronously with the HCLK clock from the AHB bus.

Cross-Trigger Module (CTM)

The cross-trigger module provides JTAG daisy-chain and embedded cross trigger function. The JTAG daisy chain provides debugging tool connection through one JTAG port. The embedded

cross trigger implements ECT function of ARM that can support synchronous execution, like stop/step/go on Multi-ICE of ARM.

Registers

This section describes the control/status registers of the design.

Registers Summary

Mailbox

Name	Offset	Size	Reset Value	Description
MB_ID	0x0000	W	0x000000A9 0x000000A7	Mailbox AHB bus ID
MB_H2CSTAT	0x0010	W	0x00000000	Host CPU to client CPU Mailbox status
MB_H2CD0	0x0020	W	0x00000000	Data word for the H2C0 mailbox
MB_H2CC0	0x0024	W	0x00000000	Command word for the H2C0 mailbox
MB_H2CD1	0x0028	W	0x00000000	Data word for the H2C1 mailbox
MB_H2CC1	0x002C	W	0x00000000	Command word for the H2C1 mailbox
MB_H2CD2	0x0030	W	0x00000000	Data word for the H2C2 mailbox
MB_H2CC2	0x0034	W	0x00000000	Command word for the H2C2 mailbox
MB_H2CD3	0x0038	W	0x00000000	Data word for the H2C3 mailbox
MB_H2CC3	0x003C	W	0x00000000	Command word for the H2C3 mailbox
MB_C2HSTAT	0x0040	W	0x00000000	Client CPU to host CPU mailbox status
MB_C2HD0	0x0050	W	0x00000000	Data word for the C2H0 mailbox
MB_C2HC0	0x0054	W	0x00000000	Command word for the C2H0 mailbox
MB_C2HD1	0x0058	W	0x00000000	Data word for the C2H1 mailbox
MB_C2HC1	0x005C	W	0x00000000	Command word for the C2H1 mailbox

MB_C2HD2	0x0060	W	0x00000000	Data word for the C2H2 mailbox
MB_C2HC2	0x0064	W	0x00000000	Command word for the C2H2 mailbox
MB_C2HD3	0x0068	W	0x00000000	Data word for the C2H3 mailbox
MB_C2HC3	0x006C	W	0x00000000	Command word for the C2H3 mailbox

Cross-Trigger Module (CTM)

Name	Offset	Size	Reset Value	Description
CTM_CTRL	0x0000	W	0x00000000	CTM control register
CTM_STAT	0x0004	W	0x00000001 or 0x00000003	CTM status register
CTM_LOCK	0x0008	W	0x00000000	CTM lock enable register
CTM_PROTECT	0x000C	W	0x00000000	CTM protection enable register
CTM_INTACK	0x0010	W	-	CTM interrupt acknowledge register
CTM_APPSET	0x0014	W	0x00000000	CTM application trigger set register
CTM_APPCLEAR	0x0018	W	0x00000000	CTM application trigger clear register
CTM_APPPULSE	0x001C	W	0x00000000	CTM application pulse register
CTM_INEN0	0x0020	W	0x00000000	CTM trigger to channel enable register 0
CTM_INEN1	0x0024	W	0x00000000	CTM trigger to channel enable register 1
CTM_INEN2	0x0028	W	0x00000000	CTM trigger to channel enable register 2
CTM_INEN3	0x002C	W	0x00000000	CTM trigger to channel enable register 3
CTM_INEN4	0x0030	W	0x00000000	CTM trigger to channel enable register 4
CTM_INEN5	0x0034	W	0x00000000	CTM trigger to channel enable register 5

CTM_INEN6	0x0038	W	0x00000000	CTM trigger to channel enable register 6
CTM_INEN7	0x003C	W	0x00000000	CTM trigger to channel enable register 7
CTM_OUTEN0	0x00A0	W	0x00000000	CTM channel to trigger Enable register 0
CTM_OUTEN1	0x00A4	W	0x00000000	CTM channel to trigger Enable register 1
CTM_OUTEN2	0x00A8	W	0x00000000	CTM channel to trigger Enable register 2
CTM_OUTEN3	0x00AC	W	0x00000000	CTM channel to trigger Enable register 3
CTM_OUTEN4	0x00B0	W	0x00000000	CTM channel to trigger Enable register 4
CTM_OUTEN5	0x00B4	W	0x00000000	CTM channel to trigger Enable register 5
CTM_OUTEN6	0x00B8	W	0x00000000	CTM channel to trigger Enable register 6
CTM_OUTEN7	0x00BC	W	0x00000000	CTM channel to trigger Enable register 7
CTM_TRIISTAT	0x0130	W	-	CTM trigger-in status register
CTM_TRIOSTAT	0x0134	W	0x00000000	CTM trigger-out status register
CTM_CHISTAT	0x0138	W	-	CTM channel-n status register
CTM_CHOSTAT	0x013C	W	0x00000000	CTM channel-out status register
CTM_PERIPHID0	0x0FE0	W	0x00000000	CTM peripheral ID0
CTM_PERIPHID1	0x0FE4	W	0x00000019	CTM peripheral ID1
CTM_PERIPHID2	0x0FE8	W	0x00000004	CTM peripheral ID2
CTM_PERIPHID3	0x0FEC	W	0x00000000	CTM peripheral ID3

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

MB_ID

Address: Operational Base + offset (0x0000)

Mailbox AHB bus ID

Bit	Attr	Reset Value	Description
31:0	R	0x000000A9 or 0x000000A7	If read from embedded ARM9/11 CPU from 0x500A_0000 based address, mailbox_id=0x000000A9. If read from external 2 nd CPU from 0x500C_0000 based address, mailbox_id=0x000000A7.

MB_H2CSTAT

Address: Operational Base + offset (0x0010)

Host CPU to client CPU Mailbox status

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved
3	R	0x00000000	H2C3_FLAG Indicate that the H2C3 mailbox interrupt has been generated. Set by host CPU writing to MB_H2CC3; cleared by client CPU reading of MB_H2CC3. Before host CPU write H2C3 mailbox data word and command word, host CPU need to check if H2C3_FLAG is clear. Before client CPU read out H2C3 mailbox, client CPU need to check if H2C3_FLAG is set.
2	R	0x00000000	H2C2_FLAG Indicate that the H2C2 mailbox interrupt has been generated. Set by host CPU writing to MB_H2CC2; cleared by client CPU reading of MB_H2CC2. Before host CPU write H2C2 mailbox data word and command word, host CPU need to check if H2C2_FLAG is clear. Before client CPU read out H2C2 mailbox, client CPU need to check if H2C2_FLAG is set.
1	R	0x00000000	H2C1_FLAG

			Indicate that the H2C1 mailbox interrupt has been generated. Set by host CPU writing to MB_H2CC1; cleared by client CPU reading of MB_H2CC1. Before host CPU write H2C1 mailbox data word and command word, host CPU need to check if H2C1_FLAG is clear. Before client CPU read out H2C1 mailbox, client CPU need to check if H2C1_FLAG is set.
0	R	0x00000000	H2C0_FLAG Indicate that the H2C0 mailbox interrupt has been generated. Set by host CPU writing to MB_H2CC0; cleared by client CPU reading of MB_H2C0. Before host CPU write H2C0 mailbox data word and command word, host CPU need to check if H2C0_FLAG is clear. Before client CPU read out H2C0 mailbox, client CPU need to check if H2C0_FLAG is set.

MB_H2CDx (x=0~3)

Address: Operational Base + offset (0x0020, 0x0028, 0x0030, 0x0038)

Data word for the H2C0/H2C1/H2C2/H2C3 mailbox

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	Data word for the H2Cx mailbox. It can be written only by host CPU and read by client CPU. Writing to this register does not generate an interrupt.

MB_H2CCx (x=0~3)

Address: Operational Base + offset (0x0024, 0x002C, 0x0034, 0x003C)

Command word for the H2C0/H2C1/H2C2/H2C3 mailbox

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	Command word for the H2Cx mailboxX.

			<p>It can be written only by host CPU and read by client CPU. Writing to this register generates the H2Cx interrupt and sets H2Cx_FLAG.</p> <p>When client CPU reads this register, the interrupt and flag are cleared.</p>
--	--	--	---

MB_C2HSTAT

Address: Operational Base + offset (0x0040)

Client CPU to host CPU mailbox status

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3	R	0x00000000	<p>C2H3_FLAG</p> <p>Indicate that the C2H3 mailbox interrupt has been generated. Set by client CPU writing to MB_C2HC3; cleared by host CPU reading of MB_C2HC3. Before client CPU write C2H3 mailbox data word and command word, client CPU needs to check if C2H3_FLAG is clear. Before host CPU read out C2H3 mailbox, host CPU need to check if C2H3_FLAG is set.</p>
2	R	0x00000000	<p>C2H2_FLAG</p> <p>Indicate that the C2H2 mailbox interrupt has been generated. Set by client CPU writing to MB_C2HC2; cleared by host CPU reading of MB_C2HC2. Before client CPU write C2H2 mailbox data word and command word, client CPU needs to check if C2H2_FLAG is clear. Before</p>
1	R	0x00000000	<p>C2H1_FLAG</p> <p>Indicate that the C2H1 mailbox interrupt has been generated. Set by client CPU writing to MB_C2HC1; cleared by host CPU reading of</p>

			MB_C2HC1. Before client CPU write C2H1 mailbox data word and command word, client CPU needs to check if C2H1_FLAG is clear. Before host CPU read out C2H1 mailbox, host CPU need to check if C2H1_FLAG is set.
0	R	0x00000000	C2H0_FLAG Indicate that the C2H0 mailbox interrupt has been generated. Set by client CPU writing to MB_C2HC0; cleared by host CPU reading of MB_C2HC0. Before client CPU write C2H0 mailbox data word and command word, client CPU need to check if C2H0_FLAG is clear. Before host CPU read out C2H0 mailbox, host CPU need to check if C2H0_FLAG is set.

MB_C2HDx (x=0~3)

Address: Operational Base + offset (0x0050, 0x0058, 0x0060, 0x0068)

Data word for the C2H0/C2H1/C2H2/C2H3 mailbox

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	Data word for the C2Hx mailbox. It can be written only by client CPU and read by host CPU. Writing to this register does not generate an interrupt.

MB_C2HCx (x=0~3)

Address: Operational Base + offset (0x0054, 0x005C, 0x0064, 0x006C)

Command word for the C2H0/C2H1/C2H2/C2H3 mailbox

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	Command word for the C2Hx mailbox. It can be written only by client CPU and read by host CPU. Writing to this register generates the C2Hx mailbox interrupt and sets C2Hx_FLAG. When host CPU reads this register, the interrupt and flag are cleared.

CTM_CTRL

Address: Operational Base + offset (0x0000)

CTM control register

Bit	Attr	Reset Value	Description
31:1	R	0x00000000	Reserved.
0	RW	0x0	<p>CTMGLBEN</p> <p>Enables or disables the ECT</p> <p>0: Disabled 1: Enabled</p> <p>When disabled, all cross triggering mapping logic functionality is disabled for this processor.</p>

CTM_STAT

Address: Operational Base + offset (0x0004)

CTM status register

Bit	Attr	Reset Value	Description
31:2	R	0x00000000	Reserved.
1	R	-	<p>DBGEN</p> <p>Indicate the status of the ECTDBGEN port. The ECTDBGEN is used to enable the Trigger and CIs to the ECT :</p> <p>0: Interfaces disabled 1: Interfaces enabled.</p> <p>Because the AHB interface is not controlled by the ECTDBGEN port, this register can be read while ECTDBGEN is LOW. The reset value depends on the ECTDBGEN port.</p>
0	R	0x1	<p>LOCKED</p> <p>Indicate the locked status of the CTM</p> <p>0: Access to the CTM is not locked 1: Access to the CTM is locked</p>

CTM_LOCK

Address: Operational Base + offset (0x0008)

CTM lock enable register

CTM_LOCK is a write-only register. This register enables or disables all other register write accesses by providing a LOCKED mode for the AHB interface. When the AHB interface is locked, write accesses to the CTM registers except CTM_LOCK are ignored. To unlock the AHB interface, a LOCKKEY value,

0x0ACCE550, must be written to the CTM_LOCK register. When unlocked, the AHB interface accepts write accesses. To lock the AHB interface, data that is not equal to LOCKKEY value, 0x0ACCE550, is written into CTM_LOCK. The LOCK mode protects the CTM from spurious writes. The LOCKED bit in the CTM_STAT register indicates the status of the lock. The write access to CTM_LOCK is affected by the protection status. Therefore if protection mode, CTM_PROTECT, is enabled, accesses to the CTM_LOCK must be handled in privilege mode.

Bit	Attr	Reset Value	Description
31:0	W	0x00000000	<p>LOCKKEY</p> <p>To enable the other registers in the CTM to be accessible, the correct access code of 0x0ACCE550 must be written to this register. To disable access to the other CTM registers, any value other than 0x0ACCE550 must be written to this register. In reset the lock is set.</p>

CTM_PROTECT

Address: Operational Base + offset (0x000C)

CTM protection enable register

CTM_PROTECT enables or disables protected register access, stopping register accesses when the processor is in user mode. If the CTM_PROTECT is enabled then all register accesses must be done in privilege mode.

Bit	Attr	Reset Value	Description
31:1	R	0x00000000	Reserved.
0	R	-	<p>Enables or disables protected register access:</p> <p>0: Protection mode disabled (reset)</p> <p>1: Protection mode enabled.</p> <p>When enabled, only privilege mode accesses (reads and</p>

		<p>writes) can access the CTM registers, that means HPROT[1] is set HIGH for the current transfer. If an access is made in user mode all registers return 0. HRESP[1:0] does not return an error because operating system might not be able to handle this type of error response.</p> <p>When disabled, both user mode and privileged mode can access the registers in the CTM, except for the CTM_PROTECT that can only be accessed in privilege mode, even when the protection mode is disabled.</p>
--	--	---

CTM_INTACK

Address: Operational Base + offset (0x0010)

CTM interrupt acknowledge register

Any bits written as a HIGH causes the CTMTRIGOUT output signal to be acknowledged. The acknowledgement is cleared when MAPTRIGOUT is deactivated. This register is used when the CTMTRIGOUT is used as a sticky class output.

Bit	Attr	Reset Value	Description
31:8	R	0x000000	Reserved.
7:0	W	-	<p>INTACK</p> <p>Acknowledges the corresponding CTMTRIGOUT output :</p> <p>0: nothing happens</p> <p>1: CTMTRIGOUT is acknowledged and will be cleared when MAPTRIGOUT is low. One bit of the register represents one CTMTRIGOUT output.</p>

CTM_APPSET

Address: Operational Base + offset (0x0014)

CTM application trigger set register

A write to this register causes a channel event to be raised, corresponding to the bit written to.

Bit	Attr	Reset Value	Description
31:4	R	0x00000000	Reserved.
3:0	RW	0x0	<p>APPSET</p> <p>Setting a bit HIGH generates a channel event for the selected channel.</p> <p>Read :</p> <ul style="list-style-type: none"> 0: Application trigger inactive. 1: Application trigger active. <p>Write :</p> <ul style="list-style-type: none"> 0: No effect. 1: Generate channel event. <p>One bit of the register represents one channel.</p>

CTM_APPCLEAR

Address: Operational Base + offset (0x0018)

CTM application trigger clear register

A write to this register causes a channel event to be cleared, corresponding to the bit written to.

Bit	Attr	Reset Value	Description
31:4	R	0x00000000	Reserved.
3:0	RW	0x0	<p>APPCLEAR</p> <p>Clears corresponding bits in the CTM_APPSET register :</p> <ul style="list-style-type: none"> 0: No effect 1: Application trigger disabled in the CTM_APPSET register. <p>One bit of the register represents one channel.</p>

CTM_APPPULSE

Address: Operational Base + offset (0x001C)

CTM application pulse register

A write to this register causes a channel event pulse (one ECTCTICLK period) to be generated, corresponding to the bit written to. The pulse external to the ECT can be extended to multi-cycle by the handshaking interface circuits. This register clears itself immediately, so it can be repeatedly written without software programming to clear it.

Bit	Attr	Reset Value	Description
31:4	R	0x0000000	Reserved.
3:0	W	-	<p>APPPULSE Setting a bit HIGH generates a channel event pulse for the selected channel.</p> <p>0: No effect 1: Channel event pulse generated for one ECTCTICLK period.</p> <p>One bit of the register represents one channel.</p>

CTM_INENx (x=0~7)

Address: Operational Base + offset (0x0020/0024/0028/002C/0030/0034/0038/003C)

CTM trigger to channel enable register x

CTMINENx enable the signaling of an event on a CTM channel/s when the core issues a trigger (ECTTRIGIN) to the CTM. There is one register for each of the eight ECTTRIGIN input. Within each register there is one bit for each of the four channels implemented. These register do not affect the application trigger operations.

Bit	Attr	Reset Value	Description
31:4	R	0x0000000	Reserved.
3:0	RW	0x0	<p>TRIGINEN Enables a cross trigger event to the corresponding channel when an ECTTRIGIN is activated :</p> <p>0: Disables the ECTTRIGIN signal from generating an event on the respective channel of the CTM 1: Enables the ECTTRIGIN signal to generate an event on the respective channel of the CTM.</p>

			There is one bit of the register for each of the 4 channels. For example in register CTM_INENx, TRIGINEN[x] set to HIGH enables ECTTRIGIN onto channel x.
--	--	--	---

CTM_OUTENx (x=0~7)

Address: Operational Base + offset (0x00A0/00A4/00A8/00AC/00B0/00B4/00B8/00BC)

CTM channel to trigger Enable register x

CTM_OUTENx defines which channel(s) can generate an ECTTRIGOUT output. There is one register for each of the eight ECTTRIGOUT output. Within each register there is one bit for each of the four channels implemented. These registers affect the mapping from application trigger to trigger output.

Bit	Attr	Reset Value	Description
31:4	R	0x00000000	Reserved.
3:0	RW	0x0	<p>TRIGOUTEN</p> <p>Changing the value of this bit from LOW to HIGH will enable a channel event for the corresponding channel to generate a ECTTRIGOUT output :</p> <p>0: Channel input (CTICHIN) from the CTM is not routed to the ECTTRIGOUT output</p> <p>1: Channel input (CTICHIN) from the CTM is routed to the ECTTRIGOUT output.</p> <p>There is one bit for each of the four channels. For example in register CTM_OUTENx, enabling bit x enables CTICHIN[x] to cause a trigger</p>

CTM_TRIISTAT

Address: Operational Base + offset (0x0130)

CTM trigger-in status register

Bit	Attr	Reset Value	Description
31:8	R	0x00000000	Reserved.
7:0	R	0x0	Indicate the status of the ECTTRIGIN inputs :

		<p>0: ECTTRIGIN is inactive</p> <p>1: ECTTRIGIN is active.</p> <p>Because the register provides a view of the raw ECTTRIGIN inputs, the reset value is unknown. There is one bit of the register for each trigger input.</p>
--	--	--

CTM_TRIOSTAT

Address: Operational Base + offset (0x0134)

CTM trigger-out status register

Bit	Attr	Reset Value	Description
31:8	R	0x000000	Reserved.
7:0	R	0x0	<p>Indicate the status of the ECTTRIGOUT outputs :</p> <p>0: ECTTRIGOUT is inactive (reset)</p> <p>1: ECTTRIGOUT is active.</p> <p>There is one bit of the register for each trigger output.</p>

CTM_CHISTAT

Address: Operational Base + offset (0x0138)

CTM channel-in status register

Bit	Attr	Reset Value	Description
31:4	R	0x0000000	Reserved.
3:0	R	0x0	<p>Indicate the status of the CTICHIN inputs :</p> <p>0: CTICHIN is inactive</p> <p>1: CTICHIN is active.</p> <p>Because the register provides a view of the raw CTICHIN inputs from the CTM, the reset value is unknown. There is one bit of the register for each channel input.</p>

CTM_CHOSTAT

Address: Operational Base + offset (0x013C)

CTM channel-out status register

Bit	Attr	Reset Value	Description
31:4	R	0x00000000	Reserved.
3:0	R	0x0	Indicate the status of the CTICHOUT outputs : 0: CTICHOUT is inactive (reset) 1: CTICHOUT is active. There is one bit of the register for each channel output.

CTM_PERIPHIDx (x=0~3)

Address: Operational Base + offset (0x0FE0)

CTM peripheral IDx register

CTM_PERIPHID0-3 can conceptually be treated as a single 32-bit register.

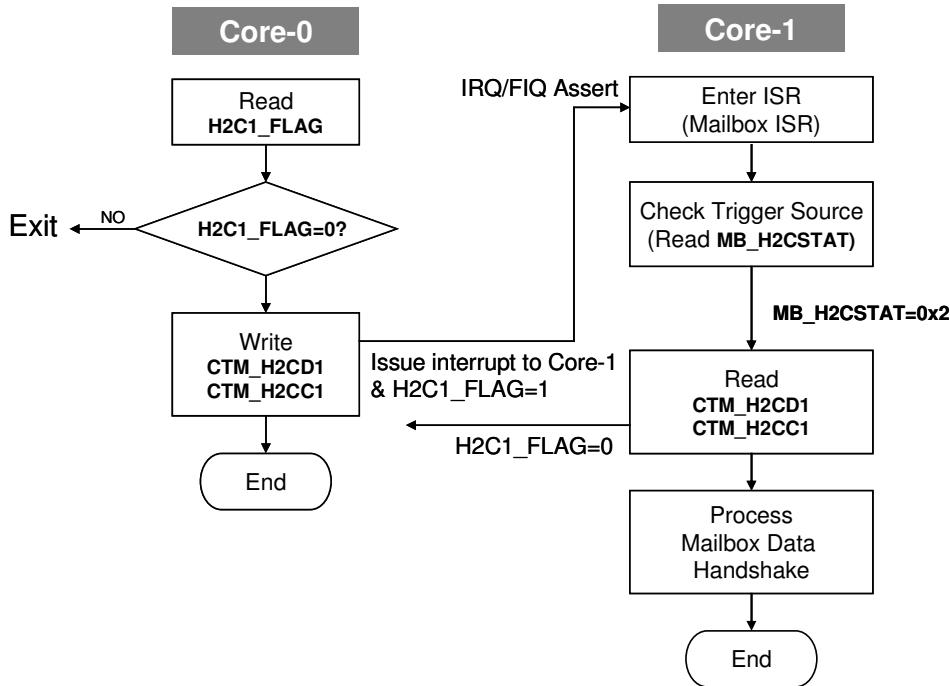
Bit	Attr	Reset Value	Description
31:8	R	0x00000000	Reserved.
7:0	R	0x0	Indicate the status of the CTICHOUT outputs : 0: CTICHOUT is inactive (reset) 1: CTICHOUT is active. There is one bit of the register for each channel output.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Mailbox Programming Sequence

Dual-core communication example flow using Mailbox H2C1



ECT Programming Sequence

When debugging with ECT enable using Multi-ICE of ARM. Follow the step below.

1. Unlock ECT, write ECT offset 0x0008 with data 0x0acce550
2. Check ECT status, read ECT offset 0x00004, check data if 0x00000002
3. Enable ECT, write ECT offset 0x0000 with data 0x00000001
4. Enable CTM INPUT, write ECT offset 0x0020 with data 0x0000000F
5. Enable CTM OUTPUT, write ECT offset 0x00A0 with data 0x0000000F
6. Set breakpoint on debugging tool, and etc.

Chapter 8 DMA Controller

Overview

AHB DMA (HDMA) is interfaced to AHB bus and provides DMA function for AHB peripherals. The AHB DMA provides two channels for software usage and provides two DMA trigger mode, software and hardware mode. Register writing triggers the software DMA mode, and the hardware DMA mode is triggered from peripherals.

Key Features

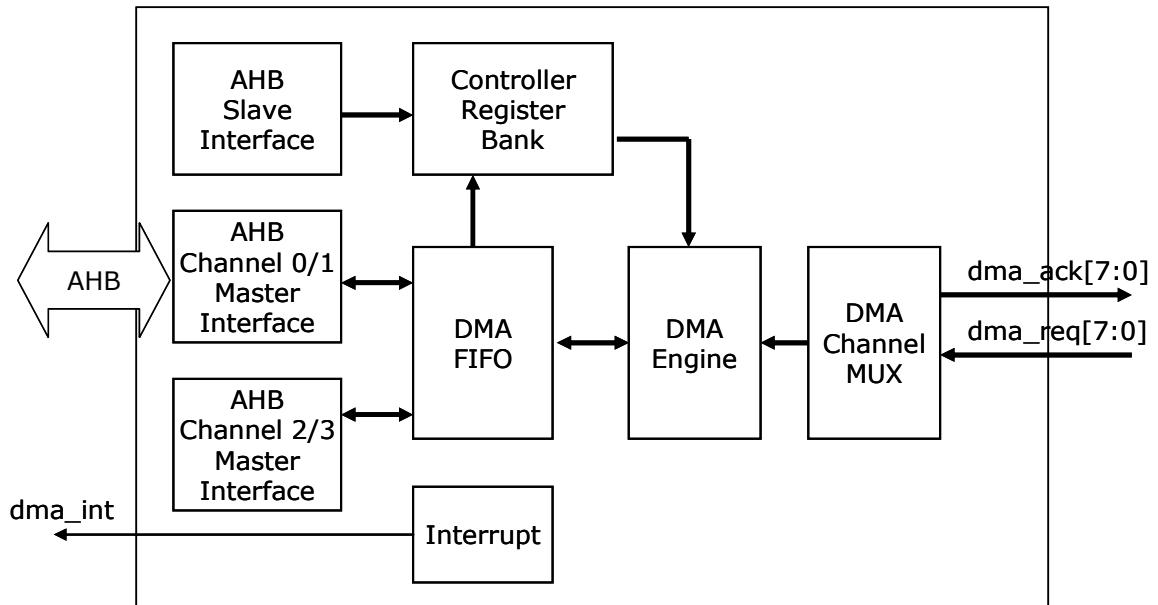
- Built-in 32x16 data FIFO
- Two DMA channels supported
- Support byte, half word and word data transfer sizes
- Support incremental and fixed addressing mode
- Support hardware and software trigger DMA transfer mode
- Fixed channel priority arbitration
- Support slice transfer base on peripheral FIFO depth
- Support turnaround start address when over address buffer size

For hardware DMA mode:

- Only support word data transfer size

Architecture

Block Diagram



Block Descriptions

AHB DMA consists of one AHB slave interface and one AHB master interfaces.

AHB slave interface is used for registers setting and AHB master interface issues command from DMA engine to AHB bus, eight request MUXs select the corresponding hardware request port and the priority MUX selects which channel is served by DMA engine.

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
HDMA_CON0	0x0000	W	0x00000000	HDMA channel 0 control register
HDMA_CON1	0x0004	W	0x00000000	HDMA channel 1 control register
HDMA_ISRC0	0x0008	W	0x00000000	HDMA channel 0 initial source address register.

HDMA_IDST0	0x000C	W	0x00000000	HDMA channel 0 initial destination address register.
HDMA_ICNT0	0x0010	W	0x00000000	HDMA channel 0 initial terminate count register.
HDMA_ISRC1	0x0014	W	0x00000000	HDMA channel 1 initial source address register.
HDMA_IDST1	0x0018	W	0x00000000	HDMA channel 1 initial destination address register.
HDMA_ICNT1	0x001C	W	0x00000000	HDMA channel 1 initial terminate count register.
HDMA_CSRC0	0x0020	W	0x00000000	HDMA channel 0 current source address register.
HDMA_CDST0	0x0024	W	0x00000000	HDMA channel 0 current destination address register.
HDMA_CCNT0	0x0028	W	0x00000000	HDMA channel 0 current count register.
HDMA_CSRC1	0x002C	W	0x00000000	HDMA channel 1 current source address register.
HDMA_CDST1	0X0030	W	0x00000000	HDMA channel 1 current destination address register.
HDMA_CCNT1	0x0034	W	0x00000000	HDMA channel 1 current count register.
HDMA_ISR01	0x0038	W	0x00000000	HDMA interrupt ch0/1 status register
HDMA_DSR01	0x003C	W	0x00000000	HDMA DMA ch0/1 status register
HDMA_ISCNT0	0x0040	W	0x00000000	HDMA channel 0 initial slice count register
HDMA_IPNCNTD0	0x0044	W	0x00000000	HDMA channel 0 initial page number count down register
HDMA_IADDR_BS0	0x0048	W	0x1FFFFFFF	HDMA channel 0 initial address buffer size register
HDMA_ISCNT1	0x004C	W	0x00000000	HDMA channel 1 initial slice count register

HDMA_IPNCNTD1	0x0050	W	0x00000000	HDMA channel 1 initial page number count down register
HDMA_IADDR_BS 1	0x0054	W	0x1FFFFFFF	HDMA channel 1 initial address buffer size register
HDMA_CSCNT0	0x0058	W	0x00000000	HDMA channel 0 current slice count register
HDMA_CPNCNTD0	0x005C	W	0x00000000	HDMA channel 0 current page number count down register
HDMA_CADDR_BS 0	0x0060	W	0x00000000	HDMA channel 1 current address buffer size register
HDMA_CSCNT1	0x0064	W	0x00000000	HDMA channel 1 current slice count register
HDMA_CPNCNTD1	0x0068	W	0x00000000	HDMA channel 1 current page number count down register
HDMA_CADDR_BS 1	0x006C	W	0x00000000	HDMA channel 1 current address buffer size register
HDMA_PACNT0	0x0070	W	0x00000000	HDMA channel 0 page accumulation count register
HDMA_PACNT1	0x0074	W	0x00000000	HDMA channel 1 page accumulation count register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

HDMA_CONx(x=0, 1)

Address: Operational Base + offset (0x00, 0x04)

HDMA control register for channel x

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23	RW	0x0	Clear CPNCNTDx register 0 : not clear 1 : Clear
22	RW	0x0	Hardware HDMA slice mode transfer data enable/disable 0: Disable

			1: Enable
21	RW	0x0	HDMA channel enable/disable 0: Disable DMA 1: Enable DMA
20	RW	0x0	Reserved.
19:18	RW	0x0	Interrupt Mode Set. 00: Polling mode 01: Interrupt mode
17:16	-	-	Reserved.
15:13	RW	0x0	Transfer Mode. 0x0: Single 0x1: Reserved 0x2: Reserved 0x3: INCR4 0x4: Reserved 0x5: INCR8 0x6: Reserved 0x7: INCR16
12:9	RW	0x0	External HDREQ source selection. HDMA0/1: 0x5: From I2S Others: Reserved
8:7	RW	0x0	Direction of source address. 0x0: Increment 0x1: Fixed 0x2 ~ 0x3: Reserved
6:5	RW	0x0	Direction of destination address. 0x0: Increment 0x1: Fixed 0x2 ~ 0x3: Reserved
4:3	RW	0x0	Data size for transfer. 0x0: Byte 0x1: Halfword 0x2: Word 0x3: Reserved

2:1	RW	0x0	Command of Software DMA operation. 0x0: No command 0x1: Start software DMA operation 0x2: Pause software DMA operation 0x3: Cancel software DMA operation
0	RW	0x0	Disable/Enable hardware trigger DMA mode. 0: Disable 1: Enable

HDMA_ISRC_x(x=0, 1)

Address: Operational Base + offset (0x08, 0x14)

HDMA initial source address register for channel x

Bit	Attr	Reset Value	Description
31:0	RW	0x0	HDMA initial source address register.

HDMA_CSRC_x(x=0, 1)

Address: Operational Base + offset (0x20, 0x2C)

HDMA current source address register for channel x

Bit	Attr	Reset Value	Description
31:0	R	0x0	HDMA current source address register.

HDMA_IDST_x(x=0, 1)

Address: Operational Base + offset (0x0C, 0x18)

HDMA initial destination address register for channel x

Bit	Attr	Reset Value	Description
31:0	RW	0x0	HDMA initial destination address register.

HDMA_CDST_x(x=0, 1)

Address: Operational Base + offset (0x24, 0x30)

HDMA current destination address register for channel x

Bit	Attr	Reset Value	Description
31:0	R	0x0	HDMA current destination address register.

HDMA_ICNT_x(x=0, 1)

Address: Operational Base + offset (0x10, 0x1C)

HDMA initial terminate count register for channel x

Bit	Attr	Reset Value	Description
31:0	-	-	Reserved.
15:0	RW	0x0	HDMA initial terminate count register.

HDMA_CCNTx(x=0, 1)

Address: Operational Base + offset (0x28, 0x34)

HDMA current terminate count register for channel x

Bit	Attr	Reset Value	Description
31:0	-	-	Reserved.
15:0	R	0x0	HDMA current terminate count register.

HDMA_ISRx(x=0, 1)

Address: Operational Base + offset (0x38)

Interrupt status

Bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13	RW	0x0	Mask channel 1 Page accumulation overflow Interrupt 0: not mask 1: mask
12	RW	0x0	Mask channel 0 Page accumulation overflow Interrupt 0: not mask 1: mask
11	RW	0x0	Mask channel 1 Page count down Interrupt 0: not mask 1: mask
10	RW	0x0	Mask channel 0 Page count down Interrupt 0: not mask 1: mask
9	RW	0x0	Mask channel 1 Interrupt 0: not mask 1: mask
8	RW	0x0	Mask channel 0 Interrupt

			0: not mask 1: mask
7	-	-	Reserved.
6	RW	0x0	Channel 1 Page accumulation counter overflow Interrupt active, write “0” to clear interrupt and write “1” is no affect 0: not active 1: active
5	RW	0x0	Channel 0 Page accumulation counter overflow Interrupt active, write “0” to clear interrupt and write “1” is no affect 0: not active 1: active
4	RW	0x0	Channel 1 Page count down to zero Interrupt active, write “0” to clear interrupt and write “1” is no affect 0: not active 1: active
3	RW	0x0	Channel 0 Page count down to zero Interrupt active, write “0” to clear interrupt and write “1” is no affect 0: not active 1: active
2	RW	0x0	Mask channel 1 Page accumulation overflow Interrupt 0: not mask 1: mask
1	RW	0x0	Channel 1 Interrupt active, clear after read write “0” to clear interrupt and write “1” is no affect 0: not active 1: active
0	RW	0x0	Channel 0 Interrupt active, write “0” to clear interrupt and write “1” is no affect 0: not active 1: active

HDMA_DSRx(x=0, 1)

Address: Operational Base + offset (0x3C)

DMA status

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	R	0x0	Channel 1 status 0: Channel 1 is ready 1: Channel 1 is performing DMA
0	R	0x0	Channel 0 status 0: Channel 0 is ready 1: Channel 0 is performing DMA

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

HDMA_ISCNTx(x=0, 1)

Address: Operational Base + offset(0x40, 0x4C)

HDMA initial slice count register for channel x

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	HDMA initial slice count register. (Unit : data size)

HDMA_CSCNTx(x=0, 1)

Address: Operational Base + offset(0x58, 0x64)

HDMA current slice count register for channel x

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	HDMA current slice count register. (Unit : data size)

HDMA_IPNCNTDx(x=0, 1)

Address: Operational Base + offset(0x44, 0x50)

HDMA initial total page number count down register for channel x

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	W	0x0	HDMA initial total page number count down register. The value will be accumulated if you write it again and again.
15:0	R	0x0	HDMA initial total accumulation page number count down register.

HDMA_CPNCNTDx(x=0, 1)

Address: Operational Base + offset(0x5C, 0x68)

HDMA current total page number count down register for channel x

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	R	0x0	HDMA current total page number count down register.

HDMA_IADDR_BSx(x=0, 1)

Address: Operational Base + offset(0x48, 0x54)

HDMA initial address buffer size register for channel x

bit	Attr	Reset Value	Description
31:0	RW	0x1FFFFFFF	HDMA initial source address buffer size register. (Unit : byte)

HDMA_CADDR_BSx(x=0, 1)

Address: Operational Base + offset(0x60, 0x6C)

HDMA current address buffer size register for channel x

bit	Attr	Reset Value	Description
31:0	R	0x1FFFFFFF	HDMA current source address buffer size register. (Unit : byte)

HDMA_PACNTx(x=0, 1)

Address: Operational Base + offset(0x70, 0x74)

HDMA page accumulation count register for channel x

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	R	0x0	HDMA page accumulation count register.

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

Functional Description

S/W Trigger DMA Mode

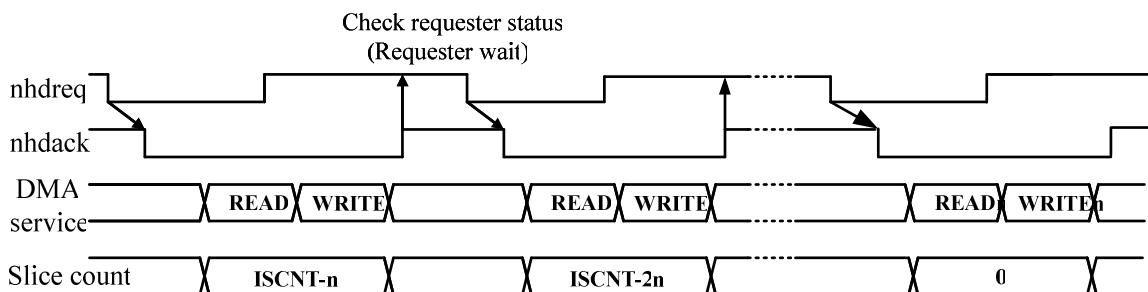
After AHB reset signal (hreset_n), the HDMA module enters the initial state. In the initial state, the HDMA doesn't move data between two memory blocks. When users finish programming configuration registers through AHB slave interface, the HDMA enter transfer state to begin to transfer data from source address to destination address. When the TC (Terminate Count) is asserted, the HDMA stop DMA operation.

H/W Trigger DMA Mode

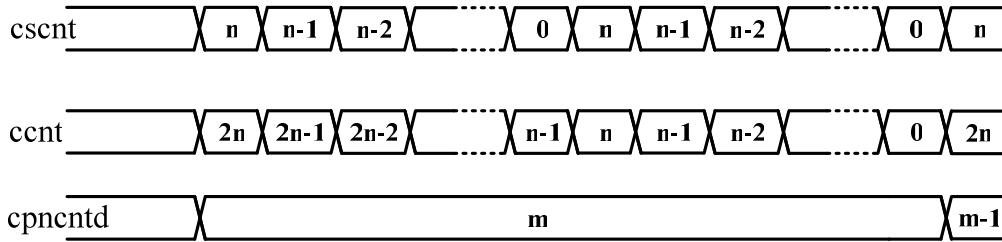
In hardware mode, when the peripheral devices want to transfer data to memory or devices need receive data from memory, it will assert request signal to HDMA. Then, the HDMA will assert acknowledge signal to peripheral device and start DMA service. In the DMA service, the HDMA read the data from source address and write the data to destination address. After one DMA service, the HDMA desserts acknowledge signal and check requester request signal. If requester de-asserts request signal, it means requester want to wait DMA service, the HDMA will de-assert acknowledge signal and wait the request. If the request signal is still assert, HDMA will start next DMA service. The following timing diagram shows the relationship between the requester and the responder.

H/W Trigger DMA with Slice Mode

In this hardware mode, when the peripheral devices want to transfer data to memory or devices need receive data from memory, it will assert request signal to HDMA. Then, the HDMA will assert acknowledge signal to peripheral device and start DMA service. In the DMA service, the HDMA read the data from source address and write the data to destination address. After one DMA service and slice count down to zero, the HDMA desserts acknowledge signal and the HDMA will wait the request. The following timing diagram shows the relationship between the requester and the responder.



The slice will be update when it count to zero and terminate counter doesn't equal to zero. The page number counter will minus one when terminate counter equals to zero. The following timing diagram shows the relationship between slice, terminate, and page number counters.



Chapter 9 LCD Controller

Overview

The LCD Controller basic function is to send out the image data from system memory to a LCD panel by properly formatting the raw image data stored in the memory.

The LCD performs translation of pixel-coded data into the required formats and timings to drive a single color LCDs. Support is provided for passive Super Twisted Nematic (STN) and active Thin Film Transistor (TFT) LCD display types:

STN displays STN display panels require algorithmic pixel pattern generation to provide pseudo gray scaling on mono, or color creation on color displays.

TFT displays TFT display panels require the digital color value of each pixel to be applied to the display data inputs.

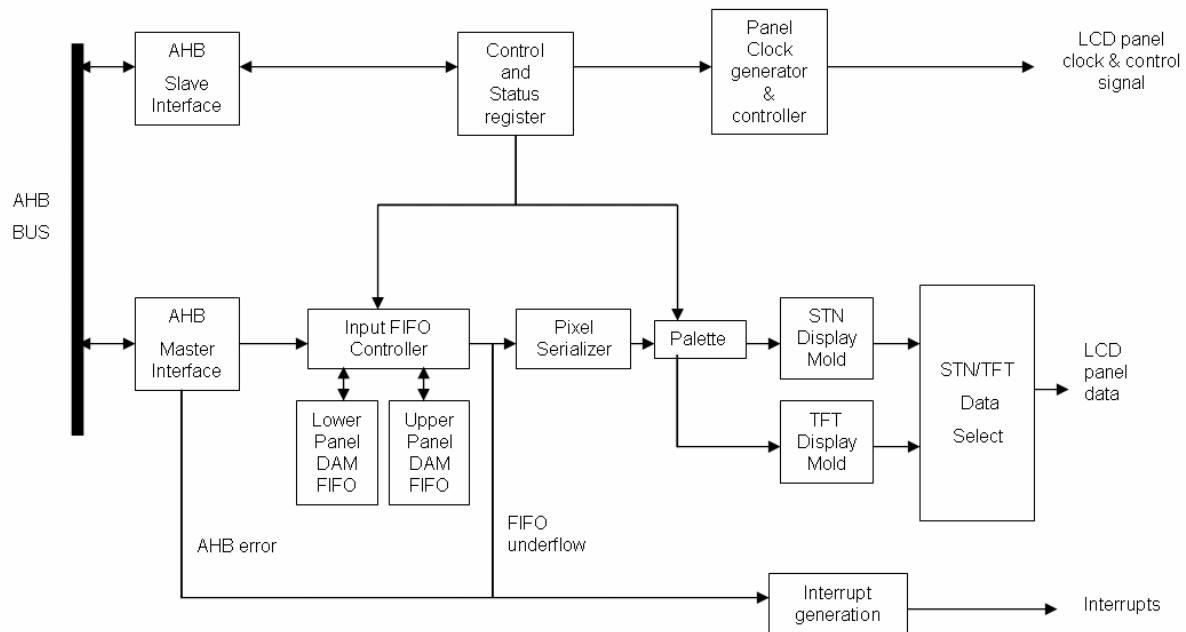
Key Features

- Dual 16-deep programmable 32-bit wide FIFOs for buffering incoming display data
- Supports single and dual panel mono *Super Twisted Nematic* (STN) displays with 4 or 8-bit interfaces
- Supports single and dual-panel color and monochrome STN displays
- Supports *Thin Film Transistor* (TFT) color displays
- Resolution programmable up to 1024 x 768
- 15 gray-level mono, 3375 color STN, and 32K color TFT support
- 1, 2, or 4 *bits-per-pixel* (bpp) palettized displays for mono STN
- 1, 2, 4 or 8 bpp palettized color displays for color STN and TFT
- 16 *bits-per-pixel* (bpp) true-color non-palettized, for color STN and TFT
- 24 bpp true-color non-palettized, for color TFT
- Programmable timing for different display panels
- 256 entry, 16-bit palette RAM, arranged as a 128 x 32-bit RAM physically
- frame, line and pixel clock signals
- AC bias signal for STN and data enable signal for TFT panels
- Patented gray scale algorithm

Architecture

The LCD controller not only formats the image data but also generates the timing control signals for the LCD panels. The LCD controller has adequate programmability to support most of the commercially available LCD panels. The raw image data from the system memory is buffered in the internal FIFO of configurable depth in order to support a variety of panels.

Block Diagram



Block Descriptions

Dual DMA FIFOs and associated control logic

The pixel data accessed from memory is buffered by two DMA FIFOs that can be independently controlled to cover single and dual-panel LCD types. Each FIFO is 128 words deep by 32 bits wide and can be cascaded to form an effective 32-word deep FIFO in single-panel mode. The water level marks within each FIFO are set so that each FIFO requests data when at least four locations become available. An interrupt signal is asserted if an attempt is made to read either of the two DMA FIFOs when they are empty, in other words an underflow condition has occurred.

Pixel serializer

This block reads the 32-bit wide LCD data from output port of the DMA FIFO and extracts 24, 16, 8, 4, 2, or 1 BPP data, depending on the current mode of operation. The LCD controller supports big-endian, little-endian, and WinCE data formats. In dual panel mode, data is alternately read from the upper and lower DMA FIFOs. Depending upon the mode of operation, you can use the extracted data to point to a color/gray scale value in the palette ram or it can be a true color value that you can apply directly to an LCD panel input. For each of the three supported data formats, the required data for each panel display pixel must be extracted from the data word. The nomenclature used in the figures is:

- *Little Endian Byte, Little Endian Pixel* (LBLP) order
- *Big Endian Byte, Big Endian Pixel* (BBBP) order
- *Little Endian Byte, Big Endian Pixel* (LBBP) order (this is the WinCE format).

RAM palette

The RAM-based palette is a 256 x 16 bit dual-port RAM physically structured as 128x32 bit. This allows two entries to be written into the palette from a single word write access. The least significant bit of the serialized pixel data is used to select between upper and lower halves of the palette RAM. Which half is selected depends on the byte ordering mode. In little-endian mode, the LSB being set selects the upper half, but in big-endian, the lower half of the palette is selected. WinCE byte ordering is little-endian, so the former case applies. The palette entries can be written and verified through this port. Port2 is used as a read-only port and is connected to the unpacker and gray scaler. For mono STN mode only the red palette field bits [4:1] are used. However, in STN color mode the green and blue [4:1] are also used. The red and blue pixel data can be swapped to support BGR data format using a Control Register bit. In 16 and 24 bpp TFT mode, the palette is bypassed and the output of the pixel serializer is used as the TFT panel data.

Gray scaler (for STN Mode)

A patented gray scale algorithm drives mono and color STN panels. This provides 15 gray scales for mono displays. In the case of STN color displays, the three color components (red, green, and blue) are gray scaled simultaneously which results in 3375 (15x15x15) colors being available. The gray scaler transforms each 4-bit gray value into a sequence of activity-per-pixel over several frames, relying to some degree on the display characteristics, to give the representation of gray scales and color.

Upper and lower panel formatters

Each formatter consists of three 3-bit (red, green, and blue) shift left registers. Red, green and

blue pixel data bit values from the gray scaler are concurrently shifted into the respective registers. When enough data is available, a byte is constructed by multiplexing the registered data to the correct bit position to satisfy the RGB data pattern of LCD panel. The byte is transferred to the three-byte FIFO which has enough space to store eight color pixels.

Panel clock generator and controller

The output of the panel clock generator block is the panel clock. This is a divided down version of **CLCDCLK**. It can be programmed in the range **CLCDCLK/2** to **CLCDCLK/33** to match the bpp data rate of the LCD panel.

The primary function of the timing controller block is to generate the horizontal and vertical timing panel signals. It also provides panel bias/enable signal. These timings are all register programmable through the AMBA AHB slave interface.

Interrupt generation

The PrimeCell CLCDC provides four individually maskable interrupts and a single combined interrupt. The single combined interrupt is asserted if any of the combined interrupts are asserted and unmasked.

Bus architecture

The LCD incorporates a master interface and can be connected directly onto the main system AHB bus, or alternatively to an AMBA AHB port of a memory controller, such as an SDRAM controller.

In addition to the AMBA AHB master interface, there is also an AMBA AHB slave interface for programming registers within the device. The slave interface and the master interface are separate AMBA AHB slaves and masters. This means that the LCD controller can be connected up in one of two ways:

- It can be built so that the master interface and the slave interface connect to a single multi-master AMBA AHB bus interface.
- The master interface can connect directly to a memory controller (for example an SDRAM controller) with an AMBA AHB slave interface, while the slave interface connects to the AMBA AHB bus.

AMBA AHB supports a wide range of on-chip bus sizes, from eight bits up to 1 024 bits. The PrimeCell CLCDC master and slave interfaces are implemented as 32-bit data bus devices only.

Registers

This section describes the control statutes of register for the LCD controller.

Register Summary

Name	Type	Offset	Size	Reset Value	Description
LCD_Timing0	RW	0x0000	W	0x00000000	Horizontal Axis Panel Control Register
LCD_Timing1	RW	0x0004	W	0x00000000	Vertical Axis Panel Control Register
LCD_Timing2	RW	0x0008	W	0x00000000	Clock and Signal Polarity Control Register
LCD_Timing3	RW	0x000C	W	0x00000000	Line End Control Register
LCD_UPBASE	RW	0x0010	W	0x00000000	Upper and Lower Panel Frame Base Address Registers
LCD_LPBASE	RW	0x0014	W	0x00000000	See Upper and Lower Panel Frame Base Address Registers
LCD_IMSC	RW	0x0018	W	0x00000000	Interrupt Mask Set/Clear Register
LCD_Control	RW	0x001C	W	0x00000000	Control Register
LCD_RIS	R	0x0020	W	0x00000000	Raw Interrupt Status Register
LCD_MIS	R	0x0024	W	0x00000000	Masked Interrupt Status Register
LCD_ICR	W	0x0028	W	0x00000000	Interrupt Clear Register
LCD_UPCURR	R	0x002C	W	0x00000000	Upper and Lower Panel Current Address Value Registers
LCD_LPCURR	R	0x0030	W	0x00000000	Upper and Lower Panel Current Address Value Registers
LCD_Palette	RW	0x200 – 0x3FC	W		Color Palette Register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

LCD_Timing0: Horizontal Axis Panel Control Register

LCD_Timing0 is a read/write register that controls the:

Horizontal Synchronization pulse Width (HSW)

Horizontal Front Porch (HFP) period

Horizontal Back Porch (HBP) period

Pixels-Per-Line (PPL).

Address: LCD Base + offset (0x0000)

bit	Attr	Reset Value	Description
31:24	R/W	0x0	Horizontal back porch is the number of CLCP periods between the falling edge of CLLP and the start of active data. Program with value minus 1. The 8-bit HBP field specifies the number of pixel clock periods inserted at the beginning of each line or row of pixels. After the line clock for the previous line has been deasserted, the value in HBP counts the number of pixel clocks to wait before starting the next display line. HBP can generate a delay of 1-256 pixel clock cycles.
23:16	R/W	0x0	Horizontal front porch is the number of CLCP periods between the end of active data and the rising edge of CLLP . Program with value minus 1. The 8-bit HFP field sets the number of pixel clock intervals at the end of each line or row of pixels, before the LCD line clock is pulsed. When a complete line of pixels is transmitted to the LCD driver, the value in HFP counts the number of pixel clocks to wait before asserting the line clock. HFP can generate a period of 1-256 pixel clock cycles.
15:8	R/W	0x0	Horizontal synchronization pulse width is the width of the CLLP signal in CLCP periods. Program with value minus 1. The 8-bit HSW field specifies the pulse width of the line clock in passive mode, or the horizontal synchronization

			pulse in active mode.
7:2	R/W	0x0	Pixels-per-line. Actual pixels-per-line = 16 * (PPL + 1). The PPL bit field specifies the number of pixels in each line or row of the screen. PPL is a 6-bit value that represents between 16 and 1 024 PPL. PPL controls how much data is read from the DMA input buffers through to the gray scalar.
1:0	R/W	0x0	Reserved, do not modify, read as zero, write as zero.

LCD_Timing1: Vertical Axis Panel Control Register

LCD_Timing1 is a read/write register that controls the:

Number of Lines-Per-Panel (LPP)

Vertical Synchronization pulse Width (VSW)

Vertical Front Porch (VFP) period

Vertical Back Porch (VBP) period

Address: LCD Base + offset (0x0004)

bit	Attr	Reset Value	Description
31:24	R/W	0x0	Vertical back porch is the number of inactive lines at the start of a frame, after vertical synchronization period. Program to 0 on passive displays or reduced contrast results. The 8-bit VBP field specifies the number of line clocks inserted at the beginning of each frame. The VBP count starts just after the vertical synchronization signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit field in passive mode. After this has occurred, the count value in VBP sets the number of line clock periods inserted before the next frame. VBP generates from 0–255 extra line clock cycles.
23:16	R/W	0x0	Vertical front porch is the number of inactive lines at the end of frame, before vertical synchronization period. Program to 0 on passive displays or reduced contrast results. The 8-bit VFP field specifies the number of line clocks to insert at the end of each frame. When a complete frame of pixels is transmitted to the LCD display, the value

			in VFP is used to count the number of line clock periods to wait. After the count has elapsed the vertical synchronization signal, CLFP , is asserted in active mode, or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates from 0–255 line clock cycles.
15:10	R/W	0x0	Vertical synchronization pulse width is the number of horizontal synchronization lines. Must be small (for example, program to zero) for passive STN LCDs. Program to the number of lines required minus one. The higher the value the worse the contrast on STN LCDs. The 6-bit VSW field specifies the pulse width of the vertical synchronization pulse. The register is programmed with the number of line clocks in VSync minus one. Number of horizontal synchronization lines. Must be small (for example, program to 0) for passive STN LCDs. Program to the number of lines required minus 1. The higher the value the worse the contrast on STN LCDs.
9:0	R/W	0x0	A line per panel is the number of active lines per screen. Program to number of lines required minus 1. The LPP field specifies the total number of lines or rows on the LCD panel being controlled. LPP is a 10-bit value that allows 1-1 024 lines. The register is programmed with the number of lines per LCD panel minus 1. For dual panel displays this register is programmed with the number of lines on each of the upper and lower panels.

LCD_Timing2: Clock and Signal Polarity Control Register

LCD_Timing2 is a read/write register that controls the CLCDC timing.

Address: LCD Base + offset (0x0008)

bit	Attr	Reset Value	Description
31:27	R/W	0x0	Upper five bits of Panel Clock Divisor. The ten-bit PCD field, comprising PCD_HI and PCD_LO (bits [4:0]), is used to derive the LCD panel clock frequency CLCP from

			the CLCDCLK frequency: CLCP = CLCDCLK / (PCD+2). For mono STN displays with a four or eight-bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. For color STN displays, 22/3 pixels are output per CLCP cycle, therefore the panel clock is 0.375 times. For TFT displays the pixel clock divider can be bypassed by setting the LCDTiming2[26] BCD bit.
26	R/W	0x0	Bypass pixel clock divider. Setting this to 1 bypass the pixel clock divider logic. This is mainly used for TFT displays.
25:16	R/W	0x0	Clocks per line. This field specifies the number of actual CLCP clocks to the LCD panel on each line. This is the number of PPL divided by 1 for TFT, 4 or 8 for mono passive, or 22/3 for color passive, minus one. This must be correctly programmed in addition to PPL for the LCD controller to work correctly.
15	R/W	0x0	Reserved, do not modify, read as zero, write as zero.
14	R/W	0x0	Invert output enable: 0 = CLAC output pin is active HIGH in TFT mode 1 = CLAC output pin is active LOW in TFT mode. The <i>Invert Output Enable</i> (IOE) bit is used to select the active polarity of the output enable signal in TFT mode. In this mode, the CLAC pin is used as an enable that indicates to the LCD panel when valid display data is available. In active display mode, data is driven onto the LCD data lines at the programmed edge of CLCP when CLAC is in its active state.
13	R/W	0x0	Invert panel clock: 0 = Data is driven on the LCDs data lines on the rising-edge of CLCP 1 = Data is driven on the LCDs data lines on the falling-edge of CLCP . The IPC bit is used to select the edge of the panel clock on which pixel data is driven out onto the LCD data lines.

12	R/W	0x0	<p>Invert horizontal synchronization:</p> <p>0 = CLLP pin is active HIGH and inactive LOW</p> <p>1 = CLLP pin is active LOW and inactive HIGH.</p> <p>The <i>Invert HSync</i> (IHS) bit is used to invert the polarity of the CLLP signal.</p>
11	R/W	0x0	<p>Invert vertical synchronization:</p> <p>0 = CLFP pin is active HIGH and inactive LOW</p> <p>1 = CLFP pin is active LOW and inactive HIGH.</p> <p>The <i>Invert VSync</i> (IVS) bit is used to invert the polarity of the CLFP signal.</p>
10:6	R/W	0x0	<p>AC bias pin frequency.</p> <p>The AC bias pin frequency is only applicable to STN displays, which require the pixel voltage polarity to be periodically reversed to prevent damage due to DC charge accumulation. Program this field with the required value minus 1 to apply the number of line clocks between each toggle of the AC bias pin, CLAC. This field has no effect if the CLCDC is operating in TFT mode when the CLAC pin is used as a data enable signal.</p>
5	R/W	0x0	<p>This bit drives the CLCDCLKSEL signal which is used as the select signal for the external LCD clock multiplexor.</p>
4:0	R/W	0x0	<p>Lower five bits of Panel Clock Divisor. The ten-bit PCD field, comprising PCD_HI (bits [31:27]) and PCD_LO, is used to derive the LCD panel clock frequency CLCP from the CLCDCLK frequency,</p> <p>CLCP = CLCDCLK / (PCD+2). For mono STN displays with a four or eight-bit interface, the panel clock is a factor of four and eight down on the actual individual pixel clock rate. For color STN displays, 2 2/3 pixels are output per CLCP cycle, so the panel clock is 0.375 times. You can bypass the pixel clock divider for TFT displays by setting the LCDTiming2[26] BCD bit.</p>

LCD_Timing3: Line End Control Register

LCD_Timing3 is a read/write register that controls the enabling of line-end signal **CLLE**. When

enabled, a positive pulse, four **CLCDCLK** periods wide, is output on **CLLE** after a programmed delay set by the LED bits. If the line-end signal is disabled then it is held permanently LOW.

Address: LCD Base + offset (0x000C)

bit	Attr	Reset Value	Description
31:17	-	-	Reserved, do not modify, read as zero, write as zero.
16	R/W	0x0	LCD Line end enable: 0 = CLLE disabled (held LOW) 1 = CLLE signal active.
15:7	R/W	0x0	Reserved, do not modify, read as zero, write as zero.
6:0	R/W	0x0	Line-end signal delay from the rising-edge of the last panel clock, CLCP . Program with number of CLCDCLK clock periods minus 1.

LCD_UPBASE

LCDUPBASE and **LCDLPBASE** are the color LCD DMA Frame Address Registers. They are read/write registers used to program the base address of the frame buffer. **LCDUPBASE** is used for:

TFT displays

Single panel STN displays

The upper panel of dual panel STN displays.

LCDLPBASE is used for the lower panel of dual panel STN displays. You must initialize **LCDUPBASE** (and **LCDLPBASE** for dual panels) before enabling the **CLCDC**. You can change the value mid-frame to enable double-buffered video displays to be created. These registers are copied to the corresponding current registers at each LCD vertical synchronization. This event causes the **LNU** bit and an optional interrupt to be generated. You can use the interrupt to reprogram the base address when generating double-buffered video. Bits [1:0] are 0 value when read.

Address: LCD Base + offset (0x0010)

bit	Attr	Reset Value	Description
31:2	R/W	0x0	LCD upper panel base address. This is the start address of the upper panel frame data in memory and is word aligned.
1:0	-	-	Reserved, do not modify, read as zero, write as zero.

LCD_LPBASE:

Address: LCD Base + offset (0x0014)

bit	Attr	Reset Value	Description
31:2	R/W	0x0	LCD lower panel base address. This is the start address of the lower panel frame data in memory and is word aligned.
1:0	-	-	Reserved, do not modify, read as zero, write as zero.

LCD_IMSC: Interrupt Mask Set/Clear Register

LCD_IMSC is the Interrupt Mask Set/Clear Register. Setting bits in this register enables the corresponding raw interrupt LCDRIS bit values to be passed to the LCDMIS Register.

Address: LCD Base + offset (0x0018)

bit	Attr	Reset Value	Description
4	R/W	0x0	AHB master error interrupt enable
3	R/W	0x0	Vertical compare interrupt enable
2	R/W	0x0	Next base update interrupt enable
1	R/W	0x0	FIFO underflow interrupt enable
0	-	-	Reserved.

LCD_Control: Control Register

LCD_Control is the Control Register. It is a read/write register that controls the mode in which the CLCDC operates.

Address: LCD Base + offset (0x001C)

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	R/W	0x0	LCD DMA FIFO Watermark level: 0 = HBUSREQM is raised when either of the two DMA FIFOs have four or more empty locations 1 = HBUSREQM is raised when either of the DMA FIFOs have eight or more empty locations.
15:14	-	-	Reserved.
13:12	R/W	0x0	Generate interrupt at: 00 = start of vertical synchronization

			01 = start of back porch 10 = start of active video 11 = start of front porch.
11	R/W	0x0	LCD power enable: 0 = power not gated through to LCD panel and CLD[23:0] signals disabled, (held LOW) 1 = power gated through to LCD panel and CLD[23:0] signals enabled, (active).
10	R/W	0x0	Big-endian pixel ordering within a byte: 0 = little-endian pixel ordering within a byte 1= big-endian pixel ordering within a byte. The BEPO bit selects between little and big-endian pixel packing for 1, 2, and 4 bpp display modes. It has no effect on 8 or 16 bpp pixel formats.
9	R/W	0x0	Big-endian byte order: 0 = little-endian byte order 1 = big-endian byte order.
8	R/W	0x0	RGB or BGR format selection: 0 = RGB normal output 1 = BGR red and blue swapped.
7	R/W	0x0	LCD interface is dual panel STN: 0 = single panel LCD is in use 1 = dual panel LCD is in use.
6	R/W	0x0	Monochrome LCD has an 8-bit interface. This bit controls whether monochrome STN LCD uses a 4 or 8-bit parallel interface: 0 = mono LCD uses 4-bit interface 1 = mono LCD uses 8-bit interface. LcdMono8 has no meaning in other modes and must be programmed to 0.
5	R/W	0x0	LCD is TFT: 0 = LCD is an STN display, use gray scaler 1 = LCD is TFT, do not use gray scaler.

4	RW	0x0	STN LCD is monochrome (black and white): 0 = STN LCD is color 1 = STN LCD is monochrome. This bit has no meaning in TFT mode.
3:1	RW	0x0	LCD bits per pixel: 000 = 1 bpp 001 = 2 bpp 010 = 4 bpp 011 = 8 bpp 100 = 16 bpp 101 = 24 bpp (TFT panel only) 110 = reserved 111 = reserved.
0	RW	0x0	LCD controller enable: 0 = CLLP , CLCP , CLFP , CLAC , and CLLE disabled (held LOW) 1 = CLLP , CLCP , CLFP , CLAC , and CLLE enabled (active).

LCD_RIS: Raw Interrupt Status Register

LCD_RIS is a read-only register. On a read it returns five bits that can generate interrupts when set.

Address: LCD Base + offset (0x0020)

bit	Attr	Reset Value	Description
4	R	0x0	AHB Master bus error status, set when the AHB Master encounters a bus error response from a slave.
3	R	0x0	Vertical compare, set when one of the four vertical regions, selected through the LCDControl Register, is reached.
2	R	0x0	LCD next address base update, mode dependent, set when the Current Base Address Registers have been successfully updated by the next Address Registers. Signifies that a new next address can be loaded if double buffering is in use.
1	R	0x0	FIFO underflow, set when either the upper or lower DMA FIFOs have been read accessed when empty causing an underflow condition to occur.
0	-	-	Reserved, read as zero.

LCD_MIS: Masked Interrupt Status Register

LCD_MIS is a read-only register. It is a bit-by-bit logical AND of the LCD_RIS Register and the LCD_IMSC Register. Interrupt lines correspond to each interrupt. A logical OR of all interrupts is provided to the system interrupt controller.

Address: LCD Base + offset (0x0024)

bit	Attr	Reset Value	Description
31:5	R	0x0	Reserved, read as zero
4	R	0x0	AHB master error interrupt status bit
3	R	0x0	Vertical compare interrupt status bit
2	R	0x0	LCD next base address update interrupt status bit
1	R	0x0	FIFO underflow interrupt status bit
0	-	-	Reserved.

LCD_ICR: Interrupt Clear Register

The LCD_ICR is a write-only register. Writing logic 1 to the relevant bit clears the

corresponding interrupt.

Address: LCD Base + offset (0x0028)

bit	Attr	Reset Value	Description
31:5	W	0x0	Reserved, do not modify, write as zero
4	W	0x0	Clear AHB Master error interrupt
3	W	0x0	Clear vertical compare interrupt
2	W	0x0	Clear LCD next base address update interrupt
1	W	0x0	Clear FIFO underflow interrupt
0	-	-	Reserved.

LCD_UPCURR:

LCDUPCURR is read-only registers that contain an approximate value of the upper panel data DMA addresses when read. The registers can change at any time and therefore can only be used as a mechanism for coarse delay.

Address: LCD Base + offset (0x002C)

bit	Attr	Reset Value	Description
31:0	R	0x0	Contains the approximate current upper panel data DMA address

LCDLPCURR:

LCDLPCURR is read-only registers that contain an approximate value of the lower panel data DMA addresses when read. The registers can change at any time and therefore can only be used as a mechanism for coarse delay.

Address: LCD Base + offset (0x0030)

bit	Attr	Reset Value	Description
31:0	R	0x0	Contains the approximate current lower panel data DMA address

LCD_Palette: Color Palette Register

The LCD_Palette Register contains 256 palette entries organized as 128 locations of two entries per word. Only TFT displays use all of the palette entry bits. Each word location contains two palette entries. This means that 128 word locations are used for the palette. When configured for little-endian byte ordering, bits [15:0] are the lower numbered palette entry and bits [31:16] are

the higher numbered palette entry. When configured for big-endian byte ordering this is reversed because bits [31:16] are the low numbered palette entry and bits [15:0] are the high numbered entry.

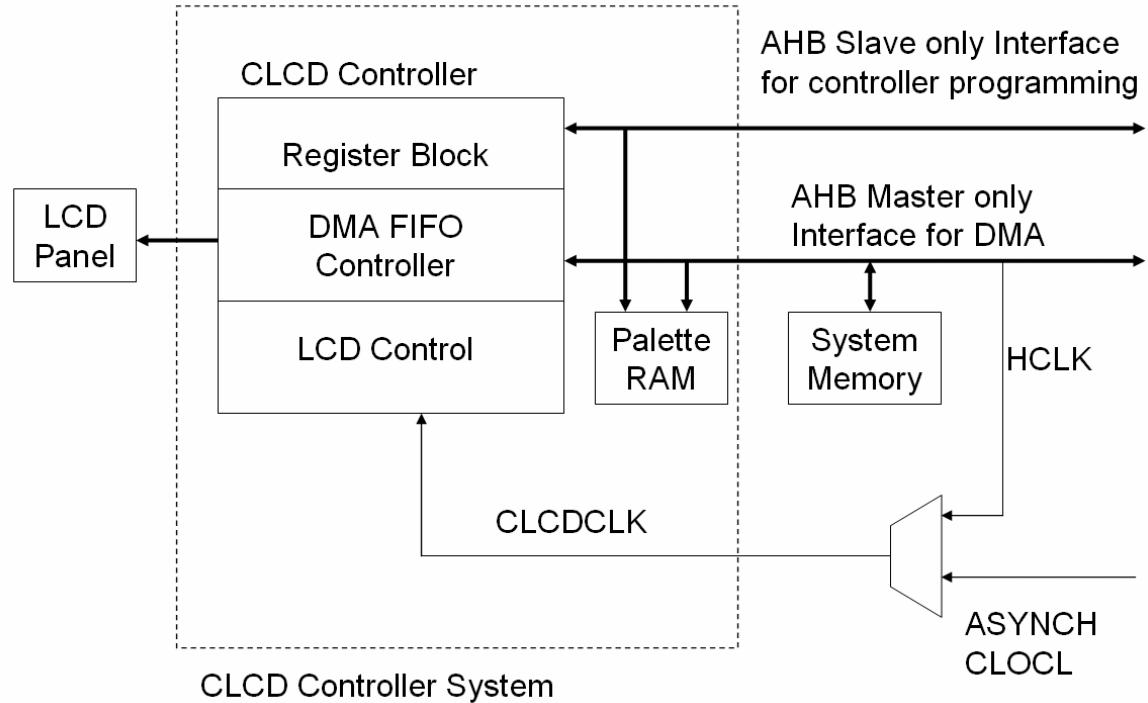
Address: LCD Base + offset (0x0200 - 0x03FC)

bit	Attr	Reset Value	Description
31	RW	0x0	Intensity/unused.
30:26	RW	0x0	Blue palette data.
25:21	RW	0x0	Green palette data.
20:16	RW	0x0	Red palette data.
15	RW	0x0	Intensity bit. Can be used as the LSB of the R, G, and B inputs to a 6:6:6 TFT display, doubling the number of colors to 64K, where each color has two different intensities.
14:10	RW	0x0	Blue palette data.
9:5	RW	0x0	Green palette data.
4:0	RW	0x0	Red palette data. For STN displays, only the four MSBs (bits [4:1]) are used. For monochrome displays only the red palette data is used. All of the Palette Registers have the same bit fields.

Functional Description

Operation

Whenever the CLCD is enabled it starts reading the image data from the system memory by read DMA operation. The image data represents the number of pixels depending on the Bit-Per-Pixel (BPP) value programmed in the CLCD registers. The image data is just a logical color value for BPP values other than 16 and 24bpp. The physical color values are stored in the 16 bit wide Palette RAM in the form of a look-up table. The pixel value (for 1, 2, 4 or 8 BPP) represents the index for the color look up table.



The Palette RAM is outside the CLCD controller block but it is a part of CLCD Controller system. This kind of hierarchy gives the flexibility for configuring the Palette RAM for any technology library. The main LCD panel control logic operates on a different clock (CLCDCLK) but can be configured to operate on the bus clock with the use of external clock multiplexer. The output CLCDCLKSEL from the controller is nothing but a register bit (to be discussed later) that can be used to control the multiplexer select as shown in the above diagram.

Chapter 10 CCIR-656 VIP Controller

Overview

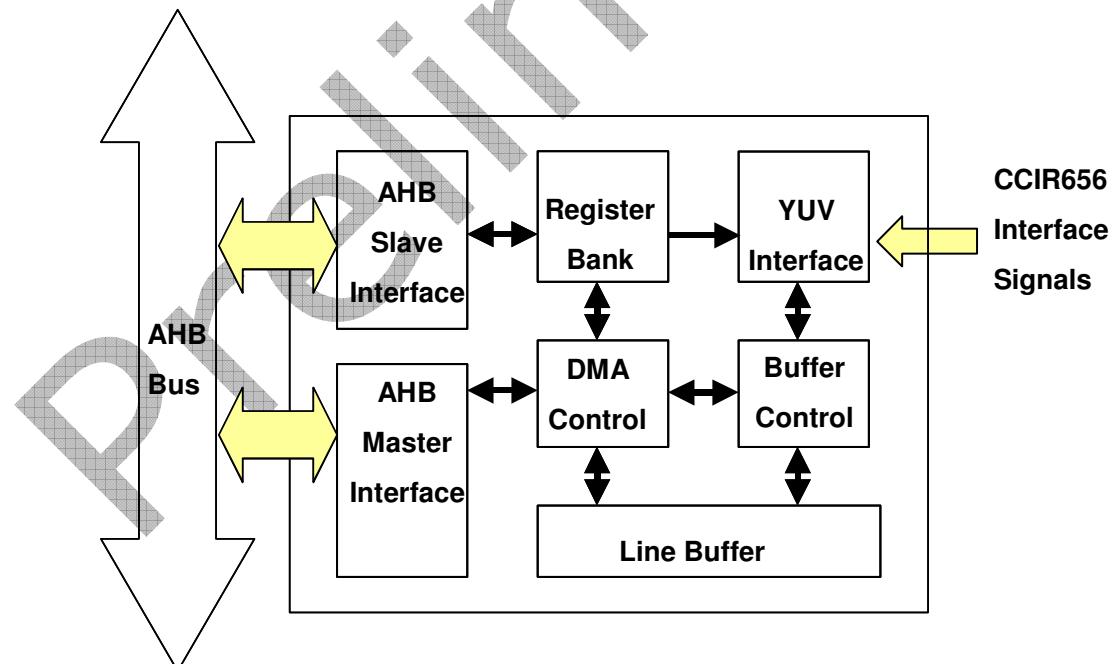
The CCIR-656 VIP can translate YUV raw data from video stream to AHB bus. It is used to accept ccir656 video bit stream from external camera module and transfer the YUV raw data into system main memory by AHB bus.

Key Features

- Support CCIR-656 YCbCr 4:2:2 raster video input for 8bit mode in 525/60 NTSC and 625/50 PAL video system
- Embedded H & V sync
- Provide YUV 4:2:2/4:2:0 Output
- For NTSC, Support up to D1 (720x480) resolution, free format in interlaced mode, 30 frames/sec (interlaced 60Hz)
- For PAL, Support up to D1 (720x576) resolution, free format in interlaced mode, 25 frames/sec (interlace 50Hz)

Architecture

Block Diagram



Block Descriptions

AHB Master/Slave Interface

The interface between AMBA AHB and Register Bank, DMA Controller

Register Bank

Configurable registers

DMA Controller

Manage the memory buffer

YUV interface

Transfer CCIR-656 4:2:2 format to YCbCr 4:2:0 format

Buffer control

Memory buffer controller

Line buffer

Line data captured buffer

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
VIP_AHBR_CTRL	0x0000	W	0x00000001	AHB write control register.
VIP_INT_MASK	0x0004	W	0x00000000	Interrupt Mask register.
VIP_INT_STS	0x0008	W	0x00000000	Interrupt status register.
VIP_STS	0x000C	W	0x00000000	Status register.
VIP_CTRL	0x0010	W	0x00000020	VIP control register.
VIP_CAPTURE_F1S_A_Y	0x0014	W	0x00000000	Capture raw data frame 1 start address for Y.
VIP_CAPTURE_F1S_A_Cb	0x0018	W	0x00000000	Capture raw data frame 1 start address for Cr.
VIP_CAPTURE_F1S_A_Cb	0x001C	W	0x00000000	Capture raw data frame 1 start address for Cb.
VIP_CAPTURE_F2S_A_Y	0x0020	W	0x00000000	Capture raw data frame 2 start address for Y.
VIP_CAPTURE_F2S_A_Cb	0x0024	W	0x00000000	Capture raw data frame 2 start address for Cr.
VIP_CAPTURE_F2S_A_Cb	0x0028	W	0x00000000	Capture raw data frame 2 start address for Cb.
VIP_FB_SR	0x002C	W	0x0000000b	Frame buffer status register for capturing raw data

VIP_FS	0x0030	W	0x02d001e6	Frame data size register
VIP_CROP	0x0038	W	0x00000000	Cropping start upper left point from D1 to other little resolution
VIP_CRM	0x003C	W	0x00000000	Y/CB/CR color modification
VIP_RESET	0x0040	W	0x00000000	Capture engine reset
VIP_L_SFT	0x0044	W	0x00000000	Line Shifter from first line

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

VIP_AHBR_CTRL

Address: Operational Base + offset (0x0000)

AHB write control register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x1	<p>AHB Data Maximum Burst Length for Reading from H/W.</p> <p>This register will set the maximum length to transmit data to AHB bus. The actual data length to be transmitted will be decided by H/W automatically. For example, if INCR8 is set, only 8 or 4 will be the actual length.</p> <p>The following is the meaning.</p> <ul style="list-style-type: none"> 000 --- unused 001 --- INCR 010 --- unused 011 --- unused 100 --- unused 101 --- INCR8 110 --- unused 111 --- INCR16

VIP_INT_MASK

Address: Operational Base + offset (0x0004)

Interrupt Mask register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2	RW	0x0	Capture data line end happened interrupt enable, 1:enable (for debug, just a cycle pulse)
1	RW	0x0	Capture frame loss happened interrupt enable. 0: Disable 1: Enable
0	RW	0x0	Capture complete interrupt enable. 0: Disable 1: Enable

VIP_INT_STS

Address: Operational Base + offset (0x0008)

Interrupt status register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2	R	0x0	Capture data line end happened interrupt (Read Clear) (for debug)
1	R	0x0	Capture frame loss happened interrupt. (Read Clear) 0: No interrupt happen 1: Interrupt happen
0	R	0x0	Capture complete interrupt. (Read Clear) 0: No interrupt happen 1: Interrupt happen

VIP_STS

Address: Operational Base + offset (0x000C)

Status register

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0x0	FIFO overflow (Read Clear) 1: FIFO overflow

VIP_CTRL

Address: Operational Base + offset (0x0010)

VIP control register

Bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10	RW	0x0	CCIR656 capture format 0: NTSC 1: PAL
9	RW	0x0	Posedge/Negedge capture by pixel clock 0: Positive edge 1: Negative edge
8	RW	0x0	Ping-Pong mode enable 0: Continuous mode 1: Ping-Pong mode Note1: Bit5 priority > Bit8 Note2: Only both Bit5 and Bit8 be set to 0 can enable continuous mode
7	RW	0x0	Field capture for ccir format 0: Field 0 start 1: Field 1 start

6	RW	0x0	422 output enable 0: 420 output (write to memory) 1: 422 output (write to memory)
5	RW	0x1	One frame stop enable 0: Continuous mode or Ping-pong mode 1: One frame complete stop (after stop will reset this register) Note1: Exist performance limitation(1/900 s CPU access time) for real time application. Note2: Bit5 priority > Bit8 Note3: Only both Bit5 and Bit8 be set to 0 can enable continuous mode
4:1	-	-	Reserved
0	RW	0x0	Enable capturing (set and clear by host) To set this bit to enable capturing, and clear it by host to disable capturing. (Set and clear by host) 0: Disable 1: Enable

VIP_CAPTURE_F1SA_Y

Address: Operational Base + offset (0x0014)

Capture raw data frame 1 start address for Y

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Raw Frame 1 Starting Address Register for Y (The bits 1-0 in this address must be 0)

VIP_CAPTURE_F1SA_Cb

Address: Operational Base + offset (0x0018)

Capture raw data frame 1 start address for Cb

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Raw Frame 1 Starting Address Register for Cb (The bits 1-0 in this address must be 0)

VIP_CAPTURE_F1SA_Cr

Address: Operational Base + offset (0x001C)

Capture raw data frame 1 start address for Cr

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Raw Frame 1 Starting Address Register for Cr (The bits 1-0 in this address must be 0)

VIP_CAPTURE_F2SA_Y

Address: Operational Base + offset (0x0020)

Capture raw data frame 2 start address for Y

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Raw Frame 2 Starting Address Register for Y (The bits 1-0 in this address must be 0)

VIP_CAPTURE_F2SA_Cb

Address: Operational Base + offset (0x0024)

Capture raw data frame 2 start address for Cb

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Raw Frame 2 Starting Address Register for Cb (The bits 1-0 in this address must be 0)

VIP_CAPTURE_F1SA_Cr

Address: Operational Base + offset (0x0028)

Capture raw data frame 2 start address for Cr

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Raw Frame 2 Starting Address Register for Cr (The bits 1-0 in this address must be 0)

VIP_FB_SR

Address: Operational Base + offset (0x002C)

Frame buffer status register for capturing raw data

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:8	RW	0x0	Frame Number. Complete VIP number
7:4	-	-	Reserved.
3	R	0x1	Indicate the latest used Frame buffer number, for example, if Bit0 and Bit 1 are both 1 and Bit3 is 1, means that Frame 2 is captured finally. 0: Frame 1 1: Frame 2
2	RW	0x0	Indicate Frame Loss (set by H/W and clear by HOST) 0: No frame loss 1: Frame loss occurred
1	RW	0x1	Status bit to indicate current status of Frame 2(set by H/W and clear by HOST) 0: data not ready 1: data ready Note: After reading this register, HOST shall assign new

			buffer addresses to “Capture Raw Frame 1/2 Starting Address Register for Y/Cb/Cr” and clear the status register for H/W capturing next frame to keep Ping-Pong mode enable.
0	RW	0x1	<p>Status bit to indicate current status of Frame 1(set by H/W and clear by HOST)</p> <p>0: data not ready 1: data ready</p> <p>Note: After reading this register, HOST shall assign new buffer addresses to “Capture Raw Frame 1/2 Starting Address Register for Y/Cb/Cr” and clear the status register for H/W capturing next frame to keep Ping-Pong mode enable.</p>

VIP_FS_(Frame Data Size Register default for D1 resolution 720x480/720x576)

Address: Operational Base + offset (0x0030)

Frame data size register

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:16	RW	0x2d0	Pixel number per line CCIR-656 Width Up to 720 (0x2D0)
15:10	-	-	Reserved.
9:0	RW	0x01e6	Line numbers per frame CCIR-656 Height (Up to 0x01e0 for NTSC setting 0x23e for PAL setting)

VIP_CROP

Address: Operational Base + offset (0x0038)

Cropping start upper left point from D1 to other little resolution

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25:16	RW	0x0	The X-coordinate of the cropping start point at up-left corner
15:10	-	-	Reserved.
9:0	RW	0x0	The Y-coordinate of the cropping start point at up-left corner

VIP_CRM

Address: Operational Base + offset (0x003C)

Y/CB/CR color modification

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26	R/W	0x0	Y direction, 0-decrease 1-increase
25	R/W	0x0	Cb direction, 0-decrease 1-increase
24	R/W	0x0	Cr direction, 0-decrease 1-increase

23:16	R/W	0x0	Y value
15:8	R/W	0x0	Cb value
7:0	R/W	0x0	Cr value

VIP_RESET

Address: Operational Base + offset (0x0040)

Capture engine reset

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Engine Reset (Refer to reset flow of video input processor) Value: 0x76543210 to reset

VIP_L_SFT

Address: Operational Base + offset (0x0044)

Line Shifter from first line

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x0	Line Shifter from first line, it is used in non-standard ccir656 input(not precisely 480/576 active line). Set this register can cut the lines at the first of both fields. Valid value: 0~15

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

Initial configuration

After HW/SW reset, CPU must initially configure VIP by control registers via AHB bus.

The configurations include frame data address, frame size, output format....etc. After the initialization, CPU must enable VIP when external Video Device (Such as camera module or video decoder) is ready.

Normal operation

Interrupt from VIP to CPU

After finishing initial configuration described above, VIP will start to translate the ccir656 video signals into Y/Cb/Cr frame data and store them into Memory at AHB bus according to the address specified in control register at initialization. CPU will be idle to VIP operation until frame data translation complete and VIP issues an interrupt to CPU. When CPU gets the interrupt, it must decide to assign next frame address to VIP or not. If VIP does not get the instructions from CPU to store new frame data before frame buffer filled completely, it will stop the operations or continue the operations without frame buffer address update according to the stop mode setting in control register(0x10). Two stop modes are available in VIP, named as ‘One Frame Stop Mode’ and ‘Ping Pong Mode’, we will describe them in the following section.

Below are the operations of different settings and control flows example between CPU and VIP.

1. VIP Software Reset

This section will introduce how to reset VIP by software. While configuring 0x40 register with 0x76543210, VIP will be software reset after 200 cycles of AHB.

2. One frame stop mode on NTSC (Only Frame 1 can be used)

Setup VIP_CAPTURE_F1SA_<Y,Cb,Cr>,

(Setup them as a continuous memory mode as easy to dump to YUV file), configure register 0x38 to set crop start point and configure register 0x30 to set your frame size. Before trigger VIP to start capture, Frame1 buffer status in register 0x2C(bit 0) must be clear to 1'b0. Then configure register 0x10 to be 0x21 value to start one frame stop mode on NTSC capture. After one frame captured, VIP will automatic stop. If register 0x4 is set to value 0x3, then an interrupt from VIP will be issued. After capturing, the image Y, Cb, Cr data will be stored at main memory location defined by VIP_CAPTURE_F1SA_Y, VIP_CAPTURE_F1SA_Cb, and VIP_CAPTRUE_F1SA_Cr separately.

3. One frame stop mode on PAL (Only Frame 1 can be used)

Setup VIP_CAPTURE_F1SA_<Y,Cb,Cr>,

(Setup them as a continuous memory mode as easy to dump to YUV file), configure register 0x10 first to be value 0x420, configure register 0x38 to set crop start point and configure register 0x30 to set your frame size. Before trigger VIP to start capture, Frame1 buffer status in register 0x2C(bit 0) must be clear to 1'b0. Then configure register 0x10 to be 0x421 value to start one frame stop mode on PAL capture. After one frame captured, VIP will automatic stop. If register 0x4 is set to value 0x3, then an interrupt from VIP will be issued. After capturing, the image Y, Cb, Cr data will be stored at main memory location defined by VIP_CAPTURE_F1SA_Y, VIP_CAPTURE_F1SA_Cb, and VIP_CAPTRUE_F1SA_Cr separately.

4. Ping-Pong mode on NTSC (Both Frame 1 & Frame 2 can be used)

Setup VIP_CAPTURE_F1SA_<Y,Cb,Cr>, and VIP_CAPTURE_F2SA_<Y,Cb,Cr>.

(Setup them as a continuous memory mode as easy to dump to YUV file), configure register 0x38 to set crop start point and register 0x30 to be your frame size. Before trigger VIP to start capture, Frame1 & Frame2 buffer status in register 0x2C(bit 1:0) must be clear to 2'b00. Then configure register 0x10 to be 0x101 value to start ping-pong mode on NTSC capture. After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically, and CPU must assign new address pointer of frame 1 and clear the frame 1 status, thus VIP will capture the third frame automatically(by F1 address pointer) without any stop and so on for the following frames. But if CPU did not update the frame buffer pointer and status, the VIP will stop after both 2 frame buffer(F1 & F2) are at data ready state(bit [1:0] of reg 0x2c is 2'b11).

5. Ping-Pong mode on PAL (Both Frame 1 & Frame 2 can be used)

Setup VIP_CAPTURE_F1SA_<Y,Cb,Cr>, and VIP_CAPTURE_F2SA_<Y,Cb,Cr>.

(Setup them as a continuous memory mode as easy to dump to YUV file), configure register 0x10 first to be value 0x420, and configure register 0x38 to set crop start point and configure

register 0x30 to set your frame size. Before trigger VIP to start capture, Frame1 & Frame2 buffer status in register 0x2C(bit 1:0) must be clear to 2'b00. Then configure register 0x10 to be 0x501 value to start ping-pong mode on NTSC capture. After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically, and CPU must assign new address pointer of frame 1 and clear the frame 1 status, thus VIP will capture the third frame automatically(by F1 address pointer) without any stop and so on for the following frames. But if CPU did not update the frame buffer pointer and status, the VIP will stop after both 2 frame buffer(F1 & F2) are at data ready state(bit [1:0] of reg 0x2c is 2'b11).

6. Continous mode on NTSC (BOTH Frame 1 & Frame 2 must be used)

Setup **VIP_CAPTURE_F1SA_<Y,Cb,Cr>, and VIP_CAPTURE_F2SA_<Y,Cb,Cr>**.

S/W configure register 0x38 to set crop start point and configure register 0x30 to set your frame size. Before trigger VIP to start capture, Frame1 & Frame2 buffer status in register 0x2C(bit 1:0) must be clear to 2'b00. Then configure register 0x10 to be 0x1 value to start continuous mode on NTSC capture. If you setup register 0x4 to be value 0x1 you will get a complete interrupt from VIP after every frame end without any frameloss interrupt. After capturing, you just only display these frames by VOP. If bus isn't busy, you can directly rewrite register after interrupt. S/W must rewrite memory start address registers before blanking-line end. So you can control continuous mode directly. Some time it is too late to rewrite Y/Cb/Cr start address registers; new frame will put a cover over last memory block. VIP can't detect this status. Note that the continuous mode will use both F1 & F2 pointer just like in ping-pong mode, but the difference between continuous mode and ping-pong mode is that continuous mode will never stop the VIP unless CPU disable VIP directly even the F1 & F2 status are both in data ready state.

7. Continuous mode on PAL (Both Frame 1 & Frame 2 must be used)

Setup **VIP_CAPTURE_F1SA_<Y,Cb,Cr>, and VIP_CAPTURE_F2SA_<Y,Cb,Cr>**.

S/W configure register 0x10 first to be value 0x400, and configure register 0x38 to set crop start point and configure register 0x30 to set your frame size. Before trigger VIP to start capture, Frame1 & Frame2 buffer status in register 0x2C(bit 1:0) must be clear to 2'b00. Then configure register 0x10 to be 0x401 value to start series mode on PAL capture. If you setup register 0x4 to be value 0x1 you will get a complete interrupt from VIP after every frame end without any frameloss interrupt. After capturing, you just only display these frames by LCD. If bus isn't busy, you can directly rewrite memory start address registers after interrupt. S/W must rewrite register before blanking-line end. So you can control continuous mode directly. Some time it is too late to rewrite Y/Cb/Cr start address registers; new frame will put a cover over last memory block. VIP can't detect this status. Note that the continuous mode will use both F1 & F2 pointer just like in ping-pong mode, but the difference between continuous mode and ping-pong mode is that continuous mode will never stop the VIP unless CPU disable VIP directly even the F1 & F2 status are both in data ready state

Chapter 11 3D/2D Graphics Accelerator Overview

The 3D/2D Graphics Accelerator includes a Mali200 pixel processor and a MaliGP2 geometry processor, and a memory management unit (MMU). The GPU provides its associated software is compatible with the major graphics standards, OpenGL ES 2.0, OpenGL ES 1.0, and OpenVG 1.0.

The Mali200 pixel processor uses a list of primitives generated by the MaliGP2 geometry processor to produce a final image that is displayed on the screen. The MaliGP2 programmable geometry processor that generates lists of primitives for the Mali200 pixel processor to draw. To efficient memory access, a full-featured Memory Management Unit (MMU) is embedded and it provides all memory accesses from the pixel and geometry processor to use the MMU for access checking and translation.

Key Features

Mali200 Pixel Processor Features:

- Programmable fragment shader
- Access to framebuffer from fragment shaders
- Alpha blending
- Arbitrary memory reads and writes.
- Complete non-power-of-2 texture support
- Cube mapping
- Dynamic recursion
- Fast dynamic branching
- Fast trigonometric functions, including arctangent
- Full floating-point arithmetic
- Framebuffer blend with destination Alpha
- High Dynamic Range (HDR) textures and framebuffers
- Indexable texture samplers
- Line, quad, triangle and point sprites
- Multiple render targets
- No limit on program length
- Perspective Anisotropic Filtering (AF)
- Perspective correct texturing
- Point sampling, bilinear, and trilinear filtering
- Programmable mipmap level-of-detail biasing and replacement
- Register indirect jumps
- Stencil buffering, 8-bit
- Two-sided stencil
- Unlimited dependent texture reads
- Virtualized texture samplers
- 4-level hierarchical Z and stencil operations

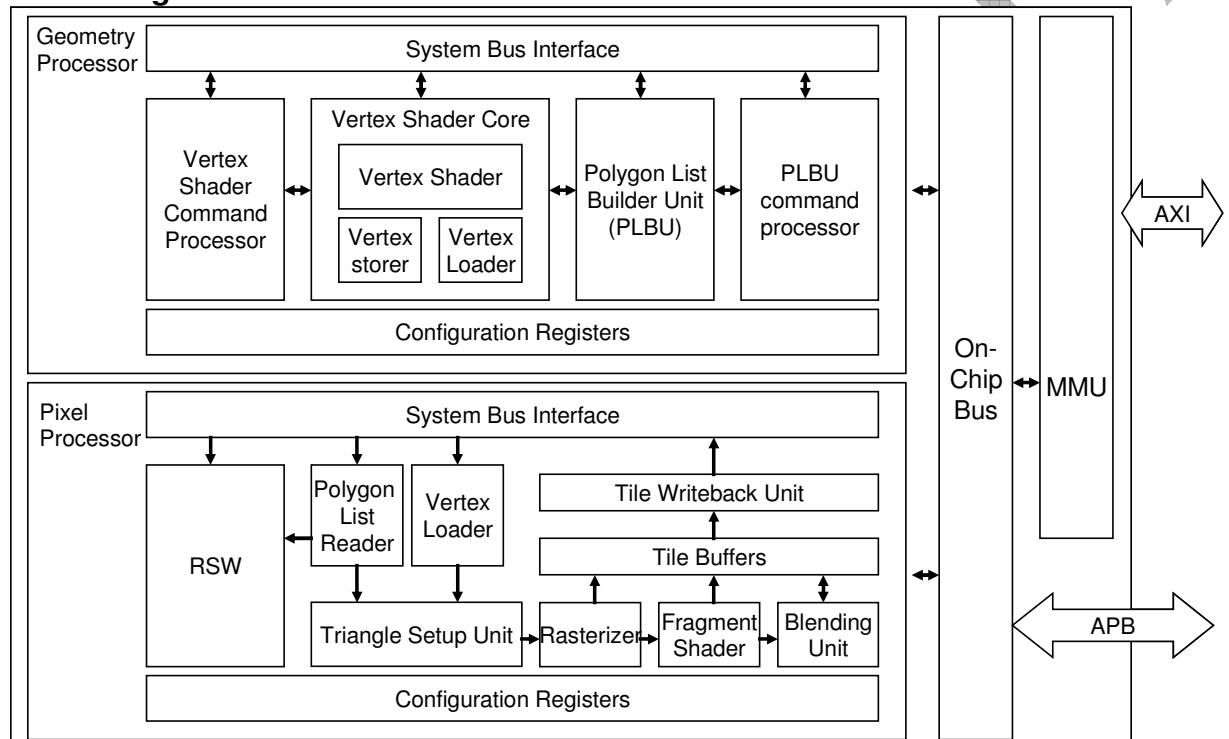
- 4 times and 16 times Full Scene Anti-Aliasing (FSAA)
- 4-bit per texel texture compression

MaliGP2 Geometry Processor Features:

- Programmable vertex shader
- Autonomous operation tile list generation
- Flexible input and output formats
- Indexed and non-indexed geometry input
- Primitive constructions with points, lines, triangles and quads

Architecture

Block Diagram



Block Descriptions

The MaliGP2 geometry processor performs the geometry processing for the system. Four main tasks are performed:

1. Transform and Lighting (T&L).

2. Primitive assembly.

This involves the PLB linking vertices together to form different primitives.

3. Automatic back face culling.

This involves culling of all primitives on the back side of the object that would not be visible, because only the back face would be visible from the viewing plane.

4. Primitive list assembly.

Because the pixel processor is a tile-based renderer, the geometry processor must prepare a list of all primitives required for it to render. For each primitive, the PLB writes a list entry for each tile that the primitive can touch.

Rendering is the term that describes the various tasks the pixel processor performs. During rendering, the pixel processor uses the information from the polygon list to produce a final frame buffer image.

The Mali200 pixel processor performs the following rendering operations:

1. Triangle setup.

This prepares the primitive for rendering by calculating various data that is required to rasterize and shade the primitive.

2. Rasterization.

The primitive is divided into independent fragments. These are pixel-sized piece of primitive that the shader pipeline processes. Fragments that might be visible proceed to the fragment shading stage, and fragments that are certain not to be visible are discarded.

3. Fragment shading.

This stage determines how the fragment actually looks. In general, the pixel processor calculates a color for the fragment.

4. Blending

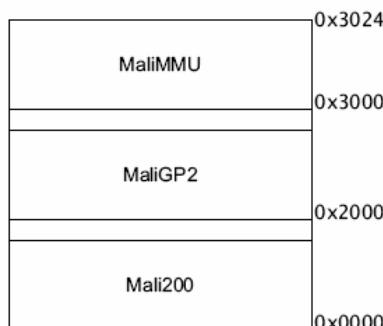
The fragment is blended into the frame buffer to produce the final image.

5. Producing the frame buffer.

After blending, the fragment becomes a pixel at a certain position in the tile buffer. If no other fragment overwrites that position, the fragment becomes a pixel in the final frame buffer. Multi-sampling techniques to obtain sharper final images can be applied to the pixel at this stage. When the internal tile buffer is completely rendered, it is written to the frame buffer in main memory.

Registers

The GPU processors are configured through a 16KB address space, divided between pixel processor, geometry processor, and memory management unit (MMU). Below figure is the configuration of GPU.



The Mali200 pixel renderer configuration registers consist of an 8KB address space divided by a set of register blocks. The memory extends from a base address of 0x0000 to a maximum

address of 0x1F40. Below Figure shows the pixel register map split into regions.



Frame Registers

The Frame Registers contain frame data that is static during the rendering of a frame, and not required in the memory data structures. Because this data is typically written to in bursts from the driver, the registers are laid out as 18 contiguous registers.

Write-back Registers

The processor is equipped with three write-back units WB0, WB1, and WB2, that can be configured independently by three similar blocks.

Below Table shows the Write-back Registers

Unit	Register base address
WB0	0x0100
WB1	0x0200
WB2	0x0300

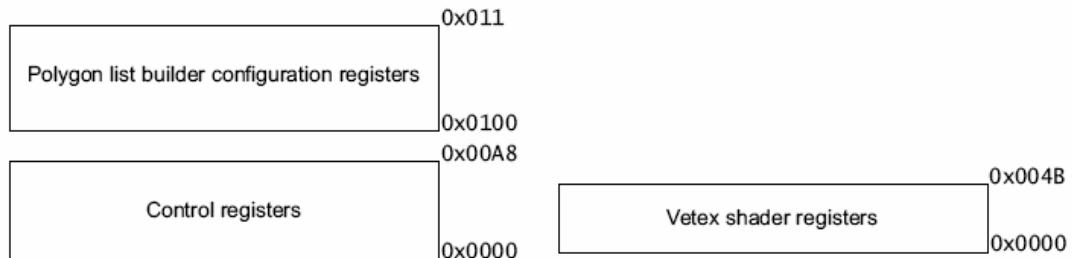
Each register block consists of a unique set of the following registers. In the description, WB_x means either WB0, WB1, or WB2 depending on the base address. In the addresses, the underscore means the differentiating nibble between the WBs, for example, 0x0_1C is 0x011C for WB0 0x021C for WB1 and 0x031C for WB2.

Management Registers

The Management Registers covering 0x1000-0x10F0 are all control and configuration registers that are not directly connected to rendering of a frame.

The MaliGP2 Geometry Processor configures register into three memory locations:

- The Polygon List Builder Configuration Registers are 64Bytes
- The Control Registers are 168Bytes
- The Vertex Shader Register are 304Bytes



Control registers

The control registers are a set of registers that monitor and control the operation of the geometry processor. The APB bus accesses and controls these registers.

Polygon list builder configuration registers

The PLB configuration registers are a set of registers that control the operation of the polygon list builder, but can only be written from the PLB command list.

Vertex shader registers

The vertex shader configuration registers are a set of registers that control the operation of vertex shader, but can only be written from the vertex shader command list.

Registers Summary

Pixel Processor Register (Frame Register)

Name	Offset	Type	Reset Value	Description
REND_LISTADDR	0x0000	RW	0x00000000	Renderer List Address Register
REND_RSW_BASE	0x0004	RW	0x00000000	Renderer State Word Base Address Register
REND_VERTEX_B_ASE	0x0008	RW	0x00000000	Renderer State Word Base Address Register
FEATURE_ENABLE	0x000C	RW	0x00000002	Feature Enable Register
Z_CLEAR_VALUE	0x0010	RW	0x00000009	Z Clear Value Register
STENCIL_CLEAR_VALUE	0x0014	RW	0x00000000	Stencil Clear Value Register
ABGR_CLEAR_VA_LUE_0	0x0018	RW	0x00000000	ABGR Clear Value 0 Register
ABGR_CLEAR_VA_LUE_1	0x001C	RW	0x00000000	ABGR Clear Value 1 Register
ABGR_CLEAR_VA_LUE_2	0x0020	RW	0x00000000	ABGR Clear Value 2 Register
ABGR_CLEAR_VA_LUE_3	0x0024	RW	0x00000000	ABGR Clear Value 3 Register
BOUNDING_BOX_LEFT_RIGHT	0x0028	RW	0x00000000	Bounding Box Left Right register
BOUNDING_BOX_	0x002C	RW	0x00000000	Bounding Box Bottom register

BOTTOM				
FS_STACK_ADDR	0x0030	RW	0x00000000	FS Stack Address Register
FS_STACK_SIZE_A ND_INIT_VAL	0x0034	RW	0x00000000	FS Stack Size and Initial Value Register
ORIGIN_OFFSET_X	0x0040	RW	0x00000000	Origin Offset X Register
ORIGIN_OFFSET_Y	0x0044	RW	0x00000000	Origin Offset Y Register
SUBPIXEL_SPECIF IER	0x0048	RW	0x00000075	Subpixel Specifier Register
TIEBREAK_MODE	0x004C	RW	0x00000000	Tiebreak mode Register

(Write-Back Register)

Name	Offset	Type	Reset Value	Description
WB0_SOURCE_SEL ECT	0x0100	RW	0x00000000	WBx Source Select Register
WB0_TARGET_AD DR	0x0104	RW	0x00000000	WBx Target Address Register
WB0_TARGET_PIX EL_FORMAT	0x0108	RW	0x00000000	WBx Target Pixel Format Register
WB0_TARGET_AA FORMAT	0x010C	RW	0x00000000	WBx Target AA Format Register
WB0_TARGET_LA YOUT	0x0110	RW	0x00000000	WBx Target Layout
WB0_TARGET_SC ANLINE_LENGTH	0x0114	RW	0x00000000	WBx Target Scanline Length
WB0_TARGET_FL AGS	0x0118	RW	0x00000000	WBx Target Flags Register
WB0_MRT_ENABL E	0x011C	RW	0x00000000	WBx MRT Enable Register
WB0_MRT_OFFSET	0x0120	RW	0x00000000	WBx MRT Offset Register
WB0_GLOBAL_TE ST_ENABLE	0x0124	RW	0x00000000	WBx Global Test Enable Register
WB0_GLOBAL_TE ST_REF_VALUE	0x0128	RW	0x00000000	WBx Global Test Reference Value Register
WB0_GLOBAL_TE ST_CMP_FUNC	0x012C	RW	0x00000000	WBx Global Test Compare Function Register
WB1_SOURCE_SEL ECT	0x0200	RW	0x00000000	WBx Source Select Register
WB1_TARGET_AD DR	0x0204	RW	0x00000000	WBx Target Address Register
WB1_TARGET_PIX EL_FORMAT	0x0208	RW	0x00000000	WBx Target Pixel Format Register
WB1_TARGET_AA FORMAT	0x020C	RW	0x00000000	WBx Target AA Format Register
WB1_TARGET_LA	0x0210	RW	0x00000000	WBx Target Layout

YOUT				
WB1_TARGET_SC_ANLINE_LENGTH	0x0214	RW	0x00000000	WBx Target Scanline Length
WB1_TARGET_FLAGS	0x0218	RW	0x00000000	WBx Target Flags Register
WB1_MRT_ENABLE	0x021C	RW	0x00000000	WBx MRT Enable Register
WB1_MRT_OFFSET	0x0220	RW	0x00000000	WBx MRT Offset Register
WB1_GLOBAL_TEST_ENABLE	0x0224	RW	0x00000000	WBx Global Test Enable Register
WB1_GLOBAL_TEST_REF_VALUE	0x0228	RW	0x00000000	WBx Global Test Reference Value Register
WB1_GLOBAL_TEST_CMP_FUNC	0x022C	RW	0x00000000	WBx Global Test Compare Function Register
WB2_SOURCE_SELECT	0x0300	RW	0x00000000	WBx Source Select Register
WB2_TARGET_ADDRESS	0x0304	RW	0x00000000	WBx Target Address Register
WB2_TARGET_PIXEL_FORMAT	0x0308	RW	0x00000000	WBx Target Pixel Format Register
WB2_TARGET_AA_FORMAT	0x030C	RW	0x00000000	WBx Target AA Format Register
WB2_TARGET_LAYOUT	0x0310	RW	0x00000000	WBx Target Layout
WB2_TARGET_SC_ANLINE_LENGTH	0x0314	RW	0x00000000	WBx Target Scanline Length
WB2_TARGET_FLAGS	0x0318	RW	0x00000000	WBx Target Flags Register
WB2_MRT_ENABLE	0x031C	RW	0x00000000	WBx MRT Enable Register
WB2_MRT_OFFSET	0x0320	RW	0x00000000	WBx MRT Offset Register
WB2_GLOBAL_TEST_ENABLE	0x0324	RW	0x00000000	WBx Global Test Enable Register
WB2_GLOBAL_TEST_REF_VALUE	0x0328	RW	0x00000000	WBx Global Test Reference Value Register
WB2_GLOBAL_TEST_CMP_FUNC	0x032C	RW	0x00000000	WBx Global Test Compare Function Register

(Management Register)

Name	Offset	Type	Reset Value	Description
VERSION	0x1000	RO	VERSION	The Register holds a static version number for the processor
CURRENT_RENDER_LIST	0x1004	RW	0x00000000	Current Renderer List Address

LIST_ADDR				Register at 0x1004
STATUS	0x1008	RW	0x00000000	Status Register
CTRL_MGMT	0x100C	WO	-	Control Management Register
INT_RAWSTAT	0x1020	RW	0x00000000	Interrupt Rawstat Register
INT_CLEAR	0x1024	WO	-	Interrupt Clear Register
INT_MASK	0x1028	RW	0x000001FF	Interrupt Mask Register
INT_STATUS	0x102C	RO	0x00000000	Interrupt Status Register
WRITE_BOUNDARY_ENABLE	0x1040	RW	0x00000000	Write Boundary Enable Register
WRITE_BOUNDARY_LOW	0x1044	RW	0x00000000	Write Boundary Low Register
WRITE_BOUNDARY_HIGH	0x1048	RW	0x00000000	Write Boundary High Register
WRITE_BOUNDARY_ADDRESS	0x104C	RO	0x00000000	Write Boundary Address Register
BUS_ERROR_STATUS	0x1050	RO	0x00000000	Bus Error Status Register
WATCHDOG_DISABLE	0x1060	RW	0x00000000	Watchdog Disable Register
WATCHDOG_TIMEOUT	0x1064	RW	0x000F4240	Watchdog Timeout Register
PERF_CNT_0_ENABLE	0x1080	RW	0x00000000	Performance Counter 0 Enable Register
PERF_CNT_0_SRC	0x1084	RW	0x00000000	Performance Counter 0 SRC Register
PERF_CNT_0_LIMIT	0x1088	RW	0x00000000	Performance Counter 0 Limit Register
PERF_CNT_0_VALUE	0x108C	RW	0x00000000	Performance Counter 0 Value Register
PERF_CNT_1_ENABLE	0x10A0	RW	0x00000000	Performance Counter 1 Enable Register
PERF_CNT_1_SRC	0x10A4	RW	0x00000000	Performance Counter 1 SRC Register
PERF_CNT_1_LIMIT	0x10A8	RW	0x00000000	Performance Counter 1 Limit Register
PERF_CNT_1_VALUE	0x10AC	RW	0x00000000	Performance Counter 1 Value Register

Geometry Processor Register (Control Register)

Name	Offset	Type	Reset Value	Description
GP CONTR REG VSCL START ADDR	0x0000	RW	0x00000000	GP Control Register VSCL Start Address

GP CONTR_REG_VSCL_END_ADDR	0x0004	RW	0x00000000	GP Control Register VSCL End
GP CONTR_REG_PLBCL_START_ADDR	0x0008	RW	0x00000000	GP Control Register PLBCL Start Address
GP CONTR_REG_PLBCL_END_ADDR	0x000C	RW	0x00000000	GP Control Register PLBCL End Address
GP CONTR_REG_PLB_ALLOC_START_ADDR	0x0010	RW	0x00000000	GP Control Register PLB Allocate Start Address
GP CONTR_REG_PLB_ALLOC_END_ADDR	0x0014	RW	0x00000000	GP Control Register PLB Allocate End Address
GP CONTR_REG_CMD	0x0020	WO	-	GP Control Register Command
GP CONTR_REG_INTERRUPT_RAWSTAT	0x0024	RW	0x00000000	GP Control Register Interrupt Rawstat
GP CONTR_REG_INTERRUPT_CLEAR	0x0028	WO	-	GP Control Register Interrupt Clear
GP CONTR_REG_INTERRUPT_MASK	0x002C	RW	0x00000000	GP Control Register Interrupt Mask
GP CONTR_REG_INTERRUPT_STAT	0x0030	RO	0x00000000	GP Control Register Interrupt Status
GP CONTR_REG_WRITE_BOUND_LOW	0x0034	RW	0x00000000	GP Control Register Write Boundary Low
GP CONTR_REG_WRITE_BOUND_HI	0x0038	RW	0x00000000	GP Control Register Write Boundary High
GP CONTR_REG_PERF_CNT_0_ENABLE	0x003C	RW	0xFFFFFFF0	GP Control Register Performance Counter 0 Enable
GP CONTR_REG_PERF_CNT_1_ENABLE	0x0040	RW	0x00000000	GP Control Register Performance Counter 1 Enable
GP CONTR_REG_PERF_CNT_0_SRC	0x0044	RW	0x00000000	GP Control Register Performance Counter 0 Source
GP CONTR_REG_PERF_CNT_1_SRC	0x0048	RW	0x00000000	GP Control Register Performance Counter 1 Source
GP CONTR_REG_PERF_CNT_0_V	0x004C	RO	0x00000000	GP Control Register Performance Counter 0 Value
GP CONTR_REG_PERF_CNT_1_VAL	0x0050	RO	0x00000000	GP Control Register Performance Counter 1 Value
GP CONTR_REG_P	0x0054	RW	0x00000000	GP Control Register

ERF_CNT_0_LIMIT				Performance Counter 0 Limit
GP CONTR_REG_P	0x0058	RW	0x00000000	GP Control Register
ERF_CNT_1_LIMIT				Performance Counter 1 Limit
GP CONTR_REG_S	0x0068	RO	0x00000000	GP Control Register Status
GP CONTR_REG_VERSION	0x006C	RO	VERSION	GP core version
GP CONTR_REG_VSCL_INITIAL_ADDRESS	0x0080	RO	0x00000000	GP Control Register VSCL Initial Address
GP CONTR_REG_PLBCL_INITIAL_ADDRESS	0x0084	RO	0x00000000	GP Control Register PLBCL Initial Address
GP CONTR_REG_WRITE_BOUNDARY_ERROR_ADDR	0x0088	RO	0x00000000	GP Control Register Write Error Address
GP CONTR_REG_AXI_BUS_ERROR_STAT	0x0094	RO	0x00000000	GP Control AXI Bus Error Status
GP CONTR_REG_WATCHDOG_DISABLE	0x00A0	RW	0x00000000	GP Control Register Watchdog Disable
GP CONTR_REG_WATCHDOG_TIMEOUT	0x00A4	RW	0x000F4240	GP Control Register Watchdog Timeout

(PLB Configuration Register)

Name	Offset	Type	Reset Value	Description
GP_PLB_CONF_REG_VERTEX_ARRAY_ADDR	0x0100	WO	0x00000000	GP PLB Configuration Register Vertex Array Address
GP_PLB_CONF_REG_INDEX_ARRAY_ADDR	0x0101	WO	0x00000000	GP PLB Configuration Register Index Array Address
GP_PLB_CONF_REG_POINT_SIZE_ADDRESS	0x0102	WO	0x00000000	GP PLB Configuration Register Point Size Address
GP_PLB_CONF_REG_HEAP_START_ADDRESS	0x0103	WO	0x00000000	GP PLB Configuration Register Heap Start Address Register
GP_PLB_CONF_REG_HEAP_END_ADDRESS	0x0104	WO	0x00000000	GP PLB Configuration Register Heap End Address Register
GP_PLB_CONF_REG	0x0105	WO	0x00000000	GP PLB Configuration Register

G_VIEWPORT_TOP				Viewport
GP_PLB_CONF_REG_VPORT_BOTTOM	0x0106	WO	0x00000000	GP PLB Configuration Register Viewport Bottom
GP_PLB_CONF_REG_VPORT_LEFT	0x0107	WO	0x00000000	GP PLB Configuration Register Viewport Left
GP_PLB_CONF_REG_VPORT_RIGHT	0x0108	WO	0x00000000	GP PLB Configuration Register Viewport Right
GP_PLB_CONF_REG_SCREEN_SIZE	0x0109	WO	0x00000000	GP PLB Configuration Register Screen Size
GP_PLB_CONF_REG_OFFSET_VERTEX_ARRAY	0x010A	WO	0x00000000	GP PLB Configuration Register Offset Vertex Array
GP_PLB_CONF_REG_PARAMETERS	0x010B	WO	0x00000000	GP PLB Configuration Register Parameters
GP_PLB_CONF_REG_TILE_SIZE	0x010C	WO	0x00000000	GP PLB Configuration Register Tile Size
GP_PLB_CONF_REG_POINT_SIZE	0x010D	WO	0x00000000	GP PLB Configuration Register Point Size
GP_PLB_CONF_REG_Z_NEAR	0x010E	WO	0x00000000	GP PLB Configuration Register Z Near
GP_PLB_CONF_REG_Z_FAR	0x010F	WO	0x3F800000	GP PLB Configuration Register Z Far

(Vertex Shader Register)

x = 0~15

Name	Offset	Type	Reset Value	Description
GP_VS_CONF_REG_INP_ADDRx	0x0000~0x001E	WO	0x00000000	GP VS Configuration Register Input Address(x) at 0x0000 + (2*x)
GP_VS_CONF_REG_INP_SPECx	0x0001~0x001F	WO	0x0000003F	GP VS Configuration Register Input Specifier(x) at 0x0001 + (2*x)
GP_VS_CONF_REG_OUTP_ADDRx	0x0020~0x003E	WO	0x00000000	GP VS Configuration Register Output Address(x)
GP_VS_CONF_REG_OUTP_SPECx	0x0021~0x003F	WO	0x0000003F	GP VS Configuration Register Output Specifier(x)
GP_VS_CONF_REG_PROG_PARAM	0x0040	WO	0x00000000	GP VS Configuration Program Parameter Create
GP_VS_CONF_REG_PREFETCH	0x0041	WO	0x00000000	GP VS Configuration Register Prefetch
GP_VS_CONF_REG	0x0042	WO	0x0F000000	GP VS Configuration Register

_OPMOD				OPMOD
GP_VS_CONF_REG_VERTICES_ALT_STRIDE	0x0043	WO	0x00000000	GP VS Configuration Register Vertices Alternative Stride
GP_VS-CONF_REG_INPUT_ALT_STRIDE_0	0x0044	WO	0x00000000	GP VS Configuration Register Input Alternative Stride 0
GP_VS-CONF_REG_INPUT_ALT_STRIDE_1	0x0045	WO	0x00000000	GP VS Configuration Register Input Alternative Stride 1
GP_VS-CONF_REG_INPUT_ALT_STRIDE_2	0x0046	WO	0x00000000	GP VS Configuration Register Input Alternative Stride 2
GP_VS-CONF_REG_INPUT_ALT_STRIDE_3	0x0047	WO	0x00000000	GP VS Configuration Register Input Alternative Stride 3
GP_VS_CONF_REG_OUTP_ALT_STRIDE_0	0x0048	WO	0x00000000	GP VS Configuration Register Output Alternative Stride 0
GP_VS_CONF_REG_OUTP_ALT_STRIDE_1	0x0049	WO	0x00000000	GP VS Configuration Register Output Alternative Stride 1
GP_VS_CONF_REG_OUTP_ALT_STRIDE_2	0x004A	WO	0x00000000	GP VS Configuration Register Output Alternative Stride 2
GP_VS_CONF_REG_OUTP_ALT_STRIDE_3	0x004B	WO	0x00000000	GP VS Configuration Register Output Alternative Stride 3

MMU Configuration Register

Name	Offset	Type	Reset Value	Description
MMU_DTE_ADDR	0x0000	RW	0x00000000	MMU Current Page Table Address Register
MMU_STATUS	0x0004	RO	0x00000018	MMU Status Register
MMU_COMMAND	0x0008	WO	-	MMU Command Register
MMU_PAGE_FAULT_ADDR	0x000C	RO	0x00000000	MMU Logical Address of Last Page Fault Register
MMU_ZAP_ONE_LINE	0x0010	WO	-	MMU Zap Cache Line Register
MMU_INT_RAWSTATUS	0x0014	RW	0x00000000	MMU Raw Interrupt Status Register
MMU_INT_CLEAR	0x0018	WO	-	MMU Interrupt Clear Register
MMU_INT_MASK	0x001C	RW	0x00000000	MMU Interrupt Mask Register
MMU_INT_STATUS	0x0020	RO	0x00000000	MMU Interrupt Status Register

Notes: All registers are WORD (32 bits) access

Detail Register Description

Pixel Processor Registers

REND_LIST_ADDR

Address: Operational Base + offset (0x0000)

The REND_LIST_ADDR register characteristics are to holds the start address of the polygon list for the current frame. Software must not write to this register during the rendering Operation. This register is available in all pixel processor configurations.

Bit	Attr	Reset Value	Description
31:5	RW	0x0	REND_LIST_ADDR; Start address of the polygon list to use for the frame
4:0	-	-	Reserved.

REND_RSW_BASE

Address: Operational Base + offset (0x0004)

Renderer State Word Base Address Register

Bit	Attr	Reset Value	Description
31:6	RW	0x0	REND_RSW_BASE; Default renderer state word base address
5:0	-	-	Reserved.

REND_VERTEX_BASE

Address: Operational Base + offset (0x0008)

Renderer Vertex Base Register

Bit	Attr	Reset Value	Description
31:6	RW	0x0	REND_VERTEX_BASE; Default vertex bundles base address
5:0	-	-	Reserved.

FEATURE_ENABLE

Address: Operational Base + offset (0x000C)

Feature Enable Register

Bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0x0	SUMMATE_QUAD_COVER; When set to 1, the coverage-to-alpha operates on a 2x2 fragment quad, and not individual fragments. This means that all 16 samples from a 2x2 quad are counted and converted to an alpha between 0.0 (0 samples) and 1.0 (16 samples), and that the alpha value is set for all the fragments in the 2x2 fragment quad.
5	RW	0x0	ORIGIN_LOWER_LEFT;

			This bit indicates whether the co-ordinate system for the screen XY position has its origin in the upper-left corner, Y axis increasing downwards or to lower-left corner, Y axis increasing upwards. The only hardware function that is currently affected by this bit is the Position Register function, and tile indices. The pixel processor considers the upper-left corner to be the origin. This bit must be set for use with OpenGL and is cleared for use with Direct3D.
4	RW	0x0	EARLYZ_DISABLE2; Setting this bit disables the second of two Early-Z mechanisms. Only use for debugging. For normal use, enable or disable Early-Z with the EARLYZ_ENABLE bit.
3	RW	0x0	EARLYZ_DISABLE1; Setting this bit disables the first of two Early-Z mechanisms. Only use for debugging. For normal use, enable or disable Early-Z with the EARLYZ_ENABLE bit.
2	RW	0x0	IGNORE_CLEAR_BITS; Setting this bit to 1 hinders the clearing of the RGB tile buffers. This feature is useful for special applications.
1	RW	0x1	EARLYZ_ENABLE; Setting this bit to 1 enables the Early Z-test mechanism in the rasterizer. EarlyZ is enabled by default. Enabling this test increases the performance in high-overdraw situations by performing multiple Z-tests per clock at the rasterizer level. The Early Z-test is not effective on Greater-Than depth test functions.
0	RW	0x0	FP_TILEBUF_ENABLE; Setting this bit to 1 sets the tile buffer to FP16 (1:5:10) component format instead of 8-bit component format. Enabling this feature has the following consequences: <ul style="list-style-type: none"> • Multiple render targets cannot be used • Internal multi-sampling or super-sampling anti-aliasing cannot be used • External write-back anti-aliasing technique cannot be used.

Z_CLEAR_VALUE

Address: Operational Base + offset (0x0010)

Z Clear Value Register

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.

23:0	RW	0x09	<p>Z_CLEAR_VALUE; The 24-bit depth value of the Z tile buffer is logically cleared whenever processing of a new tile starts. If you do not want the Z tile buffer to be cleared, the content of the Z tile buffer can be pre-loaded by using a textured quad and Z-replacement technique.</p>
------	----	------	--

STENCIL_CLEAR_VALUE

Address: Operational Base + offset (0x0014)

Stencil Clear Value Register

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	<p>STENCIL_CLEAR_VALUE; The 8-bit stencil value of the stencil tile buffer is logically cleared whenever processing of a new tile starts. If you do not want the stencil tile buffer to be cleared, the content of the stencil tile buffer can be pre-loaded by using a textured quad and stencil replacement technique.</p>

ABGR_CLEAR_VALUE_0

Address: Operational Base + offset (0x0018)

ABGR Clear Value 0 Register

The contents vary depending on use. If you have a fixed point tile buffer, this register contains the R8, G8, B8, A8 color clear value for subsample 0 and MRT0. If you have a floating-point tile buffer, this register contains the red and green components of the 4xFP16 (1:5:10).

The value of the RGBA tile buffer is logically cleared when processing of a new tile starts. If the RGBA tile buffer is not to be cleared, the contents of the RGBA tile buffer can be pre-loaded by using a textured quad primitive.

The four color clear value registers ABGR_CLEAR_VALUE_<0-3> are interpreted in different ways depending on the configuration of the tile buffer and write-back units.

- When normal fixed point tile buffers are used, that is, FP_TILEBUF_ENABLE=0, and Multiple Render Targets (MRT) is off, each of the clear values are interpreted as the clear value for one sub-pixel. To obtain normal pixel clear functionality, set all four color clear registers to the same value.
- When normal fixed point tile buffers and MRT are used, each clear color register effects its renderer target buffer.
- When the floating-point tile buffers are used, that is, when FP_TILEBUF_ENABLE=1, there is only one clear color, that is, the FP16 per component, and that uses the first two registers. The two last registers are ignored. MRT cannot be used together with floating-point tile

buffers.

If you have a fixed point tile buffer, the register bit assignments as below.
(FP_TILEBUF_ENABLE=0)

Bit	Attr	Reset Value	Description
31:24	RW	0x0	Alpha clear value
23:16	RW	0x0	Blue clear value
15:8	RW	0x0	Green clear value
7:0	RW	0x0	Red clear value

If you have a floating-point tile buffer, the register bit assignments as below.
(FP_TILEBUF_ENABLE=1)

Bit	Attr	Reset Value	Description
31:16	RW	0x0	Green clear value
15:0	RW	0x0	Red clear value

ABGR_CLEAR_VALUE_1

Address: Operational Base + offset (0x001C)

ABGR Clear Value 1 Register

If you have a fixed point tile buffer, this register contains the R8, G8, B8, A8 color clear value for subsample 1 and MRT1. If you have a floating-point tile buffer, this register contains the blue and alpha components of the 4xFP16 (1:5:10).

If you have a fixed point tile buffer, the register bit assignments as below.
(FP_TILEBUF_ENABLE=0)

Bit	Attr	Reset Value	Description
31:24	RW	0x0	Alpha clear value
23:16	RW	0x0	Blue clear value
15:8	RW	0x0	Green clear value
7:0	RW	0x0	Red clear value

If you have a floating-point tile buffer, the register bit assignments as below.
(FP_TILEBUF_ENABLE=1)

Bit	Attr	Reset Value	Description
31:16	RW	0x0	Green clear value
15:0	RW	0x0	Red clear value

ABGR_CLEAR_VALUE_2

Address: Operational Base + offset (0x0020)

ABGR Clear Value 2 Register

If you have a fixed point tile buffer, this register contains the R8, G8, B8, A8 color clear value for subsample 2 and MRT2. If you have a floating-point tile buffer, this register is empty.

If you have a fixed point tile buffer, the register bit assignments as below.

(FP_TILEBUF_ENABLE=0)

Bit	Attr	Reset Value	Description
31:24	RW	0x0	Alpha clear value
23:16	RW	0x0	Blue clear value
15:8	RW	0x0	Green clear value
7:0	RW	0x0	Red clear value

If you have a floating-point tile buffer, the register bit assignments as below.

(FP_TILEBUF_ENABLE=1)

Bit	Attr	Reset Value	Description
31:0	-	-	Reserved.

ABGR_CLEAR_VALUE_3

Address: Operational Base + offset (0x0024)

ABGR Clear Value 3 Register

If you have a fixed point tile buffer, this register contains the R8, G8, B8, A8 color clear value for subsample 3 and MRT3. If you have a floating-point tile buffer, this register is empty.

If you have a fixed point tile buffer, the register bit assignments as below.

(FP_TILEBUF_ENABLE=0)

Bit	Attr	Reset Value	Description
31:24	RW	0x0	Alpha clear value
23:16	RW	0x0	Blue clear value
15:8	RW	0x0	Green clear value
7:0	RW	0x0	Red clear value

If you have a floating-point tile buffer, the register bit assignments as below.

(FP_TILEBUF_ENABLE=1)

Bit	Attr	Reset Value	Description
31:0	-	-	Reserved.

BOUNDING_BOX_LEFT_RIGHT

Address: Operational Base + offset (0x0028)

Bounding Box Left Right Register

If write-back is configured to adhere to the BOUNDING box, then when writing back tiles to the framebuffer, write-back can only be achieved within the bounding box area defined in this register.

It is possible to perform non-tile aligned framebuffers. This is achieved by setting up an aligned framebuffer covering the whole final framebuffer and then setting up the bounding box

to match the final framebuffer.

- The top edge of the initial framebuffer must match the top of the final framebuffer because there is no BOUNDING_BOX_TOP.
- The left edge of the initial framebuffer must be within one 16x16 tile of the final framebuffer because the BOUNDING_BOX_LEFT is limited to 15 pixels.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19:16	RW	0x0	BOUNDING_BOX_LEFT; Bits [3:0] of the number of pixels from the left initial framebuffer edge to exclude from write-back, if the bounding box is honored. Bits [13:4] are always 0. If a greater bounding box than 16 is required, the modulo 16 of the bounding box is placed in this register. The remaining part is subtracted from all vertices.
15:14	-	-	Reserved.
13:0	RW	0x0	BOUNDING_BOX_RIGHT; The number of pixels from the left initial framebuffer edge - 1 to include in write-back if the bounding box is honored.

BOUNDING_BOX_BOTTOM

Address: Operational Base + offset (0x002C)

Bounding Box Bottom Register

This register contains the bottom value of the write-out bounding box.

Bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13:0	RW	0x0	BOUNDING_BOX_BOTTOM; The number of pixels from the top initial framebuffer edge, to include in write-back, if the bounding box is honored.

FS_STACK_ADDR

Address: Operational Base + offset (0x0030)

FS Stack Address Register

This register holds the address of the fragment shader.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	FS_STACK_ADDR; Fragment shader stack address.
2:0	-	-	Reserved.

FS_STACK_SIZE_AND_INIT_VAL

Address: Operational Base + offset (0x0034)

FS Stack Address Register

The purpose of this register is to hold the initial value of the fragment shader stack pointer and the stack size.

Bit	Attr	Reset Value	Description
31:16	RW	0x0	<p>FS_STACK_INIT_VAL; The initial value that the stack pointer is set to at the start of executing a fragment shader. This is achieved by setting this field to a non-zero value to enable the main function of a fragment shader to have a stack frame. This register must be set to the size of the largest stack frame of the active fragment shader main function requirements.</p> <p>If the stack pointer is decremented to a value less than FP_STACK_INIT_VAL as a result of a function return, then the fragment shader terminates, and is considered to have executed successfully. This functionality enables the main-function to be recursive, as any other function.</p>
15:0	RW	0x0	<p>FS_STACK_SIZE; The fragment shader stack size in number of 8-byte elements that the fragment shader stack is permitted to contain. The amount of memory that must be allocated for the fragment shader stack is 1KB x FP_STACK_SIZE.</p> <p>Note: The fragment shader requires 128 stacks to keep stack data for all active fragments in the pipeline. An increment of 1 unit in FS_STACK_SIZE means 8-byte elements for each stack for each fragment. This gives 8-byte x 128 stacks/fragment = 1024 bytes.</p>

ORIGIN_OFFSET_X

Address: Operational Base + offset (0x0040)

Origin Offset X Register

The register holds the X offset of the screen-space co-ordinate system. The value in this pixel is the X co-ordinate assigned to the center of the left most pixel of tile (0,0), the X co-ordinate increases to the right. The value is interpreted as a signed half-integer, that is 1 unit = 1/2 pixel. The default value of zero is suitable for Direct3D. For OpenGL, this register must be set to 1.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	ORIGIN_OFFSET_X; X offset of the screen space co-ordinate system.

ORIGIN_OFFSET_Y

Address: Operational Base + offset (0x0044)

Origin Offset Y Register

The value in this pixel is the Y co-ordinate assigned to the center of the uppermost pixel in tile (0, 0). The value is interpreted as a signed half integer, that is, 1 unit = 1/2 pixel. The default value of zero is suitable for Direct3D, for OpenGL, this register must be set to twice the vertical framebuffer resolution minus 1.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	ORIGIN_OFFSET_Y; Y offset of the screen space co-ordinate system.

SUBPIXEL_SPECIFIER

Address: Operational Base + offset (0x0048)

Subpixel Specifier Register

During triangle setup, vertices are rounded to a specific subpixel precision. The SUBPIXEL_SPECIFIER Register selects the level of precision to round to and is set to 127 minus the number of subpixel bits. For example:

- a value of 127 rounds all vertices to a 1x1 grid
- a value of 129 rounds all vertices to a 4x4 grid
- a value of 124 rounds all vertices to a 1/8x1/8 grid

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x75	SUBPIXEL_SPECIFIER; The default value is 117, and is 127 minus the number of subpixel bits. Write to this register as follows: 117 for resolutions below 1024x1024 118 for resolutions between 1024x1024-2048x2048 119 for resolutions above 2048x2048 to guarantee robust rasterization.

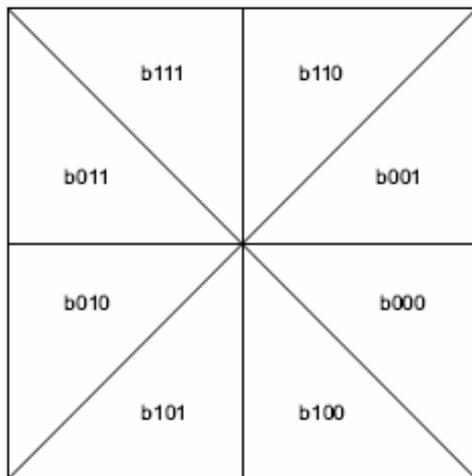
TIEBREAK_MODE

Address: Operational Base + offset (0x004C)

Tiebreak mode Register

The TIEBREAK_MODE Register selects how the rasterizer.

Below figure shows eight polygons sharing the same center vertex. If this vertex is exactly on a sample point, the rasterizer must ensure that only one of the polygons is rasterized at that point. OpenGL only requires the tie breaking rule to be consistent, so any chosen value is compliant. Direct3D specifies a tie-break rule that corresponds to the value b111.



Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x0	TIEBREAK_MODE; Selects how the rasterizer breaks ties when a polygon edge is at a sample point.

WB_x_SOURCE_SELECT (x=0, 1, 2)

Address: Operational Base + offset (0x0100, 0x0200, 0x0300)

WB_x Source Select Register

The WB_x_SOURCE_SELECT Register selects the tile buffer source for the write-back unit. Each register block consists of a unique set of the following registers. In the description, WB_x means either WB0, WB1 or WB2 depending on the base address. In the addresses, the underscore means the differentiating nibble between the WBs, for example 0x0_1C is 0x011C for WB0, 0x021C for WB1, and 0x031C for WB2.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	WB _x _SOURCE_SELECT; Tile buffer source for the write-back unit. 0 = None, WB is disabled 1 = Z/Stencil buffer 2 = ARBG buffer 3 = Reserved.

WB_x_TARGET_ADDR (x=0, 1, 2)

Address: Operational Base + offset (0x0104, 0x0204, 0x0304)

WB_x Target Address Register

The WB_x_TARGET_ADDR Register specifies the start address in memory of the target buffer.

Write-back starts from this address.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	WBx_TARGET_ADDR; The start address in memory of the target buffer.
2:0	-	-	Reserved.

WBx_TARGET_PIXEL_FORMAT (x=0, 1, 2)

Address: Operational Base + offset (0x0108, 0x0208, 0x0308)

WBx Target Pixel Format Register

The WBx_TARGET_PIXEL_FORMAT Register specifies the pixel format of the target buffer. In these formats, component order is common for little-endian image pixels, and different from the order usually seen in shader component arrays. For example ARGB8888 is a 32-bit pixel format with alpha channel in the highest byte in bits [31:24] and the Blue channel in the lowest byte in bits [7:0].

Below table shows the target pixel value and format values:

Value	Format (no flags)	Format (swapRB)	Format (reverse)	Format (reverse and swap)	Pixel bitsize
0	RGB565	BGR565	Reserved	Reserved	16
1	ARGB1555	ABGR1555	BGRA5551	RGBA5551	16
2	ARGB4444	ABGR4444	BGRA4444	RGBA4444	16
3	ARGB8888	ABGR8888	BGRA8888	RGBA8888	32
4	B8	R8	A8	A8	8
5	GB88	GR88	RA88	BA88	16
6	ARGB-FP16	ABGR-FP16	BGRA-FP16	RGBA-FP16	24
7	B-FP16	R-FB16	A-FP16	A-FP16	16
8	GB-FP16	GR-FB16	RA-FP16	BA-FP16	32
9-	Reserved	Reserved	Reserved	Reserved	-
13	S8	S8	S8	S8	8
14	Z16	Z16	Z16	Z16	16
15	S8Z24	S8Z24	S8Z24	S8Z24	32

Bit	Attr	Reset Value	Description

31:4	-	-	Reserved.
3:0	RW	0x0	WBx_TARGET_PIXEL_FORMAT; Contains the pixel format of the target buffer.

WBx_TARGET_AA_FORMAT (x=0, 1, 2)

Address: Operational Base + offset (0x010C, 0x020C, 0x030C)

WBx Target AA Format Register

The WBx_TARGET_AA_FORMAT Register specifies the log-2 number of pixel down-sampling before write-back. This gives an ordered grid super-sampled anti-aliasing effect in addition to the internal per-primitive 4x anti-aliasing.

Below table shows supported pixel formats. There are restrictions on which combinations of WBx_TARGET_AA_FORMAT and WBx_TARGET_PIXEL_FORMAT are supported. Therefore you cannot use all WBx_TARGET_PIXEL_FORMATS with all down-sampling values. The pixel bit size is the limiting factor:

- (X) marks pixel bit sizes supported in all WBx_TARGET_LAYOUTS
- marks pixel bit sizes only supported in interleave WBx_TARGET_LAYOUT.

Value	Effective tile buffer size	SSAA level	Supported Pixel Formats			
			32 bit	16 bit	8 bit	FP
0	16 x 16	0x	X	X	X	X
1	16 x 8	2x	X	X	X	-
2	8 x 8	4x	X	X	X	-
3	8 x 4	8x	X	X	X	-
4	4 x 4	16x	X	X	i	-
5	4 x 2	32x	X	X	i	-
6	2 x 2	64x	X	i	-	-
7	2 x 1	128x	X	-	-	-

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x0	WBx_TARGET_AA_FORMAT; Format table.

WBx_TARGET_LAYOUT (x=0, 1, 2)

Address: Operational Base + offset (0x0110, 0x0210, 0x0310)

WBx Target Layout Register

The WBx_TARGET_LAYOUT Register specifies the pixel layout of the target buffer.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	WBx_TARGET_LAYOUT;

			<p>0 Linear layout. The pixels are stored in normal linear layout in memory.</p> <p>1 Interleaved layout. This is a fully interleaved mode where pixels are stored in u-order in memory for best possible 2D locality. This normally requires a quadratic frame buffer with power of two sides, but can also be used if the width is twice the height and both sides are powers of two.</p> <p>2 Interleaved blocks. Each 16x16 pixel block is interleaved u-order internally and then the blocks are stored linearly in the frame buffer.</p> <p>3 Reserved.</p>
--	--	--	---

WBx_TARGET_SCANLINE_LENGTH (x=0, 1, 2)

Address: Operational Base + offset (0x0114, 0x0214, 0x0314)

WBx Target Scanline Length Register

The WBx_TARGET_SCANLINE_LENGTH specifies the offset between the beginning of two lines of the target buffer. The actual meaning depends on the value of WBx_TARGET_LAYOUT.

Below table shows the pixel layout of the target buffer:

WBx_TARGET_SCANLINE_LENGTH	Interpretation of WBx_TARGET_SCANLINE_LENGTH value
Linear	The number of bytes difference between two consecutive pixel lines divided by eight.
Interleaved	No meaning. Ignored.
Interleaved Blocks	The number of blocks between two consecutive block lines.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	WBx_TARGET_SCANLINE_LENGTH; Specifies the offset between the beginning of two lines of the target buffer. The actual meaning depends on the value of WBx_TARGET_LAYOUT.

WBx_TARGET_FLAGS (x=0, 1, 2)

Address: Operational Base + offset (0x0118, 0x0218, 0x0318)

WBx Target Flags Register

Bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5	RW	0x0	WBx_BIG_ENDIAN; When enabled, pixels are written in big-endian byte order. When disabled, pixels are written in little-endian byte order.
4	RW	0x0	WBx_DITHER_ENABLE; When enabled, dithering of the write-back data is performed using Bayer ordered dithering. Dithering is only possible when FP_TILEBUF_ENABLE is off.
3	RW	0x0	WBx_INV_COMPONENT_ORDER_ENABLE; When enabled, color formats get the order of their components inverted, for example: RGBA -> ARGB. This, together with WBx_SWAP_RED_BLUE_ENABLE makes it possible to support different component permutations of WBx_TARGET_PIXEL_FORMAT.
2	RW	0x0	WBx_SWAP_RED_BLUE_ENABLE; When enabled, color formats get their red and blue components swapped, for example: RGBA->BGRA. This, together with WBx_INV_COMPONENT_ORDER_ENABLE makes it possible to support different component permutations of WBx_TARGET_PIXEL_FORMAT.
1	RW	0x0	WBx_BOUNDING_BOX_ENABLE; When enabled, write-back is limited to inside the rectangular box defined by the BOUNDING_BOX_LEFT_RIGHT and BOUNDING_BOX_BOTTOM registers.
0	RW	0x0	WBx_DIRTY_BIT_ENABLE; When enabled, only pixels written to in the tile buffer are written back to the framebuffer.

WBx_MRT_ENABLE (x=0, 1, 2)

Address: Operational Base + offset (0x011C, 0x021C, 0x031C)

WBx MRT Enable Register

Defining multiple render targets enables the pipeline to output up to four different result buffers. This is implemented during rendering by re-using the four sub-pixels as separate result pixels.

This means that MRT is not compatible with internal anti-aliasing or floating-point tile buffer, and that sub-pixel masking techniques must be applied to get different results in the different sub-pixels during rendering.

The enable value indicates how many of the sub-pixels are to be written back to separate buffers, starting with sub-pixel 0.

When the enable value is zero, MRT is disabled and the output per pixel is the average of the four sub-pixels in the tile buffer, that is, normal rendering.

Bit	Attr	Reset Value	Description										
31:4	-	-	Reserved.										
3:0	RW	0x0	<p>WBx_MRT_ENABLE; bit for each MRT target:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Interpretation</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>MRT 0 enabled.</td> </tr> <tr> <td>1</td> <td>MRT 1 enabled.</td> </tr> <tr> <td>2</td> <td>MRT 2 enabled.</td> </tr> <tr> <td>3</td> <td>MRT 3 enabled.</td> </tr> </tbody> </table>	Bit	Interpretation	0	MRT 0 enabled.	1	MRT 1 enabled.	2	MRT 2 enabled.	3	MRT 3 enabled.
Bit	Interpretation												
0	MRT 0 enabled.												
1	MRT 1 enabled.												
2	MRT 2 enabled.												
3	MRT 3 enabled.												

WBx_MRT_OFFSET (x=0, 1, 2)

Address: Operational Base + offset (0x0120, 0x0220, 0x0320)

WBx MRT Offset Register

This offset value contained in the WBx_MRT_OFFSET Register defines the distance in memory, 8-byte aligned, between each MRT.

MRT0 is output to WBx_TARGET_ADDR, and MRT1 is output to WBx_TARGET_ADDR+WBx_MRT_OFFSET. This means all MRT buffers for a single WB must be allocated with equal spacing in memory.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	WBx_MRT_OFFSET; Offset value giving the distance in memory between each MRT.
2:0	-	-	Reserved.

WBx_GLOBAL_TEST_ENABLE (x=0, 1, 2)

Address: Operational Base + offset (0x0124, 0x0224, 0x0324)

WBx Global Test Enable Register

The WBx_GLOBAL_TEST_ENABLE Register contains the global testing enable bit. The global test enables testing the data in the tile buffer against a reference value using a compare

function. Data that passes the test is written back to the buffer. Data that fails is not written back.

Below table shows the write-back source select, FB tile buffer enable and global test data selection.

WBx_SOURCE_SELECT	FB_TILEBUF_ENABLE	Global test data
1 - Z/Stencil	Don't care	Stencil 8-bit
2 - ARGB Color	0	Alpha 8-bit
3 - ARGB Color	1	Alpha FP16

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	WBx_GLOBAL_TEST_ENABLE; Set to one to enable global write-back value testing.

WBx_GLOBAL_TEST_REF_VALUE (x=0, 1, 2)

Address: Operational Base + offset (0x0128, 0x0228, 0x0328)

WBx Global Test Reference Value Register

The WBx_GLOBAL_TEST_REF_VALUE Register provides the reference value to compare the tile buffer data against. Interpretation of the value depends on the WBx_TARGET_SELECT and FP_TILEBUF_ENABLE selection.

Below table shows write-back source select, FB tile buffer enable, and global test data selection.

WBx_SOURCE_SELECT	FB_TILEBUF_ENABLE	Global test data
1 - Z/Stencil	Don't care	Stencil 8-bit
2 - ARGB Color	0	Alpha 8-bit
3 - ARGB Color	1	Alpha FP16

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	WBx_GLOBAL_TEST_REF_VALUE; See above table for interpretation.

WBx_GLOBAL_TEST_CMP_FUNCT (x=0, 1, 2)

Address: Operational Base + offset (0x012C, 0x022C, 0x032C)

WBx Global Test Compare Function Register

The WBx_GLOBAL_TEST_CMP_FUNCT Register provides the compare function to use on the tile data and the reference value.

Below table shows meaning of the compare function bits. Several bits can be enabled

imultaneously. For example, b011 means less than or equal.

WBx_SOURCE_SELECT	FB_TILEBUF_ENABLE	Value interpretation
1 - Z/Stencil	Don't care	Stencil 8-bit
2 - ARGB Color	0	Alpha 8-bit
3 - ARGB Color	1	Alpha FP16

The function is applied in the following way:

pass = tile_value <cmp_func> ref_value

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x0	WBx_GLOBAL_TEST_CMP_FUNC; See above table for interpretation.

CURRENT_REND_LIST_ADDR

Address: Operational Base + offset (0x1004)

Current Renderer List Address Register at 0x1004

The CURRENT_REND_LIST_ADDR Register displays the address of the current polygon list item being processed. Processing a list can be resumed after an END_AFTER_TILE command has been issued, by writing this address to the REND_LIST_ADDR register.

Bit	Attr	Reset Value	Description
31:5	RW	0x0	CURRENT_REND_LIST_ADDR; Address of the current polygon list item being processed.
4:0	-	-	Reserved.

STATUS

Address: Operational Base + offset (0x1008)

Status Register

The STATUS Register is to contain the status of the processor.

Bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0x0	INTERRUPT_ASSERTED; Shows the current status of the interrupt request line of the processor.
5	RW	0x0	WRITE_BOUNDARY_ERROR; WRITE_BOUNDARY_ERROR Show that the processor attempted to write outside the write boundary set by the WRITE_BOUNDARY registers.
4	RW	0x0	BUS_STOPPED; Shows that the master bus interface of the processor has been stopped because of a STOP_BUS command

			or a performance counter limit event. The bus interface can be restarted by using the START_BUS command.
3	RW	0x0	BUS_ERROR; A bus transaction has ended with error. The processor has been stopped and has to be reset before rendering can be started again
2	RW	0x0	HANG; An invalid state of the processor has been detected. The processor has been stopped and has to be reset before rendering can be started again
1	RW	0x0	TILE_STOPPED; Rendering of the current tile has been completed as if it was the last tile of the frame. Indicates that an END_AFTER_TILE command has been issued.
0	RW	0x0	RENDERING_ACTIVE; The processor is currently active rendering.

CTRL_MGMT

Address: Operational Base + offset (0x100C)

Control Management Register

The CTRL_MGMT Register passes control signals to the pixel processor before, after, and during rendering.

Bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	W	-	START_RENDERING; Writing to this bit initiates rendering. Do not write this value during rendering.
5	W	-	FORCE_RESET; Writing to this bit resets the pixel processor, so that it can be brought out of a hang in a reasonably clean manner.
4	W	-	FORCE_HANG; Writing to this bit causes the pixel processor to hang. Only useful for debugging.
3	W	-	FLUSH_CACHES; Writing to this bit causes all the vertex, RSW, texture and palette caches to be flushed immediately. This must be done only when the renderer is idle, otherwise the hardware cannot guarantee that caches become clean or that renderer glitches do not occur. The processor must have an active clock for the flush to have an effect. The processor might have the clock shut off when idle to conserve power, depending on the processor integration. This means that the

			<p>FLUSH_CACHES command must be issued in one of the following states of operation:</p> <ol style="list-style-type: none"> 1. At the beginning of a frame, after the APB registers have been written to, but before the START_RENDERING command has been issued. 2. At the end of a frame, after the interrupt has been received but before the interrupt signal has been masked or acknowledged. <p>If neither is possible, you can use a FORCE_RESET command to flush the caches. FLUSH_CACHES is performed implicitly every time the pixel processor starts rendering, so explicitly using this bit is rarely required.</p>
2	W	-	<p>END_AFTER_TILE;</p> <p>Writing to this bit causes the renderer to treat a BEGIN-NEW-TILE command, that is, Cmd 14, as an End-Of-List command, that is Cmd 15. This action finalizes the rendering of the current tile, leaving the framebuffer incomplete in most situations, except when the current tile is the last tile of the frame. Rendering of the rest of the frame can be initiated by writing the CURRENT_RENDER_LIST_ADDR value to the REND_LIST_ADDR register and issuing the START_RENDERING command.</p>
1	W	-	<p>START_BUS;</p> <p>Writing to this bit reactivates the bus interface after it has been stopped by a STOP_BUS command or a WRITE_BOUNDARY_LIMIT event. The effect of issuing a START_BUS and a STOP_BUS command at the same time is not defined.</p>
0	W	-	<p>STOP_BUS;</p> <p>Writing to this bit causes the bus interface to hold back future transactions on the bus.</p> <p>Any current bus transactions are completed before the interface is stopped. The bus can be restarted by issuing a START_BUS command. The effect of issuing a START_BUS and a STOP_BUS command at the same time are not defined.</p>

INT_RAWSTAT

Address: Operational Base + offset (0x1020)

Interrupt Rawstat Register

The INT_RAWSTAT Register shows the unmasked status of the interrupt sources. Writing a 1 to the bit of an interrupt source forces this bit to be set and generate an interrupt if it is not masked. Writing a 0 to the bit of an interrupt source has no effect. Use the INT_CLEAR Register to clear interrupts.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	RW	0x0	WRITE_BOUNDARY_ERROR; The processor has attempted to write outside the write boundary set by the WRITE_BOUNDARY registers. The bus interface completes the previous bus transaction but does not initialize any new transactions before the WRITE_BOUNDARY is modified or disabled, or the renderer is reset.
7	RW	0x0	CNT_1_LIMIT; Performance counter PERF_CNT_1 has passed the limit set in PERF_CNT_1_LIMIT. The bus interface is stopped as with STOP_BUS, and this interrupt source set. The BUS_STOP interrupt is asserted when the bus is actually stopped. This is likely to occur after CNT_1_LIMIT has been triggered.
6	RW	0x0	CNT_0_LIMIT; Performance counter PERF_CNT_0 has passed the limit set in PERF_CNT_0_LIMIT. The bus interface is stopped as with STOP_BUS, and this interrupt source set. The BUS_STOP interrupt is asserted when the bus is actually stopped. This is likely to occur after CNT_0_LIMIT has been triggered.
5	RW	0x0	BUS_STOP; The renderer has been stopped by a STOP_BUS command. BUS_STOP is triggered only after the bus is actually stopped, making it likely that BUS_STOP is delayed a bit after CNT_x_LIMIT has been triggered. Operation can be continued by issuing the START_BUS command.
4	RW	0x0	BUS_ERROR; A bus transaction has ended with error. The processor has been stopped and has to be reset before rendering

			can be started again.
3	RW	0x0	FORCE_HANG; The processor has been forced into an illegal state by the FORCE_HANG command. The renderer must be reset before rendering can be started again.
2	RW	0x0	HANG; The processor has entered into an illegal state without force. The renderer must be reset before rendering can be started again.
1	RW	0x0	END_OF_TILE; Rendering has been ended by an END_AFTER_TILE command. The frame buffer might be incomplete.
0	RW	0x0	END_OF_FRAME; Rendering has ended by completion of the polygon list. The framebuffer is complete.

INT_CLEAR

Address: Operational Base + offset (0x1024)

Interrupt Clear Register

The INT_CLEAR Register clears interrupt sources.

Writing a 1 to the bit of an asserted source clears the interrupt in INT_RAWSTAT and in INT_STATUS, if it is not masked.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	W	-	WRITE_BOUNDARY_ERROR; Interrupt clear if write a 1.
7	W	-	CNT_1_LIMIT; Interrupt clear if write a 1.
6	W	-	CNT_0_LIMIT; Interrupt clear if write a 1.
5	W	-	BUS_STOP; Interrupt clear if write a 1.
4	W	-	BUS_ERROR; Interrupt clear if write a 1.
3	W	-	FORCE_HANG; Interrupt clear if write a 1.
2	W	-	HANG; Interrupt clear if write a 1.
1	W	-	END_OF_TILE; Interrupt clear if write a 1.
0	W	-	END_OF_FRAME; Interrupt clear if write a 1.

INT_MASK

Address: Operational Base + offset (0x1028)

Interrupt Mask Register

The INT_MASK Register holds the bit mask that enables an interrupt source if the corresponding mask bit is set to 1.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	RW	0x1	WRITE_BOUNDARY_ERROR; Enables an interrupt if set to 1.
7	RW	0x1	CNT_1_LIMIT; Enables an interrupt if set to 1.
6	RW	0x1	CNT_0_LIMIT; Enables an interrupt if set to 1.
5	RW	0x1	BUS_STOP; Enables an interrupt if set to 1.
4	RW	0x1	BUS_ERROR; Enables an interrupt if set to 1.
3	RW	0x1	FORCE_HANG; Enables an interrupt if set to 1.
2	RW	0x1	HANG; Enables an interrupt if set to 1.
1	RW	0x1	END_OF_TILE; Enables an interrupt if set to 1.
0	RW	0x1	END_OF_FRAME; Enables an interrupt if set to 1.

INT_STATUS

Address: Operational Base + offset (0x102C)

Interrupt Status Register at 0x102C

The INT_STATUS Register is raw status ANDed with the interrupt mask and shows the active and masked interrupt sources. Only the status of register bits selected by the Interrupt Mask are shown. If any of the sources are asserted in the INT_STATUS, then the processor **IRQ** line is asserted.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	R	0x0	WRITE_BOUNDARY_ERROR; Interrupt happen if set to 1.
7	R	0x0	CNT_1_LIMIT; Interrupt happen if set to 1.
6	R	0x0	CNT_0_LIMIT; Interrupt happen if set to 1.
5	R	0x0	BUS_STOP;

			Interrupt happen if set to 1.
4	R	0x0	BUS_ERROR; Interrupt happen if set to 1.
3	R	0x0	FORCE_HANG; Interrupt happen if set to 1.
2	R	0x0	HANG; Interrupt happen if set to 1.
1	R	0x0	END_OF_TILE; Interrupt happen if set to 1.
0	R	0x0	END_OF_FRAME; Interrupt happen if set to 1.

WRITE_BOUNDARY_ENABLE

Address: Operational Base + offset (0x1040)

Write Boundary Enable Register

The WRITE_BOUNDARY_ENABLE Register enables the write boundary safety function and asserts the WRITE_BOUNDARY_ERROR interrupt source.

When this bit is set to 1, the renderer is not able to write to addresses outside the boundaries set by the

WRITE_BOUNDARY_LOW and WRITE_BOUNDARY_HIGH bus address values. Attempts to write outside the write boundary halts the bus interface and asserts the WRITE_BOUNDARY_ERROR interrupt source.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	WRITE_BOUNDARY_ENABLE; When set to 1, the renderer is not able to write to addresses outside the boundaries set by the WRITE_BOUNDARY_LOW and WRITE_BOUNDARY_HIGH bus address values.

WRITE_BOUNDARY_LOW

Address: Operational Base + offset (0x1044)

Write Boundary Low Register

The WRITE_BOUNDARY_LOW Register sets the low write boundary for the renderer. When WRITE_BOUNDARY_ENABLE is set to 1, the renderer is not permitted to write below this address. The boundary can be set with a 256 byte resolution.

Bit	Attr	Reset Value	Description
31:8	RW	0x0	WRITE_BOUNDARY_LOW; Address for setting the low write boundary for the renderer.
7:0	-	-	Reserved.

WRITE_BOUNDARY_HIGH

Address: Operational Base + offset (0x1048)

Write Boundary High Register

The WRITE_BOUNDARY_LOW Register sets the high write boundary for the renderer. When WRITE_BOUNDARY_ENABLE is set to 1, the renderer is not permitted to write below this address. The boundary can be set with a 256 byte resolution.

Bit	Attr	Reset Value	Description
31:8	RW	0x0	WRITE_BOUNDARY_HIGH; Address for setting the high write boundary for the renderer.
7:0	-	-	Reserved.

WRITE_BOUNDARY_ADDRESS

Address: Operational Base + offset (0x104C)

Write Boundary Address Register

The WRITE_BOUNDARY_ADDRESS Register is read-only. After a WRITE_BOUNDARY_ERROR the write address is stored in the WRITE_BOUNDARY_ADDRESS

Bit	Attr	Reset Value	Description
31:2	R	0x0	WRITE_BOUNDARY_ADDRESS; After a WRITE_BOUNDARY_ERROR the write address is stored in this location.
1:0	-	-	Reserved.

BUS_ERROR_STATUS

Address: Operational Base + offset (0x1050)

Bus Error Status Register

The WRITE_BOUNDARY_ADDRESS Register is read-only. After a WRITE_BOUNDARY_ERROR the write address is stored in the WRITE_BOUNDARY_ADDRESS

Below table shows the type of memory access that corresponds to each error ID.

ID	Access Type
0	Reserved.
1	Tile write-back.
2	Load/store unit writes.
3	Reserved.
4	Palette reads.
5	Texture reads.
6	Polygon list reads.
7	RSW reads.

8	Vertex reads.
9	Uniform remap reads.
10	Shader program reads.
11	Varying reads.
12	Texture descriptor reads.
13	Texture descriptor remapping reads.
14	Compressed texture reads.
15	Load/store unit reads. Store instructions might cause a bus read because of caching effects.

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved.
9:6	R	0x0	Read Error ID; Id number of read error.
5:2	R	0x0	Write Error ID; Id number of write error.
1	R	0x0	Read Error; Indicates a read error.
0	R	0x0	Write Error; Indicates a write error.

WATCHDOG_DISABLE

Address: Operational Base + offset (0x1060)

Watchdog Disable Register

The processor includes a watchdog timer that can detect hang situations. This feature is enabled by default, and counts clock cycles and asserts a HANG interrupt when the WATCHDOG_TIMEOUT number of cycles is reached without activity from selected internal units.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	WATCHDOG_DISABLE; Disables watchdog timer

WATCHDOG_TIMEOUT

Address: Operational Base + offset (0x1064)

Watchdog Timeout Register

The default watchdog time is 1000000 cycles. The reset value is 0x000F4240. Valid values are in the range 0-0xFFFFFE. The watchdog timer is triggered when the number of cycles that have passed without activity in selected internal modules exceeds WATCHDOG_TIMEOUT. A 24-bit counter can never exceed 0xFFFFFFF, so this value cannot trigger the watchdog.

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.

23:0	RW	0xF4240	WATCHDOG_TIMEOUT; Watchdog value
------	----	---------	-------------------------------------

PERF_CNT_0_ENABLE

Address: Operational Base + offset (0x1080)

Performance Counter 0 Enable Register

The PERF_CNT_0_ENABLE Register enables performance counter 0. This processor has two performance counters that you can use to monitor performance.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	RW	0x0	PERF_CNT_0_LIM_EN; When set to 1, the PERF_CNT_0_LIMIT Register becomes active. If the PERF_CNT_0_VALUE Register exceeds the Performance Counter 0 Limit value, then an interrupt is asserted and the BUS_STOPPED mechanism stops the bus. The PERF_CNT_0_VALUE Register is reset to zero if you write to PERF_CNT_0_ENABLE while the counter is enabled, that is, the PERF_CNT_0_ENABLE bit is 1.
0	RW	0x0	PERF_CNT_0_ENABLE; When set to 1, the performance counter 0 is reset to zero and activated. The PERF_CNT_0_SRC Register selects the event to be counted during a frame.

PERF_CNT_0_SRC

Address: Operational Base + offset (0x1084)

Performance Counter 0 SRC Register

The PERF_CNT_0_SRC Register selects the event to count if counter 0 is enabled. Writing to this register while PERF_CNT_0_ENABLE is set resets the counter to 0.

Bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5:0	RW	0x0	PERF_CNT_0_SRC; Index value. See below table:

Index	Counter	Description
0	Active clock cycles count	Active clock cycles, between Polygon start and IRQ.
1	Total clock cycles count	Number of clock cycles from counter was set.
2	Total bus reads	Total number of 64-bit words read from the bus.

3	Total bus writes	Total number of 64-bit words written to the bus.
4	Bus read request cycles count	Number of cycles of bus read request signal HIGH.
5	Bus write request cycles count	Number of cycles of bus write request signal HIGH.
6	Bus read transactions count	Number of read requests accepted by the bus.
7	Bus write transactions count	Number of write requests accepted by the bus.
8	Reserved	
9	Tile write-back writes	Number of 64-bit words written to the bus.
10	Store unit writes	Number of 64-bit words written to the bus.
11	Reserved.	
12	Palette cache reads	Number of 64-bit words read from the bus.
13	Texture cache uncompressed reads	Number of 64-bit words read from the bus.
14	Polygon list reads	Number of 64-bit words read from the bus.
15	RSW reads	Number of 64-bit words read from the bus.
16	Vertex cache reads	Number of 64-bit words read from the bus.
17	Uniform remapping reads	Number of 64-bit words read from the bus.
18	Program cache reads	Number of 64-bit words read from the bus.
19	Varying reads	Number of 64-bit words read from the bus.
20	Texture descriptors reads	Number of 64-bit words read from the bus.
21	Texture descriptor remapping reads	Number of 64-bit words read from the bus.
22	Texture cache compressed reads	Number of 64-bit words read from the bus.
23	Load unit reads	Number of 64-bit words read from the bus.
24	Polygon count	Number of triangles read from the polygon list.
25	Pixel rectangle count	Number of pixel rectangles read from the polygon list.

26	Lines count	Number of lines read from the polygon list.
27	Points count	Number of points read from the polygon list.
28	Stall cycles PolygonListReader	Number of clock cycles Polygon list reader waits for output being collected.
29	Stall cycles triangle setup	Number of clock cycles TSC waits for input.
30	Quad rasterized count	Number of 22 quads output from rasterizer.
31	Fragment rasterized count	Number of fragment rasterized. Fragments/(Quads x 4) gives average actual fragments per quad.
32	Fragments rejected fragment-kill count	Number of fragments exiting the fragment shader as killed.
33	Fragments rejected fwd-fragment-kill count	Number of fragments killed by forward fragment kill.
34	Fragments passed z-stencil count	Number of fragments passing Z and stencil test.
35	Patches rejected early z/stencil count	Number of patches rejected by EarlyZ. A patch can be 8x8, 4x4 or 2x2 fragments.
36	Patches evaluated	Number of patches evaluated for EarlyZ rejection.
37	Instruction completed count	Number of fragment shader instruction words completed. It is a function of fragments processed and the length of the shader programs.
38	Instruction failed rendezvous count	Number of fragment shader instructions not completed because of failed Rendezvous.
39	Instruction failed varying-miss count	Number of fragment shader instructions not completed because of failed varying operation.
40	Instruction failed texture-miss count	Number of fragment shader instructions not completed because of failed texture operation.
41	Instruction failed load-miss count	Number of fragment shader instructions not completed because of failed load operation.
42	Instruction failed tile read-miss count	Number of fragment shader instructions not completed because

		of failed read from the tilebuffer.
43	Instruction failed store-miss count	Number of fragment shader instructions not completed because of failed store operation.
44	Rendezvous breakage count	Number of Rendezvous breakages reported.
45	Pipeline bubbles cycle count	Number of unused cycles in the fragment shader while rendering is active.
46	Texture mapper multipass count	Number of texture operations looped because more texture passes are required.
47	Texture mapper cycle count	Number of texture operation cycles.
48	Vertex cache hit count	Number of hits in the vertex cache. One for each vertex.
49	Vertex cache miss count	Number of misses in the vertex cache. One for each vertex.
50	Varying cache hit count	Number of hits in the varying cache. One for each varying.
51	Varying cache miss count	Number of normal misses in the varying cache. One for each varying.
52	Varying cache conflict miss count	Number of conflict misses in the varying cache. It happens when an access pattern cannot be serviced by the cache. One for each varying.
53	Texture cache hit count	Number of hits in the texture cache. One for each texel.
54	Texture cache miss count	Number of misses in the texture cache. One for each texel.
55	Texture cache conflict miss count	Number of conflict misses in the texture cache. It happens when an access pattern cannot be serviced by the cache. One for each texel.
56	Palette cache hit count	Number of hits in the palette cache.
57	Palette cache miss count	Number of misses in the palette cache.
58	Load/Store cache hit count	Number of hits in the load/store cache.
59	Load/Store cache miss count	Number of misses in the load/store cache.
60	Program cache cache hit count	Number of hits in the program cache.

61	Program cache cache miss count	Number of misses in the program cache.
62	Reserved	-
63	No event	-

PERF_CNT_0_LIMIT

Address: Operational Base + offset (0x1088)

Performance Counter 0 Limit Register

The PERF_CNT_0_LIMIT Register holds the limit for performance counter 0. If `PERF_CNT_0_VALUE > PERF_CNT_0_LIMIT` and `PERF_CNT_0_LIM_EN == 1` then the bus is stopped and the `CNT_0_LIM` interrupt source is asserted.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PERF_CNT_0_LIMIT; Holds the limit for performance counter 0.

PERF_CNT_0_VALUE

Address: Operational Base + offset (0x108C)

Performance Counter 0 Value Register

The PERF_CNT_0_VALUE Register holds the current value of the performance counter 0. Writing anything to this register resets the counter to 0.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PERF_CNT_0_VALUE; Holds the current value of the performance counter 0.

PERF_CNT_1_ENABLE

Address: Operational Base + offset (0x10A0)

Performance Counter 1 Enable Register

The PERF_CNT_1_ENABLE Register enables performance counter 1. This processor has two performance counters that you can use to monitor performance.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	RW	0x0	PERF_CNT_1_LIM_EN; When set to 1, the <code>PERF_CNT_1_LIMIT</code> Register becomes active. If the <code>PERF_CNT_1_VALUE</code> Register exceeds the Performance Counter 1 limit value, then an interrupt is asserted and the <code>BUS_STOPPED</code> mechanism stops the bus. The <code>PERF_CNT_1_VALUE</code> Register is reset to zero if you write to <code>PERF_CNT_1_ENABLE</code> while the counter is enabled, that is, the <code>PERF_CNT_1_ENABLE</code> bit is 1.

0	RW	0x0	PERF_CNT_1_ENABLE; When set to 1, the performance counter 1 is reset to zero and activated. The PERF_CNT_1_SRC Register selects the event to be counted during a frame.
---	----	-----	--

PERF_CNT_1_SRC

Address: Operational Base + offset (0x10A4)

Performance Counter 1 SRC Register

The PERF_CNT_1_SRC Register selects the event to count if counter 1 is enabled. Writing to this register while PERF_CNT_1_ENABLE is set resets the counter to 1.

Bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5:0	RW	0x0	PERF_CNT_1_SRC; Index value The same with "PERF_CNT_0_SRC" table.

PERF_CNT_1_LIMIT

Address: Operational Base + offset (0x10A8)

Performance Counter 1 Limit Register

The PERF_CNT_1_LIMIT Register holds the limit for performance counter 10. If PERF_CNT_1_VALUE > PERF_CNT_1_LIMIT and PERF_CNT_1_LIM_EN== 1 then the bus is stopped and the CNT_1_LIM interrupt source is asserted.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PERF_CNT_1_LIMIT; Holds the limit for performance counter 1.

PERF_CNT_1_VALUE

Address: Operational Base + offset (0x10AC)

Performance Counter 1 Value Register

The PERF_CNT_1_VALUE Register holds the current value of the performance counter 1. Writing anything to this register resets the counter to 0.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	PERF_CNT_1_VALUE; Holds the current value of the performance counter 1.

Geometry Processor Registers**GP_CONTR_REG_VSCL_START_ADDR**

Address: Operational Base + offset (0x0000)

GP Control Register VSCL Start Address

Reading from the GP_CONTR_REG_VSCL_START_ADDR Register at 0x0000 returns the

current command list item being processed. You can read the actual register value from register 0x0080.

Writing to the GP_CONTR_REG_VSCL_START_ADDR Register sets the start address of the vertex shader command list. The value is not visible until the vertex shader is started. This start address is 8-byte aligned.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	GP_CONTR_REG_VSCL_END_ADDR; End address of the VSCL.
2:0	-	-	Reserved.

GP_CONTR_REG_VSCL_END_ADDR

Address: Operational Base + offset (0x0004)

GP Control Register VSCL End Address

The GP_CONTR_REG_VSCL_END_ADDR Register provides the end address of the VSCL. When the VSCL execution reaches the end address, the command list processing terminates, and the GP_IRQ_VS_END_CMD_LIST interrupt is asserted. The command at this address is not executed. This address is 8-byte aligned.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	GP_CONTR_REG_VSCL_START_ADDR; Start address of the VSCL.
2:0	-	-	Reserved.

GP_CONTR_REG_PLBCL_START_ADDR

Address: Operational Base + offset (0x0008)

GP Control Register PLBCL Start Address

Reading from GP_CONTR_REG_PLBCL_START_ADDR at 0x0008 returns the current command list item being processed. You can read the actual register value from register 0x0084.

Writing to the GP_CONTR_REG_PLBCL_START_ADDR Register sets the start address of the PLB command list. The value is not visible until the vertex shader is started. This address is 8-byte aligned.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	GP_CONTR_REG_PLBCL_START_ADDR; Start address of the PLBCL.
2:0	-	-	Reserved.

GP_CONTR_REG_PLBCL_END_ADDR

Address: Operational Base + offset (0x000C)

GP Control Register PLBCL End Address

The GP_CONTR_REG_PLBCL_END_ADDR Register provides the end address of the PLBCL. When the PLBCL execution reaches the end address, the command list processing

terminates, and the GP_IRQ_PLB_END_CMD_LIST interrupt is asserted. The command at this address is not executed. This address is 8-byte aligned.

Bit	Attr	Reset Value	Description
31:3	RW	0x0	GP CONTR_REG_PLBCL_END_ADDR; End address of PLB command list.
2:0	-	-	Reserved.

GP CONTR_REG_PLB_ALLOC_START_ADDR

Address: Operational Base + offset (0x0010)

GP Control Register PLB Allocate Start Address

Writing to the GP CONTR_REG_PLB_ALLOC_START_ADDR Register sets the start address for polygon list allocation. The register does not show the new value until the GP_CMD_UPDATE_PLB_ALLOC command is given through the GP CONTR_REG_CMD Register. Reading from this register returns the current address for polygon list allocation.

Bit	Attr	Reset Value	Description
31:7	RW	0x0	GP CONTR_REG_PLB_ALLOC_START_ADDR Start/current address for the polygon list allocation.
6:0	-	-	Reserved.

GP CONTR_REG_PLB_ALLOC_END_ADDR

Address: Operational Base + offset (0x0014)

GP Control Register PLB Allocate End Address

The GP CONTR_REG_PLB_ALLOC_END_ADDR Register provides the end address for polygon list allocation. If the current address for polygon list allocation reaches this address, the PLB is stalled and the GP_IRQ_PLB_OUT_OF_MEM interrupt is asserted. The register does not show the new value until the GP_CMD_UPDATE_PLB_ALLOC command is given through the GP CONTR_REG_CMD Register.

Bit	Attr	Reset Value	Description
31:7	RW	0x0	GP CONTR_REG_PLB_ALLOC_END_ADDR; End address for the polygon list allocation
6:0	-	-	Reserved.

GP CONTR_REG_CM

Address: Operational Base + offset (0x0020)

GP Control Register Command

The GP CONTR_REG_CMD Register issues commands to start and stop functional units.

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved.
9	W	-	GP_CMD_STOP_BUS; Stop data bus.
8	W	-	GP_CMD_START_BUS;

			Start data bus.
7	-	-	Reserved.
6	W	-	GP_CMD_FORCE_HANG; Forced hang. This functionality depends on the watchdog timer. See Watchdog Disable Register
5	W	-	Force reset.
4	W	-	Update polygon list allocation addresses.
3:2	-	-	Reserved.
1	W	-	GP_CMD_START_PLB; Start PLB.
0	W	-	GP_CMD_START_VS; Start vertex shader.

GP_CONTR_REG_INT_RAWSTAT

Address: Operational Base + offset (0x0024)

GP Control Register Interrupt Rawstat

The GP_CONTR_REG_INT_RAWSTAT Register normally shows the unmasked status of the interrupt sources. Any bit written as 1 is forced on, and generates an interrupt if not masked. Any bit written as 0 is ignored and has no effect.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	RW	0x0	GP_IRQ_AXI_BUS_ERROR; AXI bus error.
10	-	-	Reserved.
9	RW	0x0	GP_IRQ_WRITE_BOUND_ERR; Write boundaries error.
8	RW	0x0	GP_IRQ_PERF_CNT_1_LIMIT; Performance counter 1 limit reached.
7	RW	0x0	GP_IRQ_PERF_CNT_0_LIMIT; Performance counter 0 limit reached.
6	RW	0x0	GP_IRQ_FORCED_HANG; Forced hang.
5	RW	0x0	GP_IRQ_HANG; Hang.
4	RW	0x0	GP_IRQ_PLB_SEM; PLB semaphore decremented.
3	RW	0x0	GP_IRQ_VS_SEM; Vertex shader semaphore incremented.
2	RW	0x0	GP_IRQ_PLB_OUT_OF_MEM; PLB out of list memory.
1	RW	0x0	GP_IRQ_PLB_END_CMD_LIST; PLB end of command list.

0	RW	0x0	GP_IRQ_VS_END_CMD_LIST; Vertex shader end of command list.
---	----	-----	---

GP_CONTR_REG_INT_CLEAR

Address: Operational Base + offset (0x0028)

GP Control Register Interrupt Clear

Any bit written as 1 clears the corresponding interrupt bit in the GP_CONTR_REG_INT_RAWSTAT register. Any bit written as 0 is ignored and has no effect.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	W	-	GP_IRQ_AXI_BUS_ERROR; AXI bus error.
10	-	-	Reserved.
9	W	-	GP_IRQ_WRITE_BOUND_ERR; Write boundaries error.
8	W	-	GP_IRQ_PERF_CNT_1_LIMIT; Performance counter 1 limit reached.
7	W	-	GP_IRQ_PERF_CNT_0_LIMIT; Performance counter 0 limit reached.
6	W	-	GP_IRQ_FORCED_HANG; Forced hang.
5	W	-	GP_IRQ_HANG; Hang.
4	W	-	GP_IRQ_PLB_SEM; PLB semaphore decremented.
3	W	-	GP_IRQ_VS_SEM; Vertex shader semaphore incremented.
2	W	-	GP_IRQ_PLB_OUT_OF_MEM; PLB out of list memory.
1	W	-	GP_IRQ_PLB_END_CMD_LIST; PLB end of command list.
0	W	-	GP_IRQ_VS_END_CMD_LIST; Vertex shader end of command list.

GP_CONTR_REG_INT_MASK

Address: Operational Base + offset (0x002C)

GP Control Register Interrupt Mask

The GP_CONTR_REG_INT_MASK Register holds the bit mask that enables an interrupt source if the corresponding mask bit is set to 1.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	RW	0x0	GP_IRQ_AXI_BUS_ERROR;

			AXI bus error.
10	-	-	Reserved.
9	RW	0x0	GP_IRQ_WRITE_BOUND_ERR; Write boundaries error.
8	RW	0x0	GP_IRQ_PERF_CNT_1_LIMIT; Performance counter 1 limit reached.
7	RW	0x0	GP_IRQ_PERF_CNT_0_LIMIT; Performance counter 0 limit reached.
6	RW	0x0	GP_IRQ_FORCED_HANG; Forced hang.
5	RW	0x0	GP_IRQ_HANG; Hang.
4	RW	0x0	GP_IRQ_PLB_SEM; PLB semaphore decremented.
3	RW	0x0	GP_IRQ_VS_SEM; Vertex shader semaphore incremented.
2	RW	0x0	GP_IRQ_PLB_OUT_OF_MEM; PLB out of list memory.
1	RW	0x0	GP_IRQ_PLB_END_CMD_LIST; PLB end of command list.
0	RW	0x0	GP_IRQ_VS_END_CMD_LIST; Vertex shader end of command list.

GP_CONTR_REG_INT_STAT

Address: Operational Base + offset (0x0030)

GP Control Register Interrupt Status

The GP_CONTR_REG_INT_STAT Register is raw status ANDed with the interrupt mask. An interrupt is active if this register is non-zero.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	R	0x0	GP_IRQ_AXI_BUS_ERROR; AXI bus error.
10	-	-	Reserved.
9	R	0x0	GP_IRQ_WRITE_BOUND_ERR; Write boundaries error.
8	R	0x0	GP_IRQ_PERF_CNT_1_LIMIT; Performance counter 1 limit reached.
7	R	0x0	GP_IRQ_PERF_CNT_0_LIMIT; Performance counter 0 limit reached.
6	R	0x0	GP_IRQ_FORCED_HANG; Forced hang.
5	R	0x0	GP_IRQ_HANG; Hang.

4	R	0x0	GP_IRQ_PLB_SEM; PLB semaphore decremented.
3	R	0x0	GP_IRQ_VS_SEM; Vertex shader semaphore incremented.
2	R	0x0	GP_IRQ_PLB_OUT_OF_MEM; PLB out of list memory.
1	R	0x0	GP_IRQ_PLB_END_CMD_LIST; PLB end of command list.
0	R	0x0	GP_IRQ_VS_END_CMD_LIST; Vertex shader end of command list.

GP_CONTR_REG_WRITE_BOUND_LOW

Address: Operational Base + offset (0x0034)

GP Control Register Write Boundary Low

The GP_CONTR_REG_WRITE_BOUND_LOW Register contains the low write boundary. Writing outside the boundary causes all bus operations to halt, and the GP_IRQ_WRITE_BOUND_ERR interrupt is asserted.

Bit	Attr	Reset Value	Description
31:8	RW	0x0	GP_CONTR_REG_WRITE_BOUND_LOW; Value of low write boundary.
7:0	-	-	Reserved.

GP_CONTR_REG_WRITE_BOUND_HIGH

Address: Operational Base + offset (0x0038)

GP Control Register Write Boundary High

The GP_CONTR_REG_WRITE_BOUND_HIGH Register contains the high write boundary. Writing outside the boundary causes all bus operations to halt, and the GP_IRQ_WRITE_BOUND_ERR interrupt is asserted.

Bit	Attr	Reset Value	Description
31:8	RW	0xFFFFFFF	GP_CONTR_REG_WRITE_BOUND_HIGH; Value of high write boundary.
7:0	-	-	Reserved.

GP_CONTR_REG_PERF_CNT_0_ENABLE

Address: Operational Base + offset (0x003C)

GP Control Register Performance Counter 0 Enable

The GP_CONTR_REG_PERF_CNT_0_ENABLE Register enables performance counter 0. Writing a 1 to the register enables the counter.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	GP_CONTR_REG_PERF_CNT_0_ENABLE;

		Enable performance counter 0
--	--	------------------------------

GP_CONTR_REG_PERF_CNT_1_ENABLE

Address: Operational Base + offset (0x0040)

GP Control Register Performance Counter 1 Enable

The GP_CONTR_REG_PERF_CNT_1_ENABLE Register enables performance counter 1. Writing a 1 to the register enables the counter.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	GP_CONTR_REG_PERF_CNT_1_ENABLE; Enable performance counter 1

GP_CONTR_REG_PERF_CNT_x_SRC (x=0, 1)

Address: Operational Base + offset (0x0044, 0x0048)

GP Control Register Performance Counter x Source

The GP_CONTR_REG_PERF_CNT_x_SRC Register contains the source value for performance counter x.

Bit	Attr	Reset Value	Description																																
31:6	-	-	Reserved.																																
5:0	RW	0x0	<p>GP_CONTR_REG_PERF_CNT_x_SRC;</p> <table border="1"> <thead> <tr> <th>value</th> <th>Source</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Cycle counter.</td> </tr> <tr> <td>1</td> <td>Active cycles, geometry processor.</td> </tr> <tr> <td>2</td> <td>Active cycles, vertex shader.</td> </tr> <tr> <td>3</td> <td>Active cycles, vertex storer.</td> </tr> <tr> <td>4</td> <td>Active cycles, vertex loader.</td> </tr> <tr> <td>5</td> <td>Cycles vertex loader waiting for vertex shader.</td> </tr> <tr> <td>6</td> <td>Number of words read, system bus. Words are 64-bits.</td> </tr> <tr> <td>7</td> <td>Number of words written, system bus. Words are 64-bits.</td> </tr> <tr> <td>8</td> <td>Number of read bursts, system bus.</td> </tr> <tr> <td>9</td> <td>Number of write bursts, system bus.</td> </tr> <tr> <td>10</td> <td>Number of vertices processed.</td> </tr> <tr> <td>11</td> <td>Number of vertices fetched.</td> </tr> <tr> <td>12</td> <td>Number of primitives fetched.</td> </tr> <tr> <td>13</td> <td>Reserved.</td> </tr> <tr> <td>14</td> <td>Number of cullings performed, including backface cullings and out-of-frustum cullings.</td> </tr> </tbody> </table>	value	Source	0	Cycle counter.	1	Active cycles, geometry processor.	2	Active cycles, vertex shader.	3	Active cycles, vertex storer.	4	Active cycles, vertex loader.	5	Cycles vertex loader waiting for vertex shader.	6	Number of words read, system bus. Words are 64-bits.	7	Number of words written, system bus. Words are 64-bits.	8	Number of read bursts, system bus.	9	Number of write bursts, system bus.	10	Number of vertices processed.	11	Number of vertices fetched.	12	Number of primitives fetched.	13	Reserved.	14	Number of cullings performed, including backface cullings and out-of-frustum cullings.
value	Source																																		
0	Cycle counter.																																		
1	Active cycles, geometry processor.																																		
2	Active cycles, vertex shader.																																		
3	Active cycles, vertex storer.																																		
4	Active cycles, vertex loader.																																		
5	Cycles vertex loader waiting for vertex shader.																																		
6	Number of words read, system bus. Words are 64-bits.																																		
7	Number of words written, system bus. Words are 64-bits.																																		
8	Number of read bursts, system bus.																																		
9	Number of write bursts, system bus.																																		
10	Number of vertices processed.																																		
11	Number of vertices fetched.																																		
12	Number of primitives fetched.																																		
13	Reserved.																																		
14	Number of cullings performed, including backface cullings and out-of-frustum cullings.																																		

		15	Number of commands written to tiles.
		16	Number of memory blocks allocated.
		17	Reserved.
		18	Reserved.
		19	Number of vertex loader cache misses.
		20	Reserved.
		21	Reserved.
		22	Active cycles, vertex shader command processor.
		23	Active cycles, PLB command processor.
		24	Active cycles, PLB list writer.
		25	Active cycles, through the prepare list commands.
		26	Reserved.
		27	Active cycles, primitive assembly.
		28	Active cycles, PLB vertex fetcher.
		29	Reserved.
		30	Active cycles, bounding box and command generator.
		31	Reserved.
		32	Active cycles, scissor tile iterator.
		33	Active cycles, polygon tile iterator.

GP_CONTR_REG_PERF_CNT_x_VAL (x=0, 1)

Address: Operational Base + offset (0x004C, 0x0050)

GP Control Register Performance Counter x Value

The GP_CONTR_REG_PERF_CNT_x_VAL Register contains the current value for performance counter x.

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	R	0x0	GP_CONTR_REG_PERF_CNT_x_VAL; Contains the current value for performance counter x.

GP_CONTR_REG_PERF_CNT_x_LIMIT (x=0, 1)

Address: Operational Base + offset (0x0054, 0x0058)

GP Control Register Performance Counter x Limit

The GP_CONTR_REG_PERF_CNT_x_LIMIT Register contains the limit value for performance counter x.

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	RW	0x0	GP_CONTR_REG_PERF_CNT_x_LIMIT;

		Limit value for performance counter x.
--	--	--

GP_CONTR_REG_STATUS

Address: Operational Base + offset (0x0068)

GP Control Register Status

The GP_CONTR_REG_STATUS Register contains the geometry processor status.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	R	0x0	GP_STATUS_WRITE_BOUND_ERR; Write boundaries error detected.
7	R	0x0	GP_STATUS_HANG; Hang detected.
6	R	0x0	GP_STATUS_BUS_ERROR; Bus error detected.
5	R	0x0	GP_STATUS_PLB_STALLED; PLB stalled on list allocation.
4	-	-	Reserved.
3	R	0x0	GP_STATUS_PLB_ACTIVE; PLB active.
2	R	0x0	GP_STATUS_BUS_STOPPED; Stop command issued.
1	R	0x0	GP_STATUS_VS_ACTIVE; Vertex shader active.
0	R	0x0	GP_STATUS_IRQ; IRQ asserted.

GP_CONTR_REG_VSCL_INITIAL_ADDR

Address: Operational Base + offset (0x0080)

GP Control Register VSCL Initial Address

The GP_CONTR_REG_VSCL_INITIAL_ADDR Register holds the initial start address of the vertex shader command list.

Bit	Attr	Reset Value	Description
31:3	R	0x0	GP_CONTR_REG_VSCL_INITIAL_ADDR; Start address of vertex shader command list.
2:0	-	-	Reserved.

GP_CONTR_REG_PLBCL_INITIAL_ADDR

Address: Operational Base + offset (0x0084)

GP Control Register PLBCL Initial Address

The GP_CONTR_REG_PLBCL_INITIAL_ADDR Register holds the initial start address of the PLB command list.

Bit	Attr	Reset Value	Description
31:3	R	0x0	GP CONTR_REG_PLBCL_INITIAL_ADDR; Start address of PLB command list.
2:0	-	-	Reserved.

GP CONTR_REG_WRITE_BOUNDARY_ERROR_ADDR

Address: Operational Base + offset (0x0088)

GP Control Register Write Error Address

The GP CONTR_REG_WRITE_BOUNDARY_ERROR_ADDR Register holds the address of the last bus-access causing a write boundary error.

Bit	Attr	Reset Value	Description
31:0	R	0x0	GP CONTR_REG_WRITE_BOUNDARY_ERROR_ADDR; Address of last bus-access causing a write boundary error

GP CONTR_REG_BUS_ERROR_STAT

Address: Operational Base + offset (0x0094)

GP Control AXI Bus Error Status

After a bus error, the GP CONTR_REG_BUS_ERROR_STAT Register stores the status of the error.

Below table shows what kind of memory access corresponds to each error ID.

ID	Access Type
0	VS vertex storer writes
1	PLB pointer array writes
2	PLB polygon list writes
3	PLB command processor register writes
4	VS vertex loader reads
5	VS command processor reads
6	Reserved
7	PLB Command Processor reads
8	PLB vertex indices reads
9	PLB pointer array reads
10	PLB vertex fetcher
11~15	Reserved

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved.
9:6	R	0x0	GP_READ_ERROR_ID; ID of read error cause.

5:2	R	0x0	GP_WRITE_ERROR_ID; ID of write error cause.
1	R	0x0	GP_READ_ERROR; Set when a read error occurs.
0	R	0x0	GP_WRITE_ERROR; Set when a write error occurs.

GP_CONTR_REG_WATCHDOG_DISABLE

Address: Operational Base + offset (0x00A0)

GP Control Register Watchdog Disable

The GP_CONTR_REG_WATCHDOG_DISABLE Register controls the watchdog timer.

Setting the watchdog disable bit disables the watchdog timer. This also prevents the force hang command from triggering a hang interrupt.

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	GP_CONTR_REG_WATCHDOG_DISABLE; Disable watchdog timer.

GP_CONTR_REG_WATCHDOG_TIMEOUT

Address: Operational Base + offset (0x00A4)

GP Control Register Watchdog Timeout

The GP_CONTR_REG_WATCHDOG_TIMEOUT Register contains the number of cycles before a watchdog interrupt is triggered.

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	RW	0xF4240	GP_CONTR_REG_WATCHDOG_TIMEOUT; Watchdog timeout period.

GP_PLB_CONF_REG_VERTEX_ARRAY_ADDR

Address: Operational Base + offset (0x0100)

GP PLB Configuration Register Vertex Array Address

The GP_PLB_CONF_REG_VERTEX_ARRAY_ADDR Register is write-only. The vertex data array base address, used for glDrawElements mode, is written to this register. The register must be 64-byte aligned.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_VERTEX_ARRAY_ADDR Vertex data array base address.

GP_PLB_CONF_REG_INDEX_ARRAY_ADDR

Address: Operational Base + offset (0x0101)

GP PLB Configuration Register Index Array Address

The vertex index array base address, used for glDrawElements mode, is written to the GP_PLB_CONF_REG_INDEX_ARRAY_ADDR Register. The register must be aligned to the size of the vertex index for correct operation.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_INDEX_ARRAY_ADDR; Vertex index array base address

GP_PLB_CONF_REG_POINT_SIZE_ADDR

Address: Operational Base + offset (0x0102)

GP PLB Configuration Register Point Size Address

The point size address, used for glDrawElements mode, is written to the

GP_PLB_CONF_REG_POINT_SIZE_ADDR Register. The register must be 16-byte aligned.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_POINT_SIZE_ADDR; Point size address.

GP_PLB_CONF_REG_HEAP_START_ADDR

Address: Operational Base + offset (0x0103)

GP PLB Configuration Register Heap Start Address Register

The polygon list heap start address is written to the

GP_PLB_CONF_REG_HEAP_START_ADDR Register. The register must be 16-byte aligned.

Bit	Attr	Reset Value	Description
31:7	W	-	PL heap start address; Polygon list heap start address.
6:0	-	-	Reserved.

GP_PLB_CONF_REG_HEAP_END_ADDR

Address: Operational Base + offset (0x0104)

GP PLB Configuration Register Heap End Address Register

The polygon list heap end address is written to the

GP_PLB_CONF_REG_HEAP_END_ADDR Register. Writing to this register updates the same register as writing to the GP_CONTR_REG_PLB_ALLOC_END_ADDR Register

Bit	Attr	Reset Value	Description
31:7	W	-	PL heap end address; Polygon list heap end address.
6:0	-	-	Reserved.

GP_PLB_CONF_REG_VIEWPORT_TOP

Address: Operational Base + offset (0x0105)

GP PLB Configuration Register Viewport Top

Use the GP_PLB_CONF_REG_VIEWPORT_TOP Register to set the top of the viewport, in screenspace pixels, in FP32 format. You must set the top of the viewport, in screenspace pixels before a Process Shaded Vertices command.

The viewport affects the list building as follows:

- Triangles are culled if they are outside the viewport, or only written to tiles within the intersection of viewport and scissor box. To get proper, pixel-aligned, viewport clipping, you must set the scissor box to the intersection of viewport and scissor box.
- Lines are culled if fully outside the viewport, but otherwise ignore the viewport. For correct rendering, you must write a viewport command to the relevant tile lists, so the pixel processor can do proper viewport clipping of the lines.
- Points are culled if the vertex is outside the viewport.

Note: The geometry processor supports the full FP32 range for viewport co-ordinates, but the viewport command required for lines restricts the usable values to integers in the range [-8192, 8191].

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_VIEWPORT_TOP; Viewport top value.

GP_PLB_CONF_REG_VIEWPORT_BOTTOM

Address: Operational Base + offset (0x0106)

GP PLB Configuration Register Viewport Bottom

Use the GP_PLB_CONF_REG_VIEWPORT_BOTTOM Register to set the bottom of the PLB viewport, in screenspace pixels, in FP32 format. You must set the bottom of the PLB viewport, in screenspace pixels before a Process Shaded Vertices command.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_VIEWPORT_BOTTOM; Viewport bottom value.

GP_PLB_CONF_REG_VIEWPORT_LEFT

Address: Operational Base + offset (0x0107)

GP PLB Configuration Register Viewport Left

Use the GP_PLB_CONF_REG_VIEWPORT_LEFT Register to set the left of the PLB viewport, in screenspace pixels, in FP32 format. You must set the left of the PLB viewport, in screenspace pixels before a Process Shaded Vertices command.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_VIEWPORT_LEFT; Viewport left value.

GP_PLB_CONF_REG_VIEWPORT_RIGHT

Address: Operational Base + offset (0x0108)
 GP PLB Configuration Register Viewport Right

Use the GP_PLB_CONF_REG_VIEWPORT_RIGHT Register to set the right of the PLB viewport, in screenspace pixels, in FP32 format. You must set the right of the PLB viewport, in screenspace pixels before a Process Shaded Vertices command.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_VIEWPORT_RIGHT; Viewport right value.

GP_PLB_REG_SCREENSIZE

Address: Operational Base + offset (0x0109)
 GP PLB Configuration Register Screen Size

Use the GP_PLB_CONF_REG_SCREENSIZE Register to configure the screen size in number of tiles. This register acts as a bounding box for all commands that must be put in all tiles, for example, end of frame and set new base addresses. The register value is inclusive, so bottom must be the number of tile lists in Y direction - 1, and right must be the number of tile lists in X direction - 1. These screen size in number of tiles must be set before issuing any commands or primitives to the polygon list.

Bit	Attr	Reset Value	Description
31:24	W	-	GP_PLB_CONF_REG_SCREENSIZE_RIGHT; RHS tile (X_MAX) that is to be included in the bounding box.
23:16	W	-	GP_PLB_CONF_REG_SCREENSIZE_LEFT; LHS tile (X_MIN) that is to be included in the bounding box.
15:8	W	-	GP_PLB_CONF_REG_SCREENSIZE_BOTTOM Bottom tile (Y_MAX) that is to be included in the bounding box.
7:0	W	-	GP_PLB_CONF_REG_SCREENSIZE_TOP; Top tile (Y_MIN) that is to be included in the bounding box.

GP_PLB_CONF_REG_OFFSET_VERTEX_ARRAY

Address: Operational Base + offset (0x010A)
 GP PLB Configuration Register Offset Vertex Array

The GP_PLB_CONF_REG_OFFSET_VERTEX_ARRAY Register contains the offset into the vertex data array for the next vertex command.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_PLB_CONF_REG_OFFSET_VERTEX_ARRAY; Index offset for vertex array

GP_PLB_CONF_REG_PARAMS
 Address: Operational Base + offset (0x010B)
 GP PLB Configuration Register Parameters

The GP_PLB_CONF_REG_PARAMS Register contains the miscellaneous PLB parameters.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19	W	-	big_endian_idx; 0 = vertex indexes are stored in little-endian byte order. 1 = vertex indexes are stored in big-endian byte order, and must be converted on loading.
18	W	-	backf_cull_cv; Backface culling: cull if clockwise.
17	W	-	backf_cull_ccv; Backface culling: cull if counterclockwise.
16:15	W	-	polygon_mode; Select polygon display mode: 0 = Normal filled polygons 1 = reserved, do not use 2 = convert triangles to three points 3 = convert triangles to three lines.
14	W	-	line_width_vertex_select; Select whether line width must be taken from the first or second vertex: 0 = first vertex 1 = second vertex Don't care when point_size_override is set to 1.
13	W	-	point_clipping_select; Select between standard point clipping, that is the center of point, and the points bounding box: 0 = standard clipping 1 = bounding box clipping.
12	W	-	point_size_override; Use point size/line width written in GP_PLB_CONF_REG_POINT_SIZE instead of the size found in the point size array: 0 = Use point size array 1 = Use register setting.
11:10	W	-	vertex_idx_format; Vertex index array format: 0 = 8-bit

			1 = 16-bit 2 = 32-bit.
9:8	W	-	list_alloc_size; Select the size of each allocated block: 0 = 128 bytes 1 = 256 bytes 2 = 512 bytes 3 = 1024 bytes. This affects both the preallocated list and the heap. The list must be aligned to the size selected here.
7:0	W	-	rsw_index; Renderer state word index.

GP_PLB_CONF_REG_TILE_SIZE

Address: Operational Base + offset (0x010C)

GP PLB Configuration Register Tile Size

The GP_PLB_CONF_REG_TILE_SIZE Register determines how binning is performed. When both X and Y are set to zero, each tile list describes primitives in one tile. With other settings, each list contains commands and primitives for several tiles. The maximum number of tile lists is 300, so QVGA is supported without tile sharing. For VGA, a sensible setting is X = 1, Y = 1.

Bit	Attr	Reset Value	Description
31:22	-	-	Reserved.
21:16	W	-	Y tile size; 2n number of tiles to be binned to one tile list in Y direction.
15:6	-	-	Reserved.
5:0	W	-	X tile size; 2n number of tiles to be binned to one tile list in X direction.

GP_PLB_CONF_REG_POINT_SIZE

Address: Operational Base + offset (0x010D)

GP PLB Configuration Register Point Size

The GP_PLB_CONF_REG_POINT_SIZE Register holds a point size in FP32 format. This point size value is used when point_size_override is set in GP_PLB_CONF_REG_PARAMS. This register is also used for line widths.

Bit	Attr	Reset Value	Description
31:0	W	-	point_size; Point size override value

GP_PLB_CONF_REG_Z_NEAR

Address: Operational Base + offset (0x010E)

GP PLB Configuration Register Z Near

The GP_PLB_CONF_REG_Z_NEAR Register holds the near Z-plane position in FP32 format.

Bit	Attr	Reset Value	Description
31:0	W	-	Z_near; Z co-ordinate used for frustum culling against near plane.

GP_PLB_CONF_REG_Z_FAR

Address: Operational Base + offset (0x010F)

GP PLB Configuration Register Z Far

The GP_PLB_CONF_REG_Z_FAR Register holds the far Z plane position in FP32 format.

Bit	Attr	Reset Value	Description
31:0	W	-	Z_far; Z co-ordinate used for frustum culling against far plane.

GP_VS_CONF_REG_INP_ADDRx

Address: Operational Base + offset (0x0000 ~ 0x001E)

GP VS Configuration Register Input Address x

The GP_VS_CONF_REG_INP_ADDRx Register contains the base data address for input data stream x. The offset address is (0x0000+2*x)

Bit	Attr	Reset Value	Description
31:0	W	-	GP_VS_CONF_REG_INP_ADDRx; Base address of vertex array.

GP_VS_CONF_REG_INP_SPECx

Address: Operational Base + offset (0x0001 ~ 0x001F)

GP VS Configuration Register Input Specifier x

The GP_VS_CONF_REG_INP_SPECx Register contains the data format and stride for input data stream x. The offset address is (0x0001+2*x).

If a given format has less than four components, the remaining components are loaded as 1.0 for the last component and 0.0 for other remaining components.

Below table shows the vertex data format:

Value	Name	Format
[63]	GP_VS_VSTREAM_NO_DATA	No data
[62:60]	No use	Reserved
[59:56]	GP_VS_VSTREAM_FORMAT_1_NORM_U32	1-4 32-bit unsigned normalized

	GP_VS_VSTREAM_FORMAT_4_NORM_U32	
[55:52]	GP_VS_VSTREAM_FORMAT_1_NORM_S32 GP_VS_VSTREAM_FORMAT_4_NORM_S32	1-4 32-bit signed normalized
[51:48]	GP_VS_VSTREAM_FORMAT_1_FP24 GP_VS_VSTREAM_FORMAT_4_FP24	1-4 floats, FP24
[47:44]	GP_VS_VSTREAM_FORMAT_1_NORM_U16 GP_VS_VSTREAM_FORMAT_4_NORM_U16	1-4 16-bit unsigned normalized
[43:40]	GP_VS_VSTREAM_FORMAT_1_NORM_S16 GP_VS_VSTREAM_FORMAT_4_NORM_S16	1-4 16-bit signed normalized
[39:36]	GP_VS_VSTREAM_FORMAT_1_NORM_U8 GP_VS_VSTREAM_FORMAT_4_NORM_U8	1-4 8-bit unsigned normalized
[35:32]	GP_VS_VSTREAM_FORMAT_1_NORM_S8 GP_VS_VSTREAM_FORMAT_4_NORM_S8	1-4 8-bit signed normalized
[31:28]	GP_VS_VSTREAM_FORMAT_1_FIX_U8 GP_VS_VSTREAM_FORMAT_4_FIX_U8	1-4 8-bit fixedpoint, unsigned
[27:24]	GP_VS_VSTREAM_FORMAT_1_FIX_S8 GP_VS_VSTREAM_FORMAT_4_FIX_S8	1-4 8-bit fixedpoint, signed
[23:20]	GP_VS_VSTREAM_FORMAT_1_FIX_U16 GP_VS_VSTREAM_FORMAT_4_FIX_U16	1-4 16-bit fixedpoint, unsigned
[19:16]	GP_VS_VSTREAM_FORMAT_1_FIX_S16 GP_VS_VSTREAM_FORMAT_4_FIX_S16	1-4 16-bit fixedpoint, signed
[15:12]	GP_VS_VSTREAM_FORMAT_1_FP16 GP_VS_VSTREAM_FORMAT_4_FP16	1-4 floats, FP16
[11:8]	GP_VS_VSTREAM_FORMAT_1_FIX_U32 GP_VS_VSTREAM_FORMAT_4_FIX_U32	1-4 32-bit fixedpoint, unsigned
[7:4]	GP_VS_VSTREAM_FORMAT_1_FIX_S32 GP_VS_VSTREAM_FORMAT_4_FIX_S32	1-4 32-bit fixedpoint, signed
[3:0]	GP_VS_VSTREAM_FORMAT_1_FP32 GP_VS_VSTREAM_FORMAT_4_FP32	1-4 floats, FP32

Bit	Attr	Reset Value	Description
31	W	-	Big Endian; 0 = the value is stored in little-endian byte order, and is converted on loading. 1 = the value is stored in big-endian byte order, and is converted on loading.
30:11	W	-	Stride; Value in 1-byte units. The stride is defined as the number of bytes separating two consecutive vertices.
10:6	W	-	Comma; Comma location, for fixed point formats only.
5:0	W	-	Vertex data form; See above table for vertex data format value. The reset value is 0x3F.

GP_VS_CONF_REG_OUTP_ADDRx

Address: Operational Base + offset (0x0020 ~ 0x003E)

GP VS Configuration Register Output Address x

The GP_VS_CONF_REG_OUTP_ADDRx Register contains
the base data address for output data stream x. The offset address is (0x0020+2*x)

Bit	Attr	Reset Value	Description
31:0	W	-	GP_VS_CONF_REG_OUTP_ADDRx; Base address of the output vertex stream specifier.

GP_VS_CONF_REG_INP_SPECx

Address: Operational Base + offset (0x0021 ~ 0x003F)

GP VS Configuration Register Output Spec x

The GP_VS_CONF_REG_INP_SPEC(X) Register contains the data format and stride for
output data stream. The offset address is (0x0021+2*x).

The vertex data for use by the pixel processor must be written using format 32.

Below table shows the vertex data format:

Value	Name	Format
[63]	GP_VS_VSTREAM_NO_DATA	No data
[62:53]	Unused	Reserved
[51:48]	GP_VS_VSTREAM_FORMAT_1_FP24	1-4 floats, FP24
	GP_VS_VSTREAM_FORMAT_4_FP24	
[47]	Unused	Reserved
[46]	GP_VS_VSTREAM_FORMAT_RGB565	RGB 565, output only
[45:34]	Unused	Reserved

[33]	GP_VS_VSTREAM_FORMAT_POINT_SIZE	Point size in FP32 format, output only
[32]	GP_VS_VSTREAM_FORMAT_VDATA_BLOCK (XYZW)	Vertex co-ordinates in FP32 format, output only
[31:28]	GP_VS_VSTREAM_FORMAT_1_FIX_U8 GP_VS_VSTREAM_FORMAT_4_FIX_U8	1-4 8-bit fixedpoint, unsigned
[27:24]	GP_VS_VSTREAM_FORMAT_1_FIX_S8 GP_VS_VSTREAM_FORMAT_4_FIX_S8	1-4 8-bit fixedpoint, signed
[23:20]	GP_VS_VSTREAM_FORMAT_1_FIX_U16 GP_VS_VSTREAM_FORMAT_4_FIX_U16	1-4 16-bit fixedpoint, unsigned
[19:16]	GP_VS_VSTREAM_FORMAT_1_FIX_S16 GP_VS_VSTREAM_FORMAT_4_FIX_S16	1-4 16-bit fixedpoint, signed
[15:12]	GP_VS_VSTREAM_FORMAT_1_FP16 GP_VS_VSTREAM_FORMAT_4_FP16	1-4 floats, FP16
[11:8]	GP_VS_VSTREAM_FORMAT_1_FIX_U32 GP_VS_VSTREAM_FORMAT_4_FIX_U32	1-4 32-bit fixedpoint, unsigned
[7:4]	GP_VS_VSTREAM_FORMAT_1_FIX_S32 GP_VS_VSTREAM_FORMAT_4_FIX_S32	1-4 32-bit fixedpoint, signed
[3:0]	GP_VS_VSTREAM_FORMAT_1_FP32 GP_VS_VSTREAM_FORMAT_4_FP32	1-4 floats, FP32

Bit	Attr	Reset Value	Description
31	W	-	Big Endian; 0 = the value is stored in little-endian byte order, and is converted on loading. 1 = the value is stored in big-endian byte order, and is converted on loading.
30:11	W	-	Stride; Value in 1-byte units. The stride is defined as the number of bytes separating two consecutive vertices.
10:6	W	-	Comma; Comma location, for fixed point formats only.
5:0	W	-	Vertex data form; See above table for vertex data format value. The reset value is 0x3F.

GP_VS_CONF_PROG_PARAM_CREATE

Address: Operational Base + offset (0x0040)

GP VS Configuration Program Parameter Create

The GP_VS_CONF_PROG_PARAM_CREATE Register specifies the number of vertex shader instructions, in the native format, to run per vertex. Also, the index of the last instruction touching the input registers is given when the instruction with this index is executed; the input registers are marked free and available for the vertex loader to load new vertices into.

Bit	Attr	Reset Value	Description
31:20	W	-	Index of last instruction to touch input registers; Index value of last instruction.
19:10	W	-	End of vertex shader program; End address.
9:0	W	-	Start of vertex shader program; Start address.

GP_VS_CONF_REG_PREFETCH

Address: Operational Base + offset (0x0041)

GP VS Configuration Register Prefetch

The GP_VS_CONF_REG_PREFETCH Register specifies the number of cache lines to prefetch in the vertex shader when loading vertex data.

Bit	Attr	Reset Value	Description															
31:2	-	-	Reserved.															
9:0	W	-	GP_VS_CONF_REG_PREFETCH; Number of cache lines required.															
			<table> <thead> <tr> <th>Value</th> <th>Cache Line</th> <th>Burst length used fetching data</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>Four 32-bits words</td> </tr> <tr> <td>1</td> <td>2</td> <td>Eight 32-bits words</td> </tr> <tr> <td>2</td> <td>-</td> <td>Reserved</td> </tr> <tr> <td>3</td> <td>4</td> <td>Sixteen 32-bits words</td> </tr> </tbody> </table>	Value	Cache Line	Burst length used fetching data	0	1	Four 32-bits words	1	2	Eight 32-bits words	2	-	Reserved	3	4	Sixteen 32-bits words
Value	Cache Line	Burst length used fetching data																
0	1	Four 32-bits words																
1	2	Eight 32-bits words																
2	-	Reserved																
3	4	Sixteen 32-bits words																

GP_VS_CONF_REG_OPMOD

Address: Operational Base + offset (0x0042)

GP VS Configuration Register OPMOD

The GP_VS_CONF_REG_OPMOD Register defines the operational mode of the vertex shader.

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.

27:24	W	-	run_input_regs; Number of input registers to read in, minus 1. Reset = 0xF.
23:12	W	-	Reserved.
11:8	W	-	run_output_regs; Number of output registers to write-back, minus 1.
7:0	W	-	GP_VS_CONF_REG_PREFETCH; Number of cache lines required.

GP_VS_CONF_REG_VERTICES_ALT_STRIDE

Address: Operational Base + offset (0x0043)

GP VS Configuration Register Vertices Alternative Stride

The GP_VS_CONF_REG_VERTICES_ALT_STRIDE Register holds the number(n) of vertices to process using the regular stride before switching to the alternative stride. A stride is an offset between two data sets. See also GP VS Configuration Register Input Specifierx 0x0001 + (2*x) and GP VS Configuration Register Output Specifierx at 0x0021-0x003F.

Bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:16	W	-	Vertices per alternative stride, output; The number of times (nth) to use the regular stride before using the alternative stride.
15:14	W	-	Reserved.
13:0	W	-	Vertices per alternative stride, input; The number of times (nth) to use the regular stride before using the alternative stride.

GP_VS_CONF_REG_INPUT_ALT_STRIDE_x (x= 0, 1, 2, 3)

Address: Operational Base + offset (0x0044, 0x0045, 0x0046, 0x0047)

GP VS Configuration Register Input Alternative Stride x

There are 16 streams of vertex data that can be input. The GP_VS_CONF_REG_INPUT_ALT_STRIDE_x Register provides alternative stride for input vertices 0-15 (Stride 0 for 0-3; Stride 1 for 4-7; Stride 2 for 8-11; Stride 3 for 12-15). The value written to this register is therefore used the nth time instead of the vertex shader regular stride.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_VS_CONF_REG_INPUT_ALT_STRIDE_x; Value.

GP_VS_CONF_REG_OUTP_ALT_STRIDE_x (x= 0, 1, 2, 3)

Address: Operational Base + offset (0x0048, 0x0049, 0x004A, 0x004B)

GP VS Configuration Register Output Alternative Stride x

There are 16 streams of vertex data that can be output. The

GP_VS_CONF_REG_OUTP_ALT_STRIDE_0 Register provides alternative stride for output vertices 0-15 (Stride 0 for 0-3; Stride 1 for 4-7; Stride 2 for 8-11; Stride 3 for 12-15). The value written to this register is therefore used the nth time instead of the vertex shader normal stride.

Bit	Attr	Reset Value	Description
31:0	W	-	GP_VS_CONF_REG_OUTP_ALT_STRIDE_x; Value.

MMU Configuration Registers

MMU_DTE_ADDR

Address: Operational Base + offset (0x0000)

MMU Current Page Table Address Register

The MMU_DTE_ADDR Register contains the base address of the current page tables. The address must be 4KB aligned. This register can only be written when the MMU is idle or stalled.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	MMU_DTE_ADDR; Page table address.

MMU_STATUS

Address: Operational Base + offset (0x0004)

MMU Status Register

The MMU_STATUS Register shows the current status of the MMU.

Bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:6	R	0x0	MMU_PAGE_FAULT_BUS_ID; Index of master responsible for last page fault.
5	R	0x0	MMU_PAGE_FAULT_IS_WRITE; The direction of access for last page fault: 0 = Read 1 = Write
4	R	0x1	MMU_REPLAY_BUFFER_EMPTY; The MMU replay buffer is empty.
3	R	0x1	MMU_IDLE; The MMU is idle.
2	R	0x0	MMU_STALL_ACTIVE; MMU stall mode currently enabled. The mode is enabled by command.
1	R	0x0	MMU_PAGE_FAULT_ACTIVE; MMU page fault mode currently enabled. The mode is enabled by command.
0	R	0x0	MMU_PAGING_ENABLED;

		Paging is enabled.
--	--	--------------------

MMU_CMD

Address: Operational Base + offset (0x0008)

MMU Command Register

The MMU_CMD Register performs certain operations affecting the MMU state.

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	W	-	<p>MMU_CMD. This can be:</p> <p>0 = MMU_ENABLE_PAGING. Enable paging.</p> <p>1 = MMU_DISABLE_PAGING. Disable paging.</p> <p>2 = MMU_ENABLE_STALL. Turn on stall mode.</p> <p>3 = MMU_DISABLE_STALL. Turn off stall mode.</p> <p>4 = MMU_ZAP_CACHE. Zap the entire page table cache.</p> <p>5 = MMU_PAGE_FAULT_DONE. Leave page fault mode.</p> <p>6 = MMU_SOFT_RESET. Reset the MMU.</p> <p>The MMU_ENABLE_STALL command can always be issued. Other commands are ignored unless the MMU is idle or stalled.</p>

MMU_PAGE_FAULT_ADDR

Address: Operational Base + offset (0x000C)

MMU Logical Address of Last Page Fault Register

The MMU_PAGE_FAULT_ADDR Register contains the address of the last page fault.

Bit	Attr	Reset Value	Description
31:0	R	0x0	MMU_PAGE_FAULT_ADDR; Address of last page fault.

MMU_ZAP_ONE_LINE

Address: Operational Base + offset (0x0010)

MMU Zap Cache Line Register

Writing an address to the MMU_ZAP_ONE_LINE Register causes the page table entry corresponding to the address to be invalidated from the page table cache.

Bit	Attr	Reset Value	Description
31:0	W	0x0	MMU_ZAP_ONE_LINE; Address to be invalidated from the page table cache.

MMU_INT_RAWSTAT

Address: Operational Base + offset (0x0014)
 MMU Raw Interrupt Status Register

The MMU_INT_RAWSTAT Register shows the unmasked status of the interrupt sources. Any bit written as 1 is forced on, and generates an interrupt if not masked. Any bit written as 0 is ignored and has no effect.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	RW	0x0	Read bus error.
0	RW	0x0	Page fault.

MMU_INT_CLEAR

Address: Operational Base + offset (0x0018)
 MMU Interrupt Clear Register

The MMU_INT_CLEAR Register is used to clear interrupt bits in the MMU_INT_RAWSTAT Registers. Any bit written as 1 to the MMU_INT_CLEAR Register clears the corresponding interrupt bit in the MMU_INT_RAWSTAT register. Any bit written as 0 to the MMU_INT_CLEAR Register is ignored and has no effect.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	W	-	Read bus error.
0	W	-	Page fault.

MMU_INT_MASK

Address: Operational Base + offset (0x001C)
 MMU Interrupt Mask Register

The MMU_INT_MASK Register holds the bit mask that enables an interrupt source if the corresponding mask bit is set to 1.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	RW	0x0	Read bus error.
0	RW	0x0	Page fault.

MMU_INT_STAT

Address: Operational Base + offset (0x0020)
 MMU Interrupt Status Register

The MMU_INT_STAT Register is raw status ANDed with the interrupt mask. An interrupt is active if this register is non-zero.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	R	0x0	Read bus error.

0	R	0x0	Page fault.
---	---	-----	-------------

Notes: Attr: RW – Read/writable, R – Read only, W – Write only

Functional Description

Geometry Processor Vertex Shader

The vertex shader consists of:

- Vertex loader
- Vertex shader core
- Vertex storer

Vertex loader

The vertex loader is a DMA unit that loads per-vertex data for processing. It can accept data from up to 16 distinct streams, each corresponding to one of the 16 input registers.

For each stream, it permits the specification of any of the following data formats:

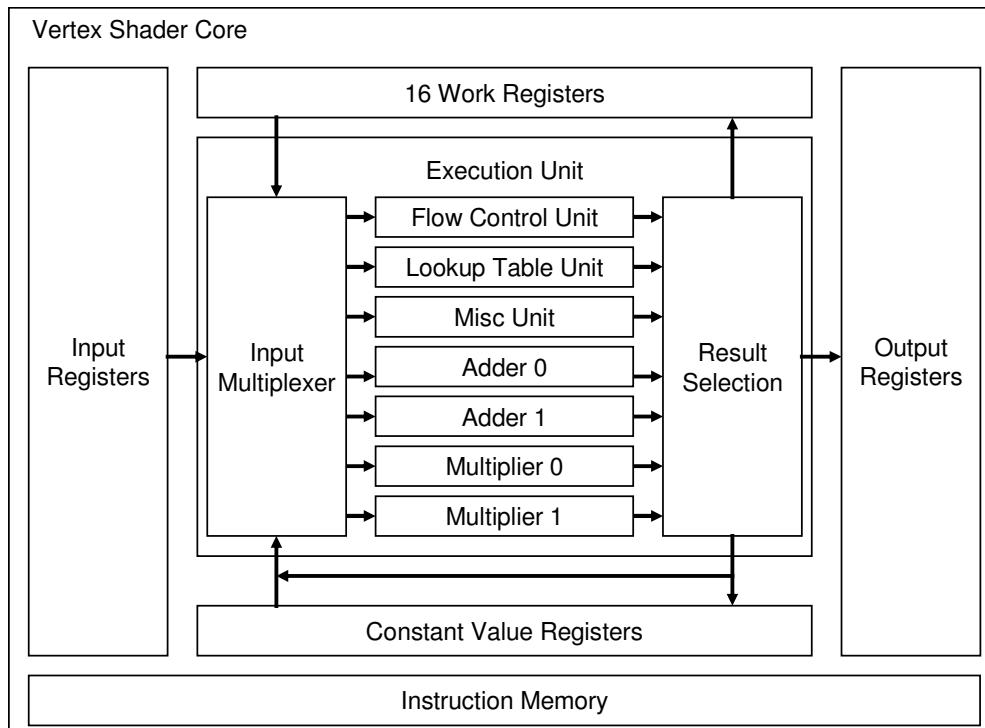
- 1, 2, 3, or 4 values in 16-bit, 24-bit, or 32-bit floating-point formats
- 1, 2, 3, or 4 values in 8-bit, 16-bit, or 32-bit signed or unsigned fixed point values
- 1, 2, 3, or 4 values in 8-bit, 16-bit, or 32-bit signed or unsigned normalized values.

For each vertex stream, you must specify a stride in 1-byte increments. A stride is an offset between two data sets.

Vertex shader core

The vertex shader core is the most important single unit of the geometry processor. This unit performs most of the required calculations for each vertex. The vertex shader runs a program in each vertex of a 3D model, typically performing T&L for the model. The program is limited to 512 instructions with limited flow control. The instructions of the program work on vector data, optimized for operations in length-4 vectors and smaller vectors.

Below Figure shows the vertex shader core block diagram.



The vertex shader core contains information about:

- Instruction memory
- Vertex shader execution units
- Working registers
- Input registers
- Output registers
- Constant value registers
- Input multiplexer

Instruction memory

The instruction memory contains the instructions to execute for each vertex. It can contain up to 512 instructions, and must be loaded through the vertex shader command list before the vertex shader is started.

Vertex shader execution units

The vertex shader performs all data calculations for the geometry processor. The following units are included:

Adders

The vertex shader core contains two full 32-bit floating-point adders. The adders can perform

additions, or other numeric operations such as minimum, maximum, and floor.

Multipliers

The vertex shader core contains two full 32-bit floating-point multipliers. You can use these multipliers for generic arithmetic or as part of a special multiply-add unit for LookUp Table (LUT) operations.

Lookup table unit

The LUT unit performs table lookups as a basis for calculating reciprocals, logarithms and other non-trivial functions. The LUT unit reciprocals, logarithms and other outputs the coefficients for an approximation equation for the selected function, and the final result must be interpolated using the multipliers and adders.

Miscellaneous operations unit

The miscellaneous operations unit handles fixed-point and floating-point conversions, in addition to other operations.

Flow control unit

The flow control unit handles program flow, and also controls some special operations. In addition to the normal program flow operations such as sequential flow, conditional/unconditional branches, and function call/return, the flow control unit also handles:

- Hardware loops
- Input register bank switching
- Constant value register writes

The hardware loop functionality contains a dedicated loop counter and a special address register. You can repeat a code sequence a pre-defined number of times, optionally updating the address register, without using the normal arithmetic units to update a loop counter. Hardware loops cannot be nested.

Working registers

The working registers are a set of 16 vector registers that you can use for storing intermediate values. Each register contains four 32-bit floating-point values.

A single instruction can read two working registers, and write one working register.

There is a 3-cycle delay from the time a value is written to a working register, until the written value can be read back.

Input registers

The input registers store data loaded by the vertex loader, and make the data available to the vertex shader core. There are 16 input registers, corresponding to the 16 input streams the vertex loader can handle. Each register is a vector of four 32-bit floating-point values. The vertex shader program can read from the input registers, but it cannot write to them.

The input registers are double-buffered for performance, so that the vertex loader can fill one bank while the vertex shader core is using the contents of the other bank. This is normally hidden from the programmer, because the banks automatically switch when processing of a vertex finishes. In some cases you can control the double buffering. For example:

- When the start and end address of the shader program is configured, it is also possible to tell the vertex shader the instruction that is last to read the input registers. The vertex shader core can use this information to switch banks earlier, giving the vertex loader more time to fetch data.
- If a shader requires more than 16 input values, it can explicitly switch banks by issuing the Switch input bank flow control operation.

Output registers

The output registers are 16 vector registers where the vertex shader program can store the results of the calculations. Each register is a vector of four 32-bit floating-point values. Each register corresponds to one output stream that the vertex storer can write to memory. The vertex shader program can write to the output registers, but it cannot read from them.

Constant value registers

The constant value registers are a set of 304 vector registers that store mostly static data. Each register contains four 32-bit floating-point values. These registers can be loaded through the vertex shader command list, or they can be written by the vertex shader. Unlike the working registers, only one constant value register can be read or written by any one instruction, and there is a 4-cycle delay from when a value is written to a constant value register to when the written value can be read back.

Input multiplexer

The input multiplexer enables each functional unit to use either register values or the result of previous calculations as inputs. This means that temporary calculations do not have to allocate register file slots, and that the functional units can work on more data items than the register file can supply in a single cycle.

Vertex storer

The vertex storer stores data from the output registers of the vertex shader in memory. The vertex loader can export data to FP24, FP16, 32-bit integer, 16-bit integer, and 8-bit integer.

Geometry Processor Polygon List Builder

The PLB creates lists of the polygons that the pixel processor must draw. For each polygon in a scene, the list builder decides which tiles the polygon covers, and adds the polygon to the lists that draw those tiles. The PLB only adds a polygon to lists where the polygon might have to be drawn, reducing the work involved when the pixel processor renders the scene.

The PLB also discards polygons that are certain not to be visible, based on the following criteria:

- They are not valid polygons. For example, any co-ordinate is not-a-number, X or Y co-ordinate is infinity, or area is zero.
- The polygons are entirely outside the view frustum.
- The polygons are facing away from the screen.
- The polygons are entirely outside the current scissoring box.

The PLB can handle up to 300 lists to support the tile-based rendering mode of the pixel

processor efficiently. For QVGA or lower resolutions, 300 lists are normally sufficient to make one list for every tile in the scene. For higher resolutions, each list covers multiple tiles, a process known as binning.

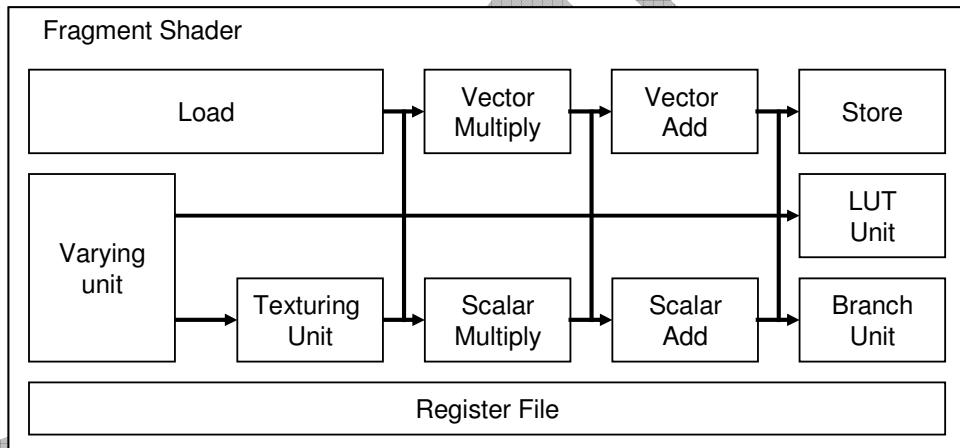
Each list ends with a return command. The driver creates a master tile list containing sets of commands that perform tasks such as beginning new tiles and calling polygon lists. This master tile list is input to the pixel processor polygon list reader. The PLB requires a variable amount of memory to store the polygon lists. Memory is allocated in blocks of 128, 256, 512, or 1024 bytes, as configured in the

GP_PLB_CONF_REG_PARAMS register. The driver must allocate the initial memory consisting of an array with one block for each polygon list. Further memory is allocated automatically by the PLB from a heap area configured through the GP_PLB_CONF_REG_HEAP_START_ADDR and

GP_PLB_CONF_REG_HEAP_STOP_ADDR registers. When this heap is exhausted, an interrupt is generated, and the driver must allocate more memory for the heap.

Pixel Processor Fragment Shader

The fragment shader is a programmable unit that calculates how each fragment of a primitive looks. The fragment shader program specified in the RSW for the primitive is executed for each fragment produced by the rasterizer. The fragment shader program consists of Very Long Instruction Words (VLIW), and can use any number of functional units in a single instruction. Below figure shows the functional units available for the fragment shader program.



Register file

The register file contains a number of vector registers, each consisting of four FP16 values. The pixel processor currently only implements six vector registers for size reasons, although the instruction set can address 16 registers. In addition to the real registers 0-5, there are three pseudo-registers #load, #var and #tex. These hold the results from the load, varying, or texturing units, and are only available in the same cycle as the instruction that produces the result.

Varying unit

The varying unit can select varying data from the vertex data block and interpolate these across the primitive. These data can be used for texture co-ordinates, lighting values, or other uses. The varying unit is the only unit that can produce texture co-ordinates for the texturing unit.

Texturing unit

The texturing unit, also referred to as the texture mapper, produces texture data for the fragment. It takes a texture descriptor index, and a set of texture co-ordinates from the varying unit, and produces filtered texture data.

Multiply-add unit

The multiply-add unit consists of one vector multiplier, one scalar multiplier, one vector adder, and one scalar adder. Each unit can be individually controlled, or you can combine their operation to perform composite operations, such as dot products.

LUT unit

The LUT unit produces interpolation coefficients for complex functions, such as square roots and arctangents. The results from the LUT unit are the coefficients of a linear or quadratic approximation curve for the selected function, and must be interpolated by the multiply-add unit to produce the final result.

Load and store unit

The load and store unit can access system memory, perform indexed register accesses, load values, or directly read tile buffer content.

Branch unit

The branch unit controls the program flow of the shader program. It can perform conditional and unconditional branches, calls, returns, and shader termination. If no branch instruction is included in the main instruction, the shader execution continues in sequential order.

Memory Management

The GPU contains an MMU to control and translate memory accesses that the processor initiates. The MMU controls data through data structures based on pages and tables. You can configure the MMU by writing to control registers in

- MMU page directory data structure
- MMU page table data structure
- MMU configuration register descriptions

The MMU divides memory into 4KB pages, where each page can configure:

- The real memory address of the page. Known as address translation or virtual memory, this enables the processor to work using addresses that differ from the physical addresses in the memory system.
- The permitted types of accesses to that page. Each page can permit reads, writes, both, or none.

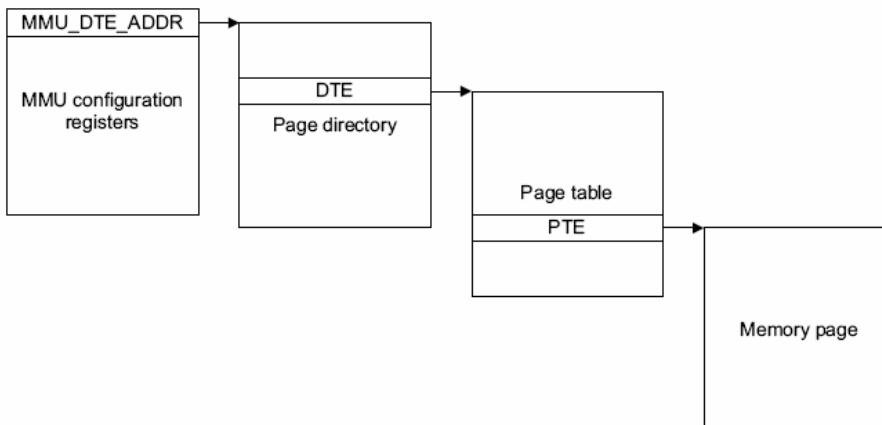
The MMU uses a 2-level page table structure:

1. The first level consists of 1024, 32-bit descriptors as Directory Table Entries (DTE),

each pointing to a Page Table Entry (PTE).

2. The second level consists of 1024 PTEs. Each PTE table is a 4KB memory-array, with each entry pointing to an actual page.

Below figure shows the structure of the two level page table.



Below Figure shows the arrangement of the MMU address bits.

31	22	21	12	11	0
DTE index		PTE index		Page offset	

The MMU uses the following algorithm to translate an address:

1. Find the DTE at address given by

$$\text{MMU_DTE_ADDR} + (4 * \text{DTE index})$$
2. Find the PTE at address given by:

$$\text{Page table address from DTE} + (4 * \text{PTE index})$$
3. Calculate effective address as follows:

$$\text{Page address from PTE} + \text{Page offset.}$$

Page fault behavior

The system enters a page fault mode if the PTE that the system is looking up indicates an access violation. An access violation can happen because:

- The present bit in the PTE is not set
- The present bit in the DTE is not set
- The GPU attempts a read access and the read permission bit is not set
- The GPU attempts a write access and the write permission bit is not set

When a page fault happens, the MMU automatically enters page fault mode and triggers an

interrupt. The MMU does not translate any more accesses until the system explicitly clears the page fault. The MMU Logical Address of Last Page Fault Register stores the virtual address that caused the fault, and more information is available in the MMU Status Register. If the page fault occurs as a result of the processor accessing memory that it is not permitted to access, you must reset the processor. After a circuit reset, paging and page fault modes are disabled by default, and the MMU is idle.

Page directory behavior

The page directory is a 4KB data structure that contains 1024, 32-bit DTEs. The page directory must align at a 4KB boundary in memory.

Each DTE contains the address of a page table and a page table present bit. The system must:

- Initialize the entire page directory before use
- Clear the page table present bit for any DTE that does not point to a valid page table.

31	12	11	1	0
Page table address		Reserved		0

The above Figure shows the format of the directory table entries.

Below Table shows the function of the directory table entries.

Bits	Name	Function
31:12	Page table address	This field stores bits [31:12] of the address for a page table.
11:1	Reserved	Reserved, write as zero.
0	Page table present	<p>This bit indicates when the page table address points to a valid page table.</p> <p>1 = page table valid</p> <p>0 = page table not valid.</p>

If the present bit is not set at either the DTE or the PTE level for a given address, this indicates that the virtual memory address does not exist, and attempts to access it result in an immediate page fault. If the present bit is not set at the DTE level, then the actual PTE table that it points to does not exist.

Updating page tables

To update page tables, the MMU writes to the tables in system memory.

The MMU is guaranteed not to access memory:

- When it is in page fault mode
- When it has been explicitly stalled.

To stall the MMU write MMU_ENABLE_STALL to the MMU Command Register. To unstall the MMU write MMU_DISABLE_STALL. The MMU buffers some DTE and PTE entries.

The system must inform the MMU when these change so that the MMU does not use the old values. When the system updates a PTE, it must invalidate the PTE in the MMU. To do this the system writes a virtual address covered by the PTE to the MMU Zap Cache Line Register. If

the change only adds privileges, then it is not strictly necessary to invalidate the line. Failure to invalidate the line might trigger a page fault later, because the MMU uses a buffered version of the PTE. If this happens, invalidate the faulting address and leave page fault mode.

If the system reduces the privileges of a PTE, or changes the address mapping, it must invalidate the PTE immediately.

When the system updates a DTE or changes the MMU Current Page Table Address Register, it must clear all entries buffered by the MMU. To do this it writes MMU_ZAP_CACHE to the MMU Command Register.

BUS error behavior

The MMU reads DTEs and PTEs from system memory. If any such read request fails, the MMU asserts the Read Bus Error interrupt. After a bus error, the MMU might be in an inconsistent state and the entire processor must be reset.

Replay buffer

If a bus access request cannot be handled immediately by the MMU, because of the appropriate DTEs or PTEs having to be read from memory, the access is placed into a special replay buffer. This enables other accesses to complete without blocking on an unrelated access. Accesses in the replay buffer are periodically retried, and when the correct page tables are available, the access completes.

The MMU_REPLAY_BUFFER_EMPTY bit in the MMU_STATUS register is set if no accesses are waiting in the replay buffer. Accesses in the replay buffer might cause page faults even after paging is disabled.

Configuration interfaces

You can configure the pixel processor and geometry processor using the following sets of configuration registers:

- You can configure the pixel processor and geometry processor control registers using a set of control registers. The control registers are accessible through an AMBA APB bus, and are usually mapped into the memory space of the system CPU.
- The geometry processor Vertex Shader configuration registers are only accessible through the vertex shader command list. These registers control the behavior of the vertex shader only.
- The geometry processor PLB configuration registers are only accessible through the PLB command list. These registers control the behavior of the PLB only.

Interrupts

Both the pixel processor and geometry processor have a number of logical interrupts, that are ORed together to a single physical interrupt line. The pixel processor has nine logical interrupts and the geometry processor has 11. When an interrupt occurs, software must read the Interrupt Status register to find what caused the interrupt.

The Interrupt Rawstat register contains one bit for each logical interrupt. The Interrupt Mask register enables masking of individual interrupts. The Interrupt Status register contains the masked interrupt status. Use the Interrupt Clear register to clear an interrupt.

Performance counters

Both pixel processor and geometry processor have two performance counters that you can use

to gather statistics or profile an application. You can configure the two performance counters individually to count any of a number of internal events. The counters can optionally generate an interrupt when they reach a pre-defined value.

Watchdog Timers

Both pixel processor and geometry processor contains a watchdog timer that you can use to detect if no progress occurs over a specified time period. The specified time period is the time in clock cycles that you specify in the watchdog timeout registers. This typically occurs if a shader program contains unterminated loops. If no progress occurs within the watchdog timeout period, the HANG interrupt is asserted.

In the pixel processor the watchdog timer starts counting each time a fragment enters the fragment shader. Each new fragment restarts the count from zero. In the geometry processor, the watchdog timer starts counting each time the PLB or the vertex loader fetches a vertex. Each new vertex restarts the count from zero.

Both watchdog timers stop counting when:

- The watchdog timer is disabled
- Any interrupt occurs
- The connected processor becomes idle

In the pixel processor, if the watchdog is enabled, you can force the watchdog timer to trigger by writing a 1 to the CTRL_MGMT register.

In the geometry processor, if the watchdog is enabled, you can force the watchdog timer to trigger by writing GP_CMD_FORCE_HANG to the GP CONTR_REG_CMD

Register. Both watchdog timers are enabled by default, with a timeout of 1000000 clock cycles.

Chapter 12 Ethernet 10/100M MAC Controller

Overview

The Ethernet 10/100 MAC, here after referred to as MAC, is a high-speed LAN controller. It implements Carrier Sense Multiple Access with Collision Detection (CSMA/CD) algorithms defined by IEEE 802.3 for media access control over the 10Mbps and 100Mbps Ethernet.

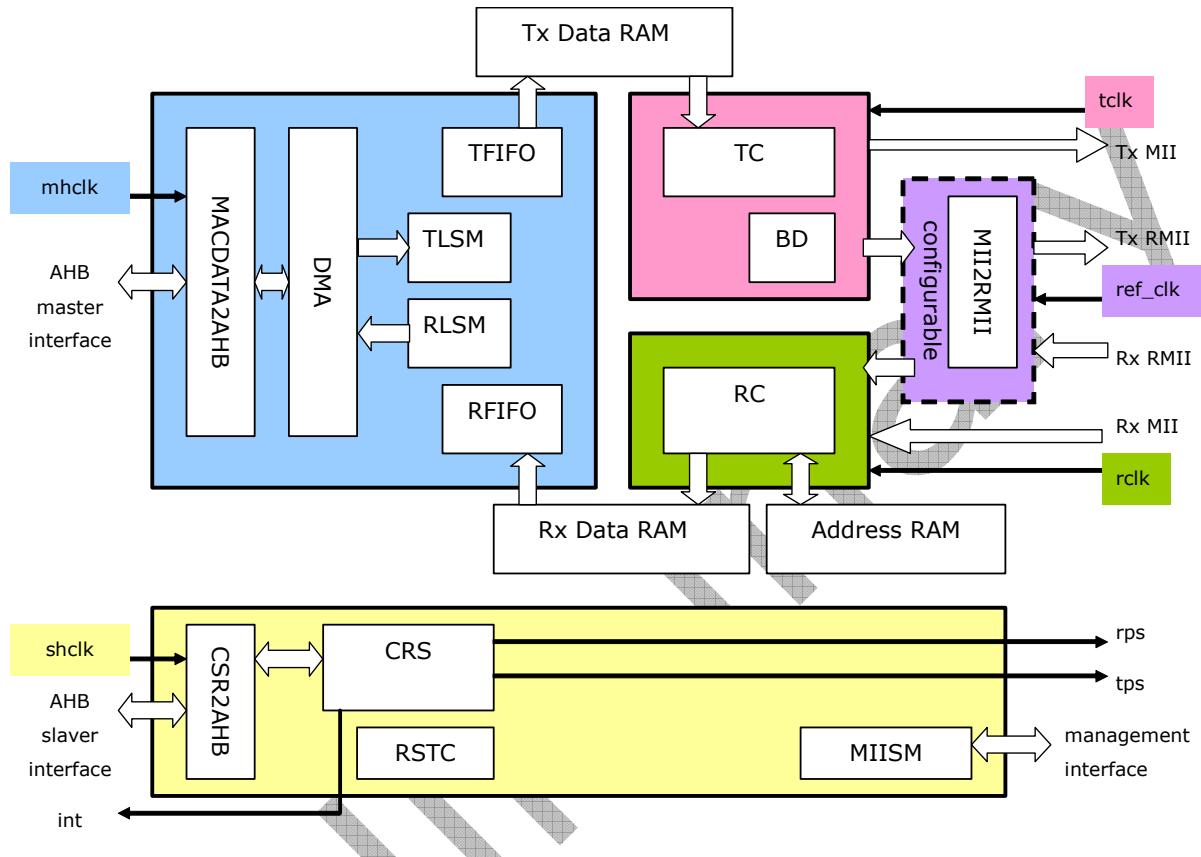
Communication with an external host is implemented via a set of Control and Status Registers and the DMA controller for external shared RAM memory. For data transfers the MAC operates as a DMA master. It automatically fetches from transmit data buffers and stores receive data buffers into external RAM with minimum CPU intervention. The linked list management enables the use of various memory allocation schemes. nh

Key Features

- Network interface features
 - Supports 10/100 Mb/s data transfer rate
 - Media Independent Interface (MII)
 - Reduced Media Independent Interface (RMII, configurable)
 - Data link layer functionality
 - Meets the IEEE 802.3 CSMA/CD standard
 - Full or half duplex operation
 - Flexible address filtering
 - Up to 16 physical addresses
 - 512-bit hash table for multicast addresses
 - Control and status registers
 - Single interrupt line
 - Interrupt mitigation control mechanism
 - DMA controller
 - Scatter/gather capabilities
 - Configurable burst length
 - Intelligent bus arbitration schemes
 - Descriptor/buffer architecture for data storage
 - Descriptor "ring" or "chain" structures
 - Single descriptor points to up to two data buffers
 - Automatic descriptor list polling
 - Low power capabilities
 - Independent clocks for data and control paths
 - Running/Suspended/Stopped modes
 - Clock switching support
 - Transmit/receive dual port RAM interfaces
 - Operates as internal configurable FIFOs
 - Programmable transmit threshold levels
- Transmit FIFO "store and forward" functionality

Architecture

Block Diagram



Block Descriptions

TC – Transmit Controller

The transmit controller implements the 802.3 transmit operation. From the network side it uses the standard 802.3 MII interface for an external PHY device. The transmit controller operates synchronously with the clkt clock from the module.

BD – Backoff/Deferring

The backoff/deferring controller implements the 802.3 half duplex operation. It operates synchronously with the clkt clock from the module. The backoff/deferring controller can be optionally removed for lower gate count if the half duplex operation is not required.

RC – Receive Controller

The receive controller implements the 802.3 receive operation. From the network side it uses the standard 802.3 MII interface for an external PHY device. The receive controller operates

synchronously with the clkr clock from the module.

TFIFO – Transmit FIFO Controller

The transmit FIFO controller is used for buffering data prepared for transmission by the MAC. It provides an interface for the external dual-port RAM working as FIFO memory. The FIFO size can be configured by the generic parameters of the core. The $TFIFODEPTH$ parameter defines the total FIFO size. The FIFO size is equal to $2^{TFIFODEPTH} - 1$. The $TCDEPTH$ parameter defines the maximum number of frames that can reside in the transmit FIFO at the moment. The actual number is equal to $2^{TCDEPTH} - 1$. The transmit FIFO controller operates synchronously with the mhclk clock from the host interface.

RFIFO – Receive FIFO Controller

The receive FIFO controller is used for buffering data received by the MAC. It provides an interface for the external dual-port RAM working as FIFO memory. The FIFO size can be configured by the generic parameters of the core. The $RFIFODEPTH$ parameter defines the total FIFO size. The FIFO size is equal to $2^{RFIFODEPTH} - 1$. The $RCDEPTH$ parameter defines the maximum number of frames that can reside in the receive FIFO at the moment. The actual number is equal to $2^{RCDEPTH} - 1$. The receive FIFO controller operates synchronously with the mhclk clock from the host interface.

TLSM – Transmit linked List State Machine

The transmit linked list state machine implements the descriptor / buffer architecture of the MAC. It manages the transmit descriptor list, and fetches the data prepared for transmission from the data buffers into the transmit FIFO. The transmit linked list state machine controller operates synchronously with the mhclk clock from the host interface.

RLSM – Receive linked List State Machine

The receive linked list state machine implements the descriptor/ buffer architecture of the MAC. It manages the receive descriptor list, and moves the data the receive FIFO into the data buffers. The receive linked list state machine controller operates synchronously with the mhclk clock from the host interface.

DMA – Direct Memory Access controller

The direct memory access controller implements the host Data interface. It services both the receive and the transmit channels. The direct memory access controller operates synchronously with the mhclk clock from the host interface.

MACDATA2AHB – AHB master wrapper

The MAC data to AHB interface wrapper. It translates MAC data access operations to AHB control, address and data signals. The AHB wrapper component operates synchronously with the mhclk clock from the host AHB master interface.

CSR – Control and Status Registers

The CSR component is used to control the MAC operation by the host. It implements the register set, the interrupt controller, and the power management functionality of the MAC. It

also provides an interface for the host.

MACCSR2AHB – AHB slave wrapper

The MAC CSR to AHB slave interface wrapper. It translates host read or write AHB operations into MAC CSR interface operations. The APB wrapper component operates synchronously with the shclk clock from the host AHB slave interface.

RSTC – Reset Controller

The reset controller is used to reset all components of the MAC. It generates a reset signal synchronous to all clock domains in the design from the single external reset line.

Dual Port RAMs

Receive data RAM

Synchronous dual port RAM working as the receive FIFO.

Transmit data RAM

Synchronous dual port RAM working as the transmit FIFO.

Address RAM

Synchronous dual port RAM working as the addresses memory.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
MAC_CSR0	0x0000	W	0xFE000000	Bus mode
MAC_CSR1	0x0008	W	0xFFFFFFFF	Transmit poll demand
MAC_CSR2	0x0010	W	0xFFFFFFFF	Receive poll demand
MAC_CSR3	0x0018	W	0xFFFFFFFF	Receive list base address
MAC_CSR4	0x0020	W	0xFFFFFFFF	Transmit list base address
MAC_CSR5	0x0028	W	0xF0000000	Status
MAC_CSR6	0x0030	W	0x32000040	Operation mode
MAC_CSR7	0x0038	W	0xF3FE0000	Interrupt enable
MAC_CSR8	0x0040	W	0xE0000000	Missed frames and overflow counters
MAC_CSR9	0x0048	W	0xFFFF483FF	MII management and serial ROM
MAC_CSR11	0x0058	W	0xFFFFE0000	Timer and interrupt mitigation control

Notes:

Size: B – Byte (8 bits) access, HW – Half WORD (16 bits) access, W – WORD (32 bits) access

Detail Register Description

MAC_CSR0

Address: Operational Base + offset (0x0000)

MAC_CSR0 is consisted of various controlling bits on AHB bus

Bit	Attr	Reset Value	Description																											
31:21	-	-	Reserved																											
20	RW	0x0	<p>Descriptor byte ordering mode (DBO)</p> <p>1 – big endian mode used for data descriptors.</p> <p>0 – little endian mode used for data descriptors.</p> <p>Changing this bit is allowed only when both the transmitter and receiver processes are in the stopped state.</p>																											
19:17	RW	0x0	<p>Transmit automatic polling (TAP)</p> <p>If the TAP is written with a nonzero value, the MAC performs an automatic transmit descriptor polling when operating in suspended state. When the descriptor is available, the transmit process goes into the running state. When the descriptor is marked as owned by the host, the transmit process remains suspended.</p> <p>The poll is always performed at the current transmit descriptor list position. The time interval between two consecutive polls is shown below.</p> <table border="1"> <thead> <tr> <th>CSR0[19:17]</th><th>10Mb/s</th><th>100Mb/s</th></tr> </thead> <tbody> <tr> <td>000</td><td>TAP disabled</td><td>TAP disabled</td></tr> <tr> <td>001</td><td>819us</td><td>81.9us</td></tr> <tr> <td>010</td><td>2.45ms</td><td>245us</td></tr> <tr> <td>011</td><td>5.73ms</td><td>573us</td></tr> <tr> <td>100</td><td>51.2us</td><td>5.12us</td></tr> <tr> <td>101</td><td>102.4us</td><td>10.24us</td></tr> <tr> <td>110</td><td>153.6us</td><td>15.36us</td></tr> <tr> <td>111</td><td>358.4us</td><td>35.84us</td></tr> </tbody> </table>	CSR0[19:17]	10Mb/s	100Mb/s	000	TAP disabled	TAP disabled	001	819us	81.9us	010	2.45ms	245us	011	5.73ms	573us	100	51.2us	5.12us	101	102.4us	10.24us	110	153.6us	15.36us	111	358.4us	35.84us
CSR0[19:17]	10Mb/s	100Mb/s																												
000	TAP disabled	TAP disabled																												
001	819us	81.9us																												
010	2.45ms	245us																												
011	5.73ms	573us																												
100	51.2us	5.12us																												
101	102.4us	10.24us																												
110	153.6us	15.36us																												
111	358.4us	35.84us																												
16:14	-	-	Reserved																											
13:8	RW	0x0	<p>Programmable burst length (PBL)</p> <p>Specifies the maximum number of words that can be transferred within one DMA transaction. Values permissible are 0, 1, 2, 4, 8, 16, and 32. When a '0' value is written, the bursts are limited only by the internal FIFO's threshold levels. For PBL values other than 0, the MAC uses sequential address mode for burst operations.</p> <p>Note that the PBL is valid only for the data buffers.</p> <p>Because of a FIFOs' configurability, some of the PBL settings can be invalid for some FIFO configurations. The total burst size (in bytes) should be less than the transmit FIFO size for the proper operation of the core.</p> <p>Changing this bit is allowed only when both the transmitter and receiver processes are in the stopped state.</p>																											
7	RW	0x0	<p>Big/Little endian (BLE)</p> <p>Selects the byte-ordering mode used by the data buffers.</p>																											

			1 – big endian mode used for the data buffers. 0 – little endian mode used for the data buffers. Changing this bit is allowed only when both the transmitter and receiver processes are in the stopped state.
6:2	RW	0x0	Descriptors skip length (DSL) Specifies the number of longwords between two consecutive unchained descriptors. Changing this bit is allowed only when both the transmitter and receiver processes are in the stopped state.
1	RW	0x0	Bus arbitration scheme (BAR) 1 - transmit and receive processes have equal priority to access the bus. 0 - intelligent arbitration where the receive process has priority over the transmit process. Changing this bit is allowed only when both the transmitter and receiver processes are in the stopped state.
0	RW	0x0	Software reset (SWR) It resets all internal components.

MAC_CSR1

Address: Operational Base + offset (0x0008)

Transmit poll demand (TPD)

Bit	Attr	Reset Value	Description
31:0	RW	0xFFFF_FFFF	Writing this field with any value instructs the MAC to check for frames to be transmitted. This operation is valid only when the transmit process is suspended. If no descriptor is available the transmit process remains suspended. When the descriptor is available the transmit process enters the running state.

MAC_CSR2

Address: Operational Base + offset (0x0010)

Receive poll demand (RPD)

Bit	Attr	Reset Value	Description
31:0	RW	0xFFFF_FFFF	Writing this field with any value instructs the MAC to check for receive descriptors to be acquired. This operation is valid only when the receive process is suspended. If no descriptor is available the receive process remains suspended. When the descriptor is available the receive process enters the running state.

MAC_CSR3

Address: Operational Base + offset (0x0018)

The receive descriptor list base address register (RLA)

Bit	Attr	Reset Value	Description
31:0	RW	0xFFFF_FFFF	Start of the receive list address. Contains the address of the first descriptor in a receive descriptor list. This address should be long-word aligned (RLA(1..0)=00).

MAC_CSR4

Address: Operational Base + offset (0x0020)

The transmit descriptor list base address register (TLA)

Bit	Attr	Reset Value	Description
31:0	RW	0xFFFF_FFFF	Start of the transmit list address. Contains the address of the first descriptor in a transmit descriptor list. This address should be long-word aligned (TLA(1..0)=00).

MAC_CSR5

Address: Operational Base + offset (0x0028)

MAC_CSR5 reflects current MAC status.

Bit	Attr	Reset Value	Description
31:23	-	-	Reserved
22:20	R	0x0	Transmit process state (TS) Indicates the current state of a transmit process: 000 – Stopped, RESET or STOP TRANSMIT command issued 001 – Running, fetching transmit descriptor. 010 – Running, waiting for end of transmission. 011 – Running, transferring data buffer from host memory to FIFO. 100 – Reserved. 101 – Running, setup packet. 110 – Suspended, FIFO underflow or unavailable descriptor. 111 – Running, closing transmit descriptor.
19:17	R	0x0	Receive process state (RS) Indicates the current state of a receive process: 000 – Stopped, RESET or STOP RECEIVE command issued. 001 – Running, fetching receive descriptor. 010 – Running, waiting for the end of receive packet before prefetch descriptor. 011 – Running, waiting for receive packet. 100 – Suspended, unavailable receive buffer. 101 – Running, closing receive descriptor.

			110 – Not used 111 – Running, transferring data from FIFO to host memory.
16	RW	0x0	<p>Normal interrupt summary (NIS)</p> <p>This bit is logical or on the following bits:</p> <ul style="list-style-type: none"> MAC_CSR5.0 – Transmit interrupt MAC_CSR5.2 – Transmit buffer unavailable MAC_CSR5.6 – Receive interrupt MAC_CSR5.11 – General purpose timer overflow MAC_CSR5.14 – Early receive interrupt <p>Only the unmasked bits affect the normal interrupt summary bit.</p> <p>The user can clear this bit by writing a '1'. Writing a '0' has no effect.</p>
15	RW	0x0	<p>Abnormal interrupt summary (AIS)</p> <p>This bit is logical or on the following bits:</p> <ul style="list-style-type: none"> MAC_CSR5.1 – Transmit process stopped MAC_CSR5.5 – Transmit underflow MAC_CSR5.7 – Receive buffer unavailable MAC_CSR5.8 – Receive process stopped MAC_CSR5.10 – Early transmit interrupt <p>Only the unmasked bits affect the abnormal interrupt summary bit. The user can clear this bit by writing a '1'. Writing a '0' has no effect.</p>
14	RW	0x0	<p>Early receive interrupt (ERI)</p> <p>Set when the MAC fills the data buffers of the first descriptor. Cleared by the MAC after the receive interrupt (MAC_CSR5.6).</p> <p>The user can clear this bit by writing 1. Writing 0 has no effect.</p>
13:12	-	-	Reserved
11	RW	0x0	<p>General-purpose timer expiration (GTE)</p> <p>Set when the general-purpose timer reaches a value of zero.</p> <p>The user can clear this bit by writing a '1'. Writing a '0' has no effect.</p>
10	RW	0x0	<p>Early transmit interrupt (ETI)</p> <p>Indicates that the packet to be transmitted was fully transferred into the FIFO. This bit is cleared by the MAC after the transmit interrupt (MAC_CSR5.0).</p> <p>The user can clear this bit by writing a '1'. Writing a '0' has no effect.</p>
9	-	-	Reserved
8	RW	0x0	Receive process stopped (RPS) RPS is set when a receive process enters a stopped state.

			The user can clear this bit by writing a '1'. Writing a '0' has no effect.
7	RW	0x0	Receive buffer unavailable (RU) When set, indicates that the next receive descriptor is owned by the host and is unavailable for the MAC. When RU becomes set, the MAC enters a suspended state, and returns to receive descriptor processing when the host changes ownership of the descriptor and either a receive poll demand command is issued or a new frame is recognized by the MAC. The user can clear this bit by writing a '1'. Writing a '0' has no effect.
6	RW	0x0	Receive interrupt (RI) Indicates the end of a frame receive. The complete frame has been transferred into the receive buffers. The user can clear this bit by writing a '1'. Writing a '0' has no effect.
5	RW	0x0	Transmit underflow (UNF) Indicates that the transmit FIFO was empty during a transmission. The transmit process goes into a suspended state. The user can clear this bit by writing a '1'. Writing a '0' has no effect.
4:3	-	-	Reserved
2	RW	0x0	Transmit buffer unavailable (TU) When set, TU indicates that the host owns the next descriptor on the transmit descriptor list therefore it cannot be used by the MAC. When the TU becomes set, the transmit process goes into a suspended state and can resume normal descriptor processing when the host changes ownership of the descriptor and either a transmit poll demand command is issued or transmit automatic polling is enabled. The user can clear this bit by writing a '1'. Writing a '0' has no effect.
1	RW	0x0	Transmit process stopped (TPS) TPS is set when the transmit process goes into a stopped state. The user can clear this bit by writing a '1'. Writing a '0' has no effect.
0	RW	0x0	Transmit interrupt (TI) Indicates the end of a frame transmission process. The user can clear this bit by writing a '1'. Writing a '0' has no effect.

MAC_CSR6

Address: Operational Base + offset (0x0030)

MAC_CSR6 is for controlling MAC operation modes such as 10/100 Mpbs, full/half duplex, and various filtering modes, etc.

Bit	Attr	Reset Value	Description
31	RW	0x1	Speed Selection (SS) Control bit for setting MII to RMII Converter operating at 100Mbps or 10Mbps. 0 – MII to RMII Converter operating at 10Mbps. 1 – MII to RMII Converter operating at 100Mbps. If MII2RMII Converter is not integrated in MAC, this bit can be ignored.
30	RW	0x0	Receive all (RA) When set, all incoming frames are received regardless of their destination address. An address check is performed and the result of the check is written into the receive descriptor (RDES0.30).
29:23	-	-	Reserved
22	RW	0x0	Transmit threshold mode (TTM) 0 – transmit threshold mode appropriate for 100 Mb/s mode. 1 – transmit threshold mode appropriate for 10 Mb/s mode. Changing this bit is allowed only when the transmitter process is in the stopped state.
21	RW	0x0	Store and forward (SF) When set, the transmission starts after a full packet is written into the transmit FIFO, regardless of the current FIFO threshold level. Changing this bit is allowed only when the transmitter process is in the stopped state.
20:16	-	-	Reserved
15:14	RW	0x0	Threshold control bits (TR) This bit, together with the TTM, the SF and the PS, control the threshold level for the transmit FIFO. Changing this bit is allowed only when the transmitter process is in the stopped state.

			MAC_CSR6[21]	MAC_CSR6 [15:14]	MAC_CSR6[22] =1	MAC_CSR6[22]=0
			0	00	64	128
			0	01	128	256
			0	10	128	512
			0	11	256	1024
			1	XX	Store and Forward	Store and Forward
13	RW	0x0	<p>Start/stop transmit command (ST) Setting this bit when the transmit process is in a stopped state causes transition into a running state. In the running state the MAC checks the transmit descriptor at a current descriptor list position. If the MAC owns the descriptor then the data starts to transfer from memory into the internal transmit FIFO. When the host owns the descriptor, the MAC enters a suspended state. Clearing this bit when the transmit process is in a running or a suspended state instructs the MAC to enter the stopped state.</p>			
12:10	-	-	Reserved			
9	RW	0x0	<p>Full duplex mode (FD) 0 – half duplex mode. 1 – forcing full duplex mode. Changing this bit is allowed only when both the transmitter and receiver processes are in the stopped state.</p>			
8			Reserved			
7	RW	0x0	<p>Pass all multicast (PM) When set, all the frames with the multicast destination addresses will be received regardless of the address check result.</p>			
6	RW	0x1	<p>Promiscuous mode (PR) When set all the frames will be received regardless of the address check result. An address check is not performed.</p>			
5	-	-	Reserved			
4	R	0x0	<p>Inverse filtering (IF) If this bit is set when working in a perfect filtering mode, the receiver performs an inverse filtering during the address check process. The "filtering type" bits of the setup frame determine a state of this bit.</p>			
3	RW	0x0	<p>Pass bad frames (PB) When set, the MAC transfers all frames into the data</p>			

			buffers, regardless of the receive errors. This allows the runt frames, collided fragments and truncated frames to be received.
2	R	0x0	Hash only filtering mode (HO) When set, the MAC performs an imperfect filtering over both the multicast and the physical addresses. The "filtering type" bits of the setup frame determine the state of this bit.
1	RW	0x0	Start/stop receive command (SR) Setting this bit when the receive process is in a stopped state causes the transition into a running state. In the running state the MAC checks the receive descriptor at the current descriptor list position. If the MAC owns the descriptor, then it can process an incoming frame. When the host owns the descriptor, the receiver enters a suspended state and also sets the MAC_CSR5.7 (receive buffer unavailable) bit. Clearing this bit when the receive process is in running or suspended state instructs the MAC to enter a stopped state after receiving the current frame.
0	R	0x0	Hash/perfect receive filtering mode (HP) 0 – perfect filtering of the incoming frames is performed according to the physical addresses specified in a setup frame. 1 – imperfect filtering over the frames with the multicast addresses is performed according to the hash table specified in a setup frame. A physical addresses check is performed according to the MAC_CSR6.2 (HO - Hash only) bit. When the HO and HP are both set, an imperfect filtering is performed on all of the addresses. The "filtering type" bits of the setup frame determine the state of this bit.

Notes: MAC_CSR6[4], MAC_CSR6[2] and MAC_CSR6[0] bits are read-only bits reflecting the filtering type FT0 and FT1 of TDES1 in the setup frame. All the possible combinations of the address filtering bits are listed in the table below. Please see TDES1 for details.

PM MAC_ CSR6[7]	PR MAC_ CSR6[6]	IF MAC_ CSR6[4]	HO MAC_ CSR6[2]	HP MAC_ CSR6[0]	Current Filtering Mode
0	0	0	0	0	16 physical addresses – perfect filtering mode
0	0	0	0	1	1 physical address for physical addresses and 512-bit hash table for multicast addresses
0	0	0	1	1	512-bit hash table for both physical and

					multicast addresses
0	0	1	0	0	Inverse filtering
X	1	0	0	X	Promiscuous mode
0	1	0	1	1	Promiscuous mode
1	0	0	0	X	Pass all multicast frames
1	0	0	1	1	Pass all multicast frames

MAC_CSR7

Address: Operational Base + offset (0x0038)

MAC_CSR7 lists all interrupts provided by MAC and is for enabling the interrupts required.

Bit	Attr	Reset Value	Description
31:17	-	-	Reserved
16	RW	0x0	Normal interrupt summary enable (NIE) When set, normal interrupts are enabled. Normal interrupts are listed below: CSR5.0 – Transmit interrupt CSR5.2 – Transmit buffer unavailable CSR5.6 – Receive interrupt CSR5.11 – General-purpose timer expired CSR5.14 – Early receive interrupt
15	RW	0x0	Abnormal interrupt summary enable (AIE) When set, abnormal interrupts are enabled. Abnormal interrupts are listed below: CSR5.1 – Transmit process stopped CSR5.5 – Transmit underflow CSR5.7 – Receive buffer unavailable CSR5.8 – Receive process stopped CSR5.10 – Early transmit interrupt
14	RW	0x0	Early receive interrupt enable (ERE) When both the ERE and normal interrupt enable bits are set, early receive interrupt is enabled.
13:12	-	-	Reserved
11	RW	0x0	General-purpose timer overflow enable (GTE) When both the GTE and normal interrupt summary enable bits are set, the general purpose timer overflow interrupt is enabled.
10	RW	0x0	Early transmit interrupt enable (ETE) When both the ETE and abnormal interrupt summary enable bits are set, the early transmit interrupt is enabled.
9	-	-	Reserved
8	RW	0x0	Receive stopped enable (RSE) When both the RSE and abnormal interrupt summary enable bits are set, the receive stopped interrupt is enabled.
7	RW	0x0	Receive buffer unavailable enable (RUE) When both the RUE and abnormal interrupt summary

			enable bits are set, the receive buffer unavailable is enabled.
6	RW	0x0	Receive interrupt enable (RIE) When both the RIE and normal interrupt summary enable bits are set, the receive interrupt is enabled.
5	RW	0x0	Underflow interrupt enable (UNE) When both the UNE and abnormal interrupt summary enable bits are set, the transmit underflow interrupt is enabled.
4:3	-	-	Reserved
2	RW	0x0	Transmit buffer unavailable enable (TUE) When both the TUE and normal interrupt summary enable bits are set, the transmit buffer unavailable interrupt is enabled.
1	RW	0x0	Transmit stopped enable (TSE) When both the TSE and abnormal interrupt summary enable bits are set, the transmit process stopped interrupt is enabled.
0	RW	0x0	Transmit interrupt enable (TIE) When both the TIE and normal interrupt summary enable bits are set, the transmit interrupt is enabled.

MAC_CSR8

Address: Operational Base + offset (0x0040)

MAC_CSR8 is consisted of counters recording the number of FIFO overflow and Missed frame.

Bit	Attr	Reset Value	Description
31:29	-	-	Reserved
28	R	0x0	Overflow counter overflow (OCO) Gets set when the FIFO overflow counter overflows. Reset when read.
27:17	R	0x0	FIFO overflow counter (FOC) Counts the number of frames not accepted due to the receive FIFO overflow. The counter resets when read.
16	R	0x0	Missed frame overflow counter (MFO) Set when a missed frame counter overflows. Reset when read.
15:0	R	0x0	Missed frame counter (MFC) Counts the number of frames not accepted due to the unavailability of the receive descriptor. The counter resets when read.

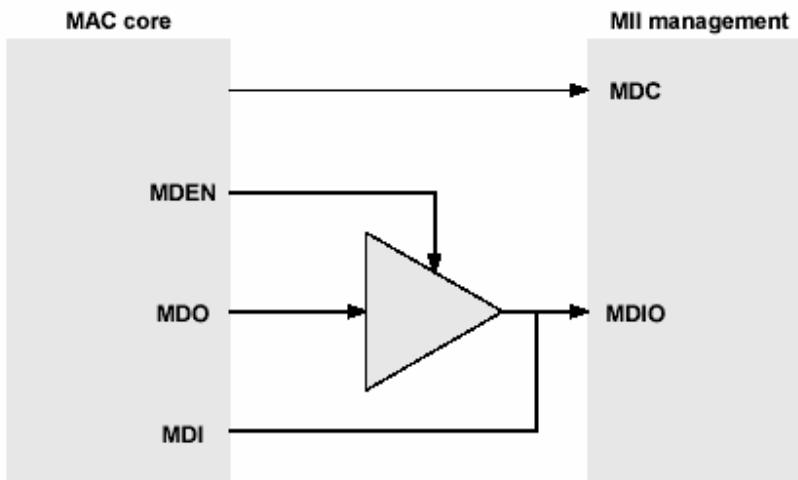
MAC_CSR9

Address: Operational Base + offset (0x0048)

MAC_CSR9 is used for control MII management signals interface.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved
19	R	0x0	MII management data in signal (MDI) This bit reflects the sample on the MDI pin during the read operation on the MII management interface.
18	RW	0x1	MII management operation mode (MII) 0 – indicates that the MAC reads the MII PHY registers. 1 – indicates that the MAC writes to the MII PHY registers.
17	RW	0x0	MII management write data (MDO) The value of this bit drives the MDO pin when a write operation is performed.
16	RW	0x0	MII management clock (MDC) The value of this bit drives the MDC pin.
15:0	-	-	Reserved

The MII management interface can be used to control the external PHY device from the host side. It allows access to all of the internal PHY registers via a simple two-wire interface. There are two signals on the MII management interface: the MDC (Management Data Clock), and the MDIO (Management Data Input/Output). The IEEE 802.3 indirection tristate signal defines the MDIO. The MAC core uses four unidirectional external signals to control the management interface. For proper operation of the interface, the user should supply an external tristate buffer as shown on figure below. Note that all access sequences and timing of the MII management interface are handled by the software.



MAC_CSR11

Address: Operational Base + offset (0x0058)

Brief function description of register REG_B

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31	RW	0x1	<p>Cycle size (CS)</p> <p>Controls the time units for the transmit and receive timers according to the following table:</p> <p>1 – MII 100Mb mode – 5.12us, MII 10Mb mode – 51.2us</p> <p>0 – MII 100Mb mode – 81.92us, MII 10Mb mode – 819.2us</p>
30:27	RW	0xF	<p>Transmit timer (TT)</p> <p>Controls the maximum time that must elapse between the end of a transmit operation and and setting the CSR5.TI (Transmit Interrupt) bit. This time is equal to TT * (16*CS).</p> <p>The transmit timer is enabled when written with nonzero value. After each frame transmission the timer starts to count down if it has not already started. It is reloaded after every frame transmitted.</p> <p>Writing '0' to this field disables the timer effect on the transmit interrupt mitigation mechanism.</p> <p>Reading this field gives the actual count value of the timer.</p>
26:24	RW	0x7	<p>Number of transmit packets (NTP)</p> <p>Controls the maximum number of the frames transmitted before setting the CSR5.TI (Transmit Interrupt) bit.</p> <p>The transmit counter is enabled when written with nonzero value. It is decremented after every frame transmitted. It is reloaded after setting the CSR5.TI (Transmit Interrupt) bit</p> <p>Writing '0' to this field disables the counter effect on the transmit interrupt mitigation mechanism.</p> <p>Reading this field gives the actual count value of the counter.</p>
23:20	RW	0xF	<p>Receive timer (RT)</p> <p>Controls the maximum time that must elapse between the end of a receive operation and setting the CSR5.RI (Receive Interrupt) bit.</p> <p>This time is equal to RT * CS.</p> <p>The receive timer is enabled when written with nonzero value. After each frame reception the timer starts to count down if it has not already started. It is reloaded after every frame received.</p> <p>Writing '0' to this field disables the timer effect on the receive interrupt mitigation mechanism.</p> <p>Reading this field gives the actual count value of the timer.</p>
19:17	RW	0x7	<p>Number of receive packets (NRP)</p> <p>Controls the maximum number of the received frames before setting the CSR5.RI (Receive Interrupt) bit.</p> <p>The receive counter is enabled when written with nonzero</p>

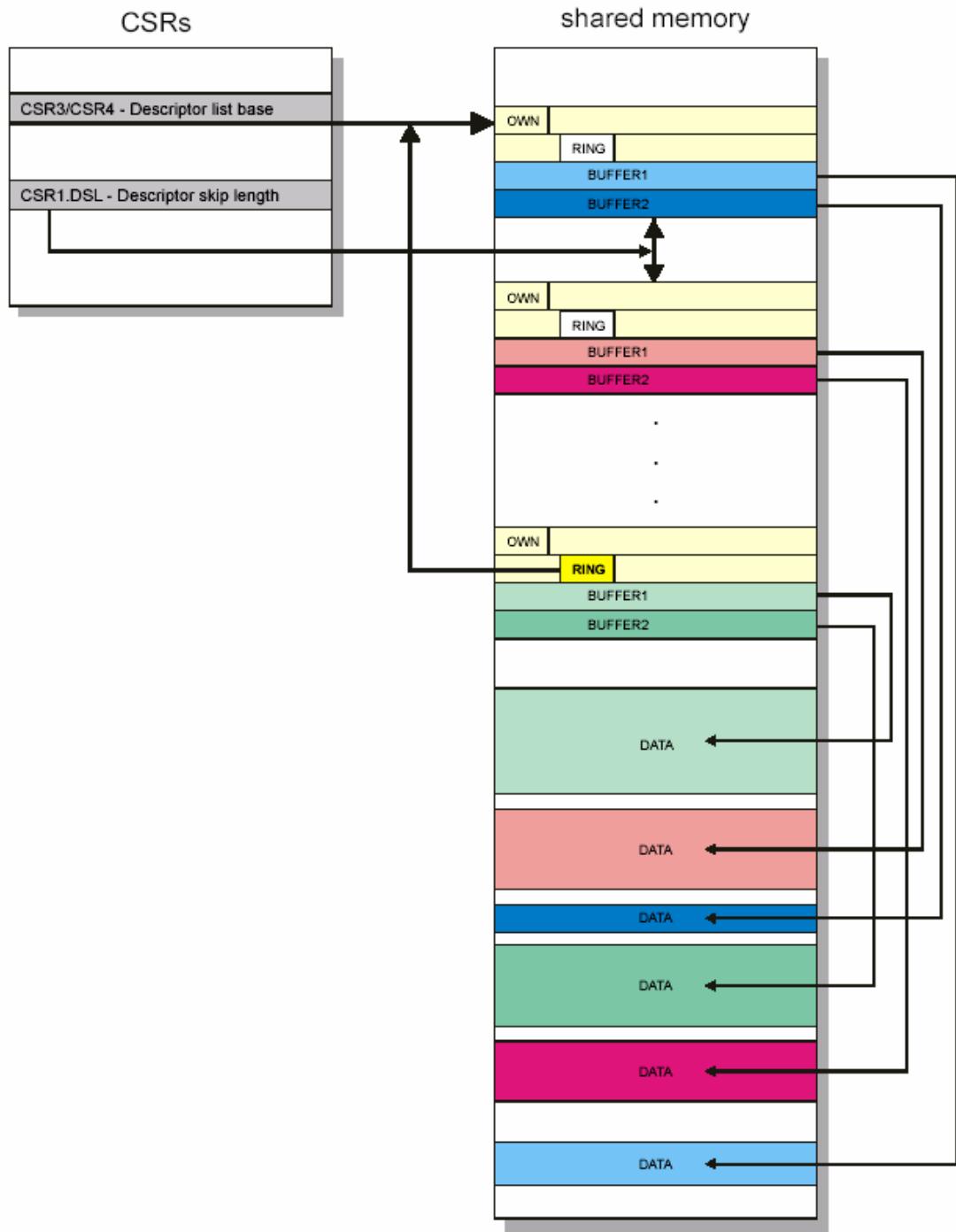
			value. It is decremented after every frame received. It is reloaded after setting the CSR5.RI (Receive Interrupt) bit. Writing '0' to this field disables the timer effect on the receive interrupt mitigation mechanism. Reading this field gives the actual count value of the counter.
16	RW	0x0	Continuous mode (CON) 1 – general-purpose timer works in continuous mode 0 – general-purpose timer works in one-shot mode
15:0	RW	0x0	Timer value (TIM) Contains the number of iterations of the general-purpose timer. Each iteration duration is MII 100Mb mode – 81.92us MII 10Mb mode – 819.2us

Notes: Attr: RW – Read/writable, R – Read only, W – Write only

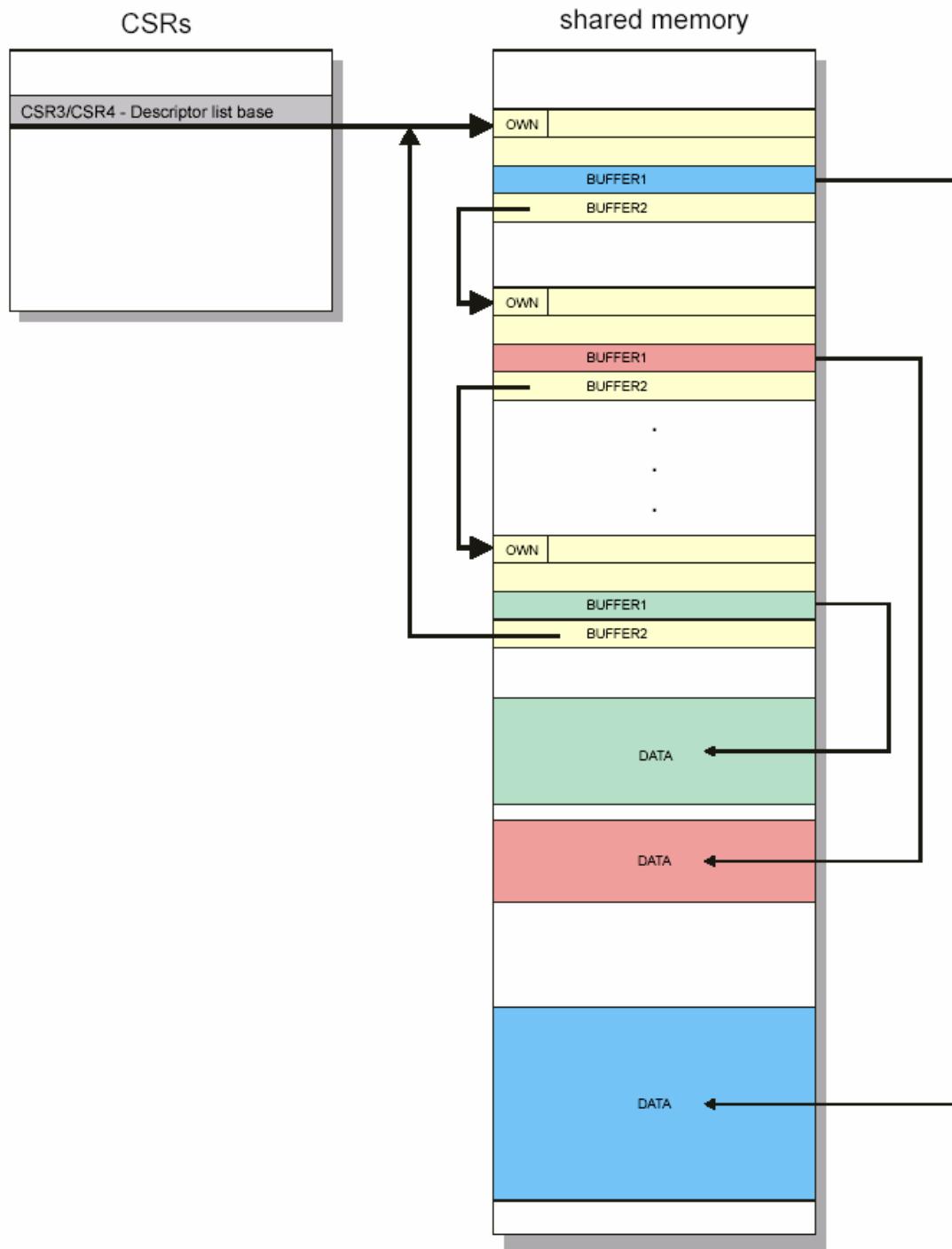
Functional Description

Descriptors/buffers architecture overview

A data exchange between the host and the MAC is performed via the descriptor lists and data buffers, which reside in the system shared RAM. The buffers hold the host data to be transmitted or received by the MAC. The descriptors act as pointers to these buffers. Each descriptor list should be constructed by the host in a shared memory area, and can be of an arbitrary size. There is a separate list of descriptors for both the transmit and receive processes. The position of the first descriptor in the descriptor list is described by MAC_CSR3 for the receive list, and by MAC_CSR4 for the transmit list. The descriptors can be arranged in either a "chained" or "ring" structure. In a chained structure every descriptor contains a pointer to the next descriptor in the list. In the ring structure the address of the next descriptor is determined by the MAC_CSR0.6,2 (DSL - Descriptor Skip Length). Every descriptor can point to up to two data buffers. When using descriptor chaining the address of the second buffer is used as a pointer to the next descriptor, and thus only one buffer is available. A frame can occupy one or more data descriptors and buffers, but one descriptor cannot exceed a single frame.



Descriptors in "ring" structure

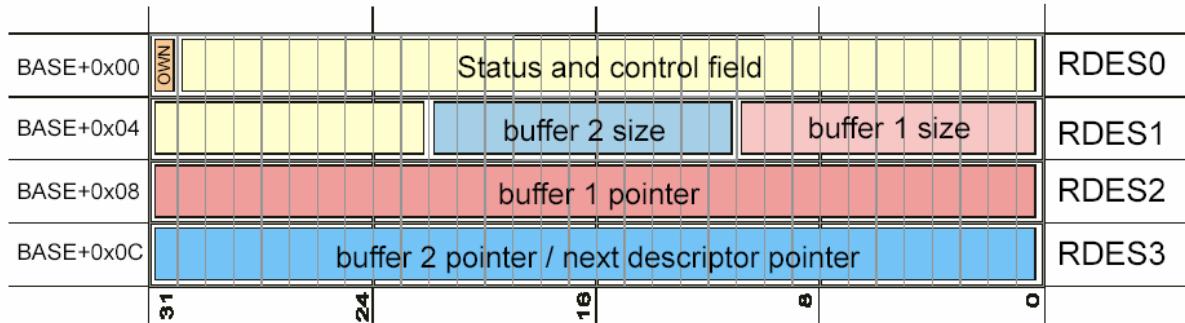


Descriptors in "chain" structure

Receive Descriptors

Each receive descriptor consists of 4 32-bit fields named RDES0, RDES1, RDES2 and RDES3

The receive descriptor fields are described below.



Receive descriptor status field (RDES0)

Bit	Symbol	Function
RDES0[31]	OWN	Ownership bit. 1 – the MAC owns the descriptor. 0 – the host owns the descriptor. The MAC will clear this bit when it completes a current frame reception, or when the data buffers associated with a given descriptor are already full.
RDES0[30]	FF	Filtering fail. When set, indicates that a received frame did not pass the address recognition process. This bit is valid only for the last descriptor of the frame (RDES0.8 set), when the CSR6.30 (receive all) bit is set, and the frame is at least 64 bytes long.
RDES0[29:16]	FL	Frame length. Indicates the length, in bytes, of the data transferred into a host memory for a given frame. This bit is valid only when the RDES0.8 (last descriptor) is set and RDES0.14 (descriptor error) is cleared.
RDES0[15]	ES	Error summary. This bit is logical or over following bits: RDES0.1 – CRC error. RDES0.6 – Collision seen. RDES0.7 – Frame too long. RDES0.11 – Runt frame. RDES0.14 – Descriptor error. This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[14]	DE	Descriptor error. Set by the MAC when no receive buffer was available when trying to store the received data. This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[13:12]	-	Reserved

RDES0[11]	RF	Runt frame. When set, indicates that the frame is damaged by a collision or by a premature termination before the end of a collision window. This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[10]	MF	Multicast frame. When set, indicates that the frame has a multicast address. This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[9]	FS	First descriptor. When set, indicates that this is the first descriptor of a frame.
RDES0[8]	LS	Last descriptor. When set, indicates that this is the last descriptor of a frame.
RDES0[7]	TL	Frame too long. When set, indicates that a current frame is longer than the maximum size of 1518 bytes, as specified by 802.3. This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[6]	CS	Collision seen. When set, indicates that a late collision was seen (collision after 64 bytes following SFD). This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[5]	FT	Frame type. When set, indicates that the frame has the length field greater than 1500 (Ethernet type frame). When cleared, indicates the 802.3 type frame. This bit is valid only when the RDES0.8 (last descriptor) is set. Additionally, the FT is invalid for the runt frames of a length shorter than 14 bytes.
RDES0[4]	-	Reserved
RDES0[3]	RE	Report on MII error. When set, indicates that an error has been detected by a physical layer chip connected through the MII interface. This bit is valid only when the RDES0.8 (last descriptor) is set.
RDES0[2]	DB	Dribbling bit. When set, indicates that the frame was not byte aligned. This bit is valid only when the RDES0.8 (last descriptor) is set.

RDES0[1]	CE	CRC error. When set, indicates that the CRC error has occurred in the received frame. This bit is valid only when the RDES0.8 (last descriptor) is set. Additionally, the CE is not valid when the received frame is a runt frame.
RDES0[0]	ZERO	This bit is reset for the frames with the legal length.

Receive descriptor control and count field (RDES1)

Bit	Symbol	Function
RDES1[31:26]	-	Reserved
RDES1[25]	RER	Receive end of ring. When set, indicates that this is the last descriptor in the receive descriptor ring. The MAC returns to the first descriptor in the ring, as specified by the CSR3 (Start of receive list address).
RDES1[24]	RCH	Second address chained. When set, indicates that the second buffer's address points to the next descriptor and not to the data buffer. Note that the RER takes precedence over the RCH.
RDES1[23]	-	Reserved
RDES1[21:11]	RBS2	Buffer 2 size. Indicates size, in bytes, of memory space used by the second data buffer. This number must be a multiple of 4. If it is 0, then the MAC ignores the second data buffer and fetches the next data descriptor. This number is valid only when the RDES1.24 (Second address chained) is cleared.
RDES1[10:0]	RBS1	Buffer 1 size. Indicates the size, in bytes, of memory space used by the first data buffer. This number must be a multiple of 4. If it is 0, then the MAC ignores the first data buffer and uses the second data buffer.

Receive descriptor buffer address 1 field (RDES2) bit functions

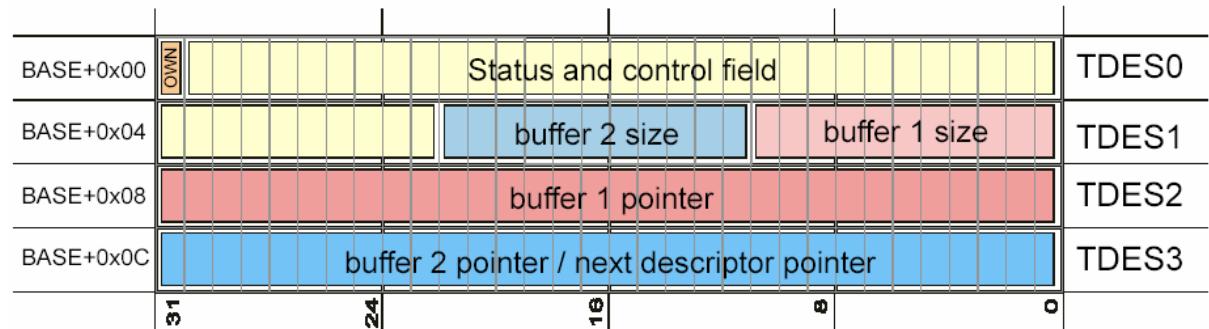
Bit	Symbol	Function
RDES2[31:0]	RBA1	Receive buffer 1 address. It Indicates the address of memory allocated for the first receive buffer. This number must be longword aligned (RDES2.1..0 = 00).

Receive descriptor buffer address 2 field (RDES3) bit functions

Bit	Symbol	Function
RDES3[31:0]	RBA2	Receive buffer 2 address. It indicates the address of memory allocated for the second receive buffer, if RDES1.24 (RCH) bit is cleared. If RDES1.24 bit is set, RBA1 points to the next descriptor and not to the data buffer. This number must be longword aligned (RDES3.1..0 = 00).

Transmit Descriptors

Each transmit descriptor consist of 4 32-bit fields named TDES0, TDES1, TDES2 and TDES3. The receive descriptor fields are described below.



Transmit descriptor status field (TDES0)

Bit	Symbol	Function
TDES0[31]	OWN	Ownership bit. 1 – the MAC owns the descriptor. 0 – the host owns the descriptor. The MAC will clear this bit when it completes a current frame transmission or when the data buffers associated with a given descriptor are empty.
TDES0[30:16]	-	Reserved
TDES0[15]	ES	Error summary. This bit is a logical or of the following bits: TDES0.1 – Underflow error. TDES0.8 – Excessive collision error. TDES0.9 – Late collision. TDES0.10 – No carrier. TDES0.11 – Loss of carrier. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[14:12]	-	Reserved
TDES0[11]	LO	Loss of carrier. When set, indicates a loss of the carrier during a

		transmission. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[10]	NC	No carrier. When set, indicates that the carrier was not asserted by an external transceiver during the transmission. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[9]	LC	Late collision. When set, indicates that a collision was detected after transmitting 64 bytes. This bit is not valid when TDES0.1 (underflow error) is set. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[8]	EC	Excessive collisions. When set, indicates that the transmission was aborted after 16 retries. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[7]	-	Reserved
TDES0[6:3]	CC	Collision count. This field indicates the number of collisions that occurred before the end of a frame transmission. This value is not valid when TDES0.8 (Excessive collisions bit) is set. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[2]	-	Reserved
TDES0[1]	UF	Underflow error. When set, indicates that the FIFO was empty during the frame transmission. This bit is valid only when TDES1.30 (last descriptor) is set.
TDES0[0]	DE	Deferred. When set, indicates that the frame was deferred before transmission. Deferring occurs if the carrier is detected when the transmission is ready to start. This bit is valid only when TDES1.30 (last descriptor) is set.

Transmit descriptor control and count field (TDES1)

Bit	Symbol	Function
TDES1[31]	IC	Interrupt on completion. Setting this flag instructs the MAC to set CSR5.0

		(transmit interrupt) immediately after processing a current frame. This bit is valid when TDES1.30 (last descriptor) is set, or for a setup packet.
TDES1[30]	LS	Last descriptor. When set, indicates the last descriptor of the frame.
TDES1[29]	FS	First descriptor. When set, indicates the first descriptor of the frame.
TDES1[28]	FT1	Filtering type. This bit together with TDES1.22 (FT0) controls a current filtering mode. This bit is valid only for the setup frames.
TDES1[27]	SET	Setup packet. When set, indicates that this is a setup frame descriptor.
TDES1[26]	AC	Add CRC disable. When set, the MAC does not append the CRC value at the end of the frame. The exception is when the frame is shorter than 64 bytes and automatic byte padding is enabled. In that case the CRC field is added despite the state of the AC flag.
TDES1[25]	TER	Transmit end of ring. When set, indicates the last descriptor in the descriptors ring.
TDES1[24]	TCH	Second address chained. When set, indicates that the second descriptor's address points to the next descriptor and not to the data buffer. This bit is valid only when TDES1.25 (transmit end of ring) is reset.
TDES1[23]	DPD	Disabled padding. When set, automatic byte padding is disabled. The MAC normally appends the PAD field after the INFO field when the size of a frame is less than 64 bytes. After padding bytes, the CRC field is also inserted regardless of the state of the AC flag. When the DPD is set, no padding bytes are appended.
TDES1[22]	FT0	Filtering type. This bit together with TDES1.28 (FT1), controls the current filtering mode. This bit is valid only when TDES1.27 (SET) bit is set.
TDES1[21:11]	TB2	Buffer 2 size. Indicates the size, in bytes, of memory space used by the second data buffer. If it is zero, the MAC ignores the second data buffer and fetches the next data descriptor. This bit is valid only when TDES1.24 (second address chained) is cleared.

TDES1[10:0]	TB1	Buffer 1 size. Indicates the size, in bytes, of memory space used by the first data buffer. If it is 0, the MAC ignores the first data buffer and uses the second data buffer.
-------------	-----	---

Transmit descriptor buffer address 1 field (TDES2) bit functions

Bit	Symbol	Function
TDES2[31:0]	TBA1	Transmit buffer 1 address. Contains the address of the first data buffer. For the setup frame this address must be longword aligned (TDES3.1..0 = 00). In all other cases there are no restrictions on buffer alignment.

Transmit descriptor buffer address 2 field (TDES3) bit functions

Bit	Symbol	Function
TDES3[31:0]	TBA2	Transmit buffer 2 address. Contains the address of the second data buffer. There are no restrictions on buffer alignment.

Setup Frames

The setup frames define the addresses which are used for the receive address filtering process. These frames are never transmitted on the Ethernet. They are used to fill the address filtering RAM. A valid setup frame must be exactly 192 bytes long, and must be allocated in a single buffer that is long-word aligned. Also, both TDES1.29 (First descriptor) and TDES1.30 (Last descriptor) must be cleared for the setup frame. The FT1 and FT0 bits of the setup frame define the current filtering mode.

Table below lists all possible combinations.

TDES1[22] FT0	TDES1[28] FT1	Description
0	0	Perfect filtering mode. Setup frame buffer is interpreted as a set of 16 48-bit physical addresses.
0	1	Hash filtering mode. Setup frame buffer contains 512-bit hash table plus a single 48-bit physical address.
1	0	Inverse filtering mode. Setup frame buffer is interpreted as a set of 16 48-bit physical addresses.
1	1	Hash only filtering mode. Setup frame buffer is interpreted as a 512-bit hash table.

Table below shows the setup frame buffer format for perfect filtering modes.

Note that “xxxxxxxxxxxxxxxx” represents “Don’t Care” bits.

Byte#	Data bits [31:16]	Data bits [15:0]
3:0	XXXXXXXXXXXXXXXXXX	Physical Address 0 [15:00]
7:4	XXXXXXXXXXXXXXXXXX	Physical Address 0 [31:16]
11:8	XXXXXXXXXXXXXXXXXX	Physical Address 0 [47:32]
15:12	XXXXXXXXXXXXXXXXXX	Physical Address 1 [15:00]
19:16	XXXXXXXXXXXXXXXXXX	Physical Address 1 [31:16]
23:20	XXXXXXXXXXXXXXXXXX	Physical Address 1 [47:32]
...
...
171:168	XXXXXXXXXXXXXXXXXX	Physical Address 14 [15:00]
175:172	XXXXXXXXXXXXXXXXXX	Physical Address 14 [31:16]
179:176	XXXXXXXXXXXXXXXXXX	Physical Address 14 [47:32]
183:180	XXXXXXXXXXXXXXXXXX	Physical Address 15 [15:00]
187:184	XXXXXXXXXXXXXXXXXX	Physical Address 15 [31:16]
191:188	XXXXXXXXXXXXXXXXXX	Physical Address 15 [47:32]

Table below shows the setup frame buffer format for imperfect filtering modes.

Byte#	Data bits [31:16]	Data bits [15:0]
3:0	XXXXXXXXXXXXXXXXXX	Hash Filter [015:000]
7:4	XXXXXXXXXXXXXXXXXX	Hash Filter [031:016]
11:8	XXXXXXXXXXXXXXXXXX	Hash Filter [047:032]
...
...
123:120	XXXXXXXXXXXXXXXXXX	Hash Filter [495:480]
127:124	XXXXXXXXXXXXXXXXXX	Hash Filter [511:496]
131:128	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
135:132	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
...	...	
...	...	
159:156	XXXXXXXXXXXXXXXXXX	Physical Address [15:00]
163:160	XXXXXXXXXXXXXXXXXX	Physical Address [31:16]
167:164	XXXXXXXXXXXXXXXXXX	Physical Address [47:32]
171:168	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
175:172	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
...	...	
...	...	
183:180	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
187:184	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
191:188	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	

DMA Controllers

The DMA (Direct Memory Access) is used to control a data flow between the host and the MAC.

The DMA services the following types of requests from the MAC transmit and receive processes:

- Transmit request:
- Descriptor fetch
- Descriptor closing
- Setup packet processing
- Data transfer from host buffer to transmit the FIFO
- Receive request
- Descriptor fetch
- Descriptor closing
- Data transfer from receive the FIFO to the host buffer

The key task for the DMA is to perform an arbitration between receive and transmit processes. Two arbitration schemes are possible, selected by the CSR0.1 (BAR) bit:

1 – round-robin arbitration scheme in which two processes have equal priorities.
With round robin arbitration scheme DMA accesses can look like: T-R-T-R-T-R, where T denotes DMA access for transmit process and R denotes DMA access for receive process.

0 – the receive process has priority over the transmit process unless transmission is in progress.

In this case the following rules apply:

- When the transmission is not in progress the transmit process request should be serviced by the DMA between two consecutive receive transfers

In this case DMA accesses can look like: T-R-R-T-R-R-T.

- When the transmission is in progress the receive process request should be serviced by the DMA between two consecutive transmit transfers.

In this case DMA accesses can look like: R-T-T-R-T-T-R.

Transfers between the host and the MAC performed by the DMA component are either single data transfers or "burst" transfers.

Descriptors are always accessed with single transfers of 32 bit size regardless of the direction of transfer.

The data buffers are transferred using burst transfers. The burst length is defined by CSR0(13..8) (Programmable burst length), and can be set to 0, 1, 2, 4, 8, 16, or 32. When set to 0, no maximum burst size is defined and the transfer ends when the transmit FIFOs are full or the receive FIFOs are empty to transfer remaining data.

Transmit Process

The transmit process can operate in one of three modes: running, stopped or suspended. After a software or hardware reset, or after a stop transmit command, the transmit process is in a stopped state. The transmit process can leave a stopped state only after the start transmit command.

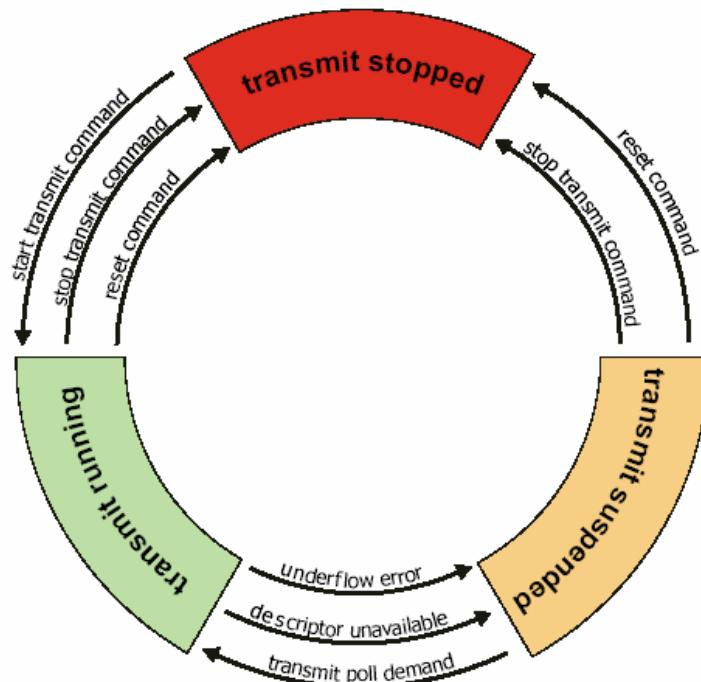
When in a running state the transmit process performs descriptor/buffer processing. When operating in a suspended or stopped state the transmit process retains the position of the next descriptor i.e., the address of the descriptor following the last descriptor being closed. After entering a running state, that position is used for the next descriptor fetch. The only exception is when the host writes the transmit descriptor base address register (CSR4). In that case the descriptor address is reset and the fetch is directed to the first position in the list.

The transmit process remains running until one of the following events occur:

- The hardware or software reset is issued. Setting the CSR0.0 (SWR) bit initiates the software reset. After the reset, all internal registers return to their default states. The current descriptor's position in the transmit descriptors list is lost.
- A stop transmit command is issued by the host. This can be performed by writing 0 to the CSR6.13 (ST) bit. The current descriptor's position is retained.
- The descriptor owned by the host is found. A current descriptor's position is retained.
- The transmit FIFO underflow error is detected. An underflow error is generated when the transmit FIFO is empty during the transmission of the frame. When it occurs, the transmit process enters a suspended state. Transmit automatic polling is internally disabled, even if it is enabled by the host by writing the TAP bits. The current descriptor's position is retained.

Leaving a suspended state is possible in one of the following situations:

- A transmit poll demand command is issued. This can be performed by writing CSR1 with nonzero value. The transmit poll demand command can be also generated automatically when Transmit automatic polling is enabled. Transmit automatic polling is enabled only if the CSR0(19..17) (TAP) bits are written with a nonzero value, and when there was no underflow error prior to entering the suspended state.
- A stop transmit command is issued by the host. This can be performed by writing a 0 to the CSR6.13 (ST) bit. The current descriptor's position is retained.



Receive Process

The receive process can operate in one of three modes: running, stopped or suspended. After a software or hardware reset, or after a stop receive command, the receive process is in the stopped state. The receive process can leave a stopped state only after a start receive command.

In the running state, the receiver performs descriptor/buffer processing. In the running state the receiver fetches from the receive descriptor list. It performs this fetch regardless of whether there is any frame on the link. When there is no frame pending, the receive process reads the descriptor and simply waits for the frames. When a valid frame is recognized, the receive process starts to fill the memory buffers pointed to by the current descriptor. When the frame ends, or when the memory buffers are completely filled, the current frame descriptor is closed (ownership bit cleared). Immediately, the next descriptor on the list is fetched in the same manner, and so on.

When operating in a suspended or stopped state, the receive process retains the position of the next descriptor i.e., the address of the descriptor following the last descriptor being closed. After entering a running state, that position is used for the next descriptor fetch. The only exception is when the host writes the receive descriptor base address register (CSR3). In that case the descriptor address is reset and the fetch is pointed to the first position in the list.

The receive process remains running until one of the following events occur:

- A hardware or software reset is issued by the host. a software reset can be performed by setting the CSR0.0 (SWR) bit. After reset, all internal registers return to their default states.

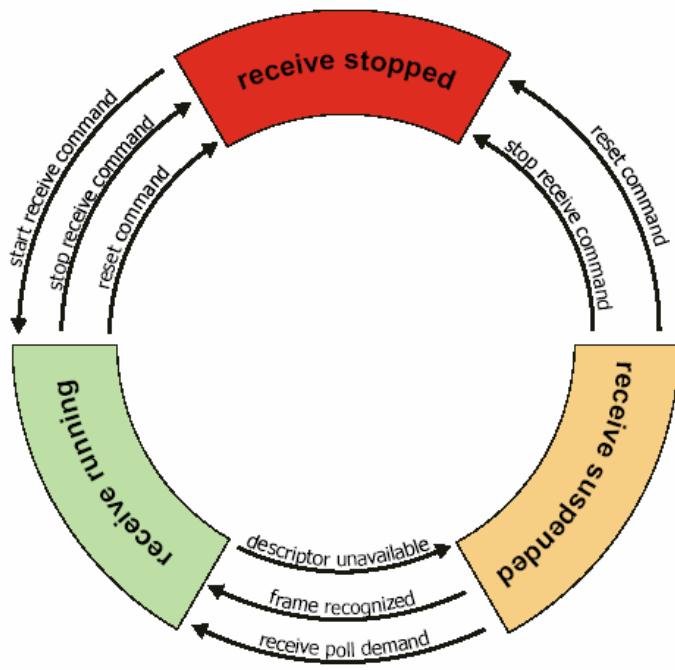
The current descriptor's position in the receive descriptors list is lost.

- A stop receive command is issued by the host. This can be performed by writing a 0 to the CSR6.1 (SR) bit. The current descriptor's position is retained.
- The descriptor owned by the host is found by the MAC during the descriptor fetch.

The current descriptor's position is retained.

Leaving a suspended state is possible in one of the following situations:

- A receive poll command is issued by the host. This can be performed by writing a nonzero value to CSR2.
- A new frame is detected by the MAC on a receive link.
- A stop receive command is issued by the host. This can be performed by writing a 0 to the CSR6.1 (SR) bit. The current descriptor's position is retained.



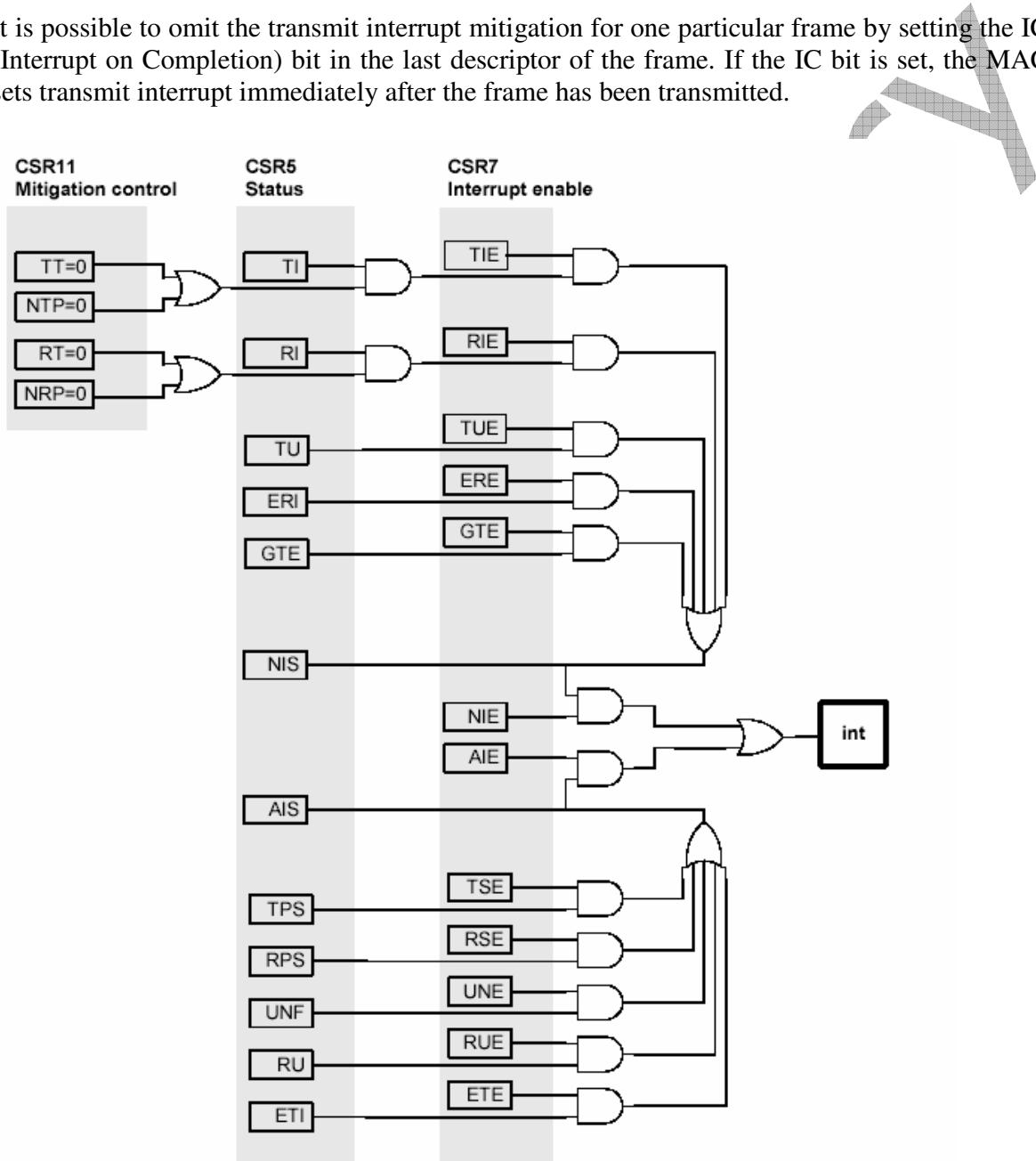
Interrupt Control

The interrupt controller uses three internal Control and Status Registers: CSR5, CSR7 and CSR11. The CSR5 contains the MAC's status information. It has 10 bits that can trigger an interrupt. These bits are collected in two groups: normal interrupts, and abnormal interrupts. Each group has its own summary bit, the NIS and AIS, respectively. The NIS and AIS bits directly control the int pin of the MAC. Every status bit in the CSR5 that can source an interrupt can be individually masked by writing an appropriate value into the CSR7 - Interrupt Enable Register.

An interrupt mitigation mechanism is provided for reducing the CPU usage in servicing interrupts. The interrupt mitigation is controlled via the CSR11. There are separate interrupt mitigation control blocks for the transmit and receive interrupts. Both of these blocks consist

of a 4-bit frame counter and a 4-bit timer. The operation of these blocks is similar for the receive and transmit processes. After the end of a successful receive or transmit operation, an appropriate counter is decremented and the timer starts to count down, if it has not already started. An interrupt is triggered when either the counter or timer reaches zero. This provides the possibility to generate a single interrupt for a few received/transmitted frames, or after a specified time from the last successful receive/transmit operation.

It is possible to omit the transmit interrupt mitigation for one particular frame by setting the IC (Interrupt on Completion) bit in the last descriptor of the frame. If the IC bit is set, the MAC sets transmit interrupt immediately after the frame has been transmitted.



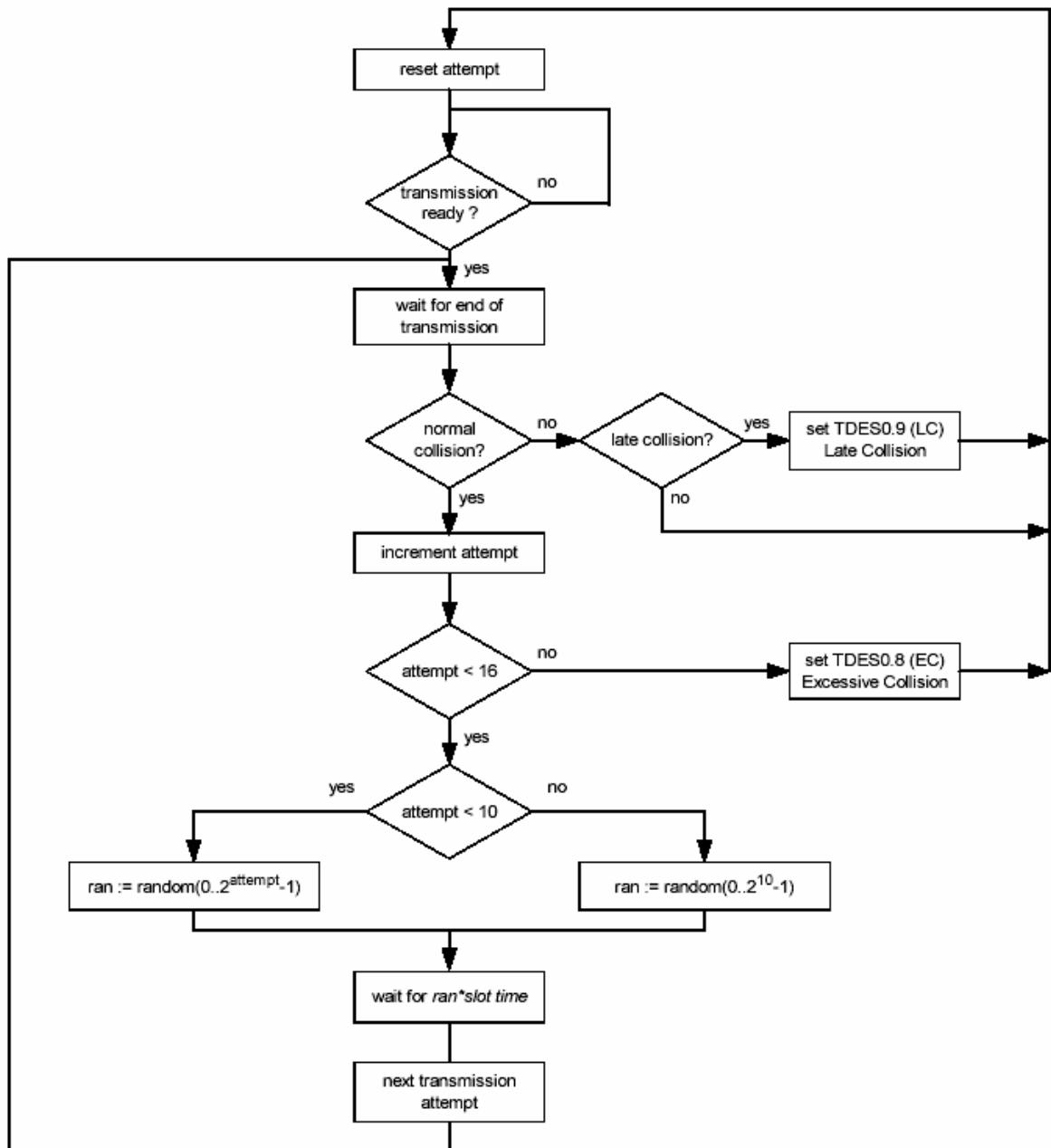
Collision Handling

Collision detection is performed via the col input pin. If a collision is detected before the end

of the PREAMBLE / SFD, then the MAC completes the PREAMBLE / SFD, transmits the JAM sequence, and then initiates a backoff computation. If a collision is detected after the transmission of the PREAMBLE and SFD, but prior to 512 bits being transmitted, the MAC immediately aborts the transmission, transmits the JAM sequence, and then initiates a backoff. If a collision is detected after 4096 bits have been transmitted, the collision is termed a late collision. The MAC aborts the transmission and appends the JAM sequence. The transmit message is flushed from the FIFO. The MAC does not initiate a backoff and does not attempt to retransmit the frame when a late collision is detected.

The MAC uses a "truncated binary exponential backoff" algorithm for backoff computing, as defined in the IEEE 802.3 standard, and outlined on figure below. Backoff processing is performed only in a half duplex mode. In full duplex mode, collision detection is disabled.

preliminary!

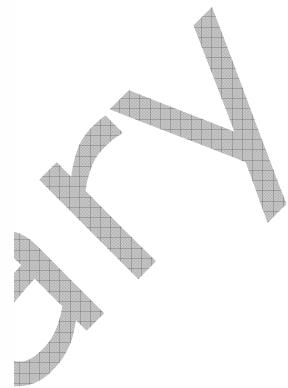
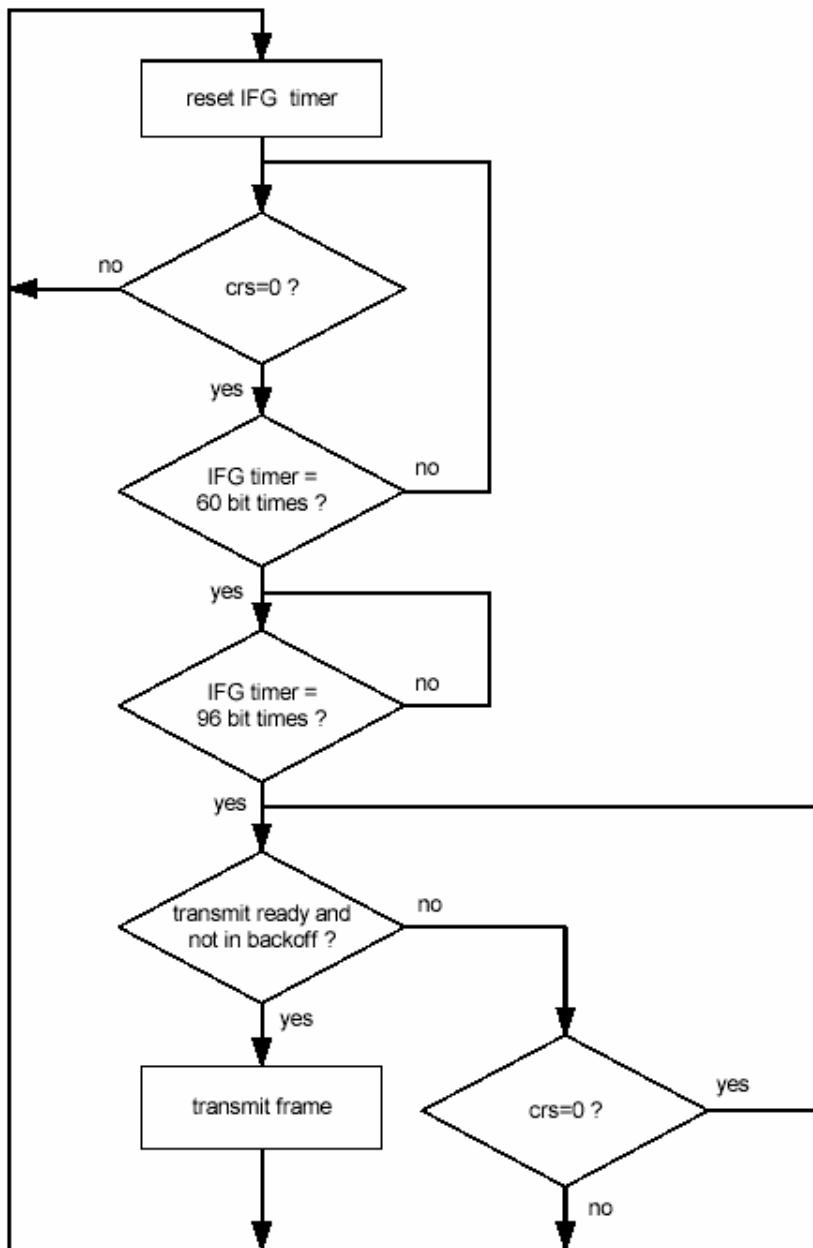


Deferring

The deference algorithm is implemented as required by the 802.3 specification and outlined on figure below. The IFG (InterFrame Gap) timer starts to count whenever the link is not idle. If the activity on the link is detected during the first 60 bit times of the IFG timer, the timer is reset, and restarted once activity has stopped. During the final 36 bit times of the IFG timer, the link activity is ignored.

Carrier sensing is performed only when operating in half duplex mode. In full duplex mode,

the state of the crs input is ignored.

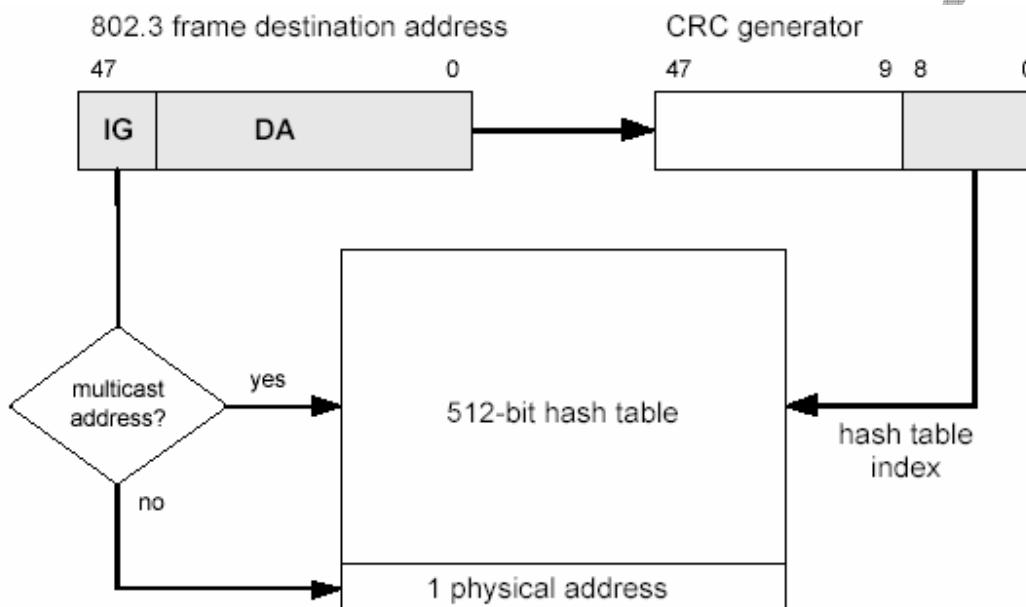


Receive address filtering

There are three kinds of addresses on the LAN: the unicast addresses, the multicast addresses, and the broadcast addresses. If the first bit of the address (IG bit) is 0, the frame is unicast i.e., dedicated to a single station. If the first bit is 1, the frame is multicast i.e., destined for a group of stations. If the address field contains all 1's, the frame is broadcast and should be received by all stations on the LAN.

When the MAC operates in perfect filtering mode, all frames are checked against the addresses in the address filtering RAM. The unicast, multicast, and broadcast frames are treated in the same manner.

When the MAC operates in the imperfect filtering mode, the frames with the unicast addresses are checked against a single physical address. The multicast frames are checked using the 512-bit hash table. The MAC applies the standard Ethernet CRC function to the first 6 bytes of the frame, containing a destination address. The least significant 9 bits of the CRC value are used to index the table. If the indexed bit is set, the frame is accepted. If this bit is cleared, the frame is rejected. The algorithm is shown on figure below.



It is important that one bit in the hash table can correspond to many Ethernet addresses. Therefore it is possible that some frames can be accepted by the MAC, even if they are not intended to be received. This is because some frames that should not have been received have addresses that hash to the same bit in the table as one of the proper addresses. The software should perform additional address filtering to reject all such frames.

General purpose timer

A 16-bit general-purpose timer is provided to simplify host calculated time intervals. The timer operates synchronously with the transmit clock `clkt` generated by the PHY device. This gives the host the possibility of measuring the time intervals based on actual Ethernet bit time.

The timer can operate in a "one shot" mode or a continuous mode. In a "one shot" mode the timer stops after reaching zero, and in a continuous mode, upon reaching zero, the timer is automatically reloaded and continues counting.

The actual count value can be tested with an accuracy of ± 1 bit by reading CSR11[15:0]. When writing to CSR11[15:0], the data are stored in the internal reload register. The timer is immediately reloaded and starts to count down.

preliminary

Chapter 13 USB2.0 HS OTG

Overview

USB is a popular standard for connecting peripherals and portable consumer electronic devices such as digital cameras and hand-held computers to host PCs. USB OTG technology consumer electronics, peripherals and portable devices can connect to each other (for example, a digital camera can connect directly to a printer, or a keyboard can connect to a Personal Digital Assistant) to exchange data.

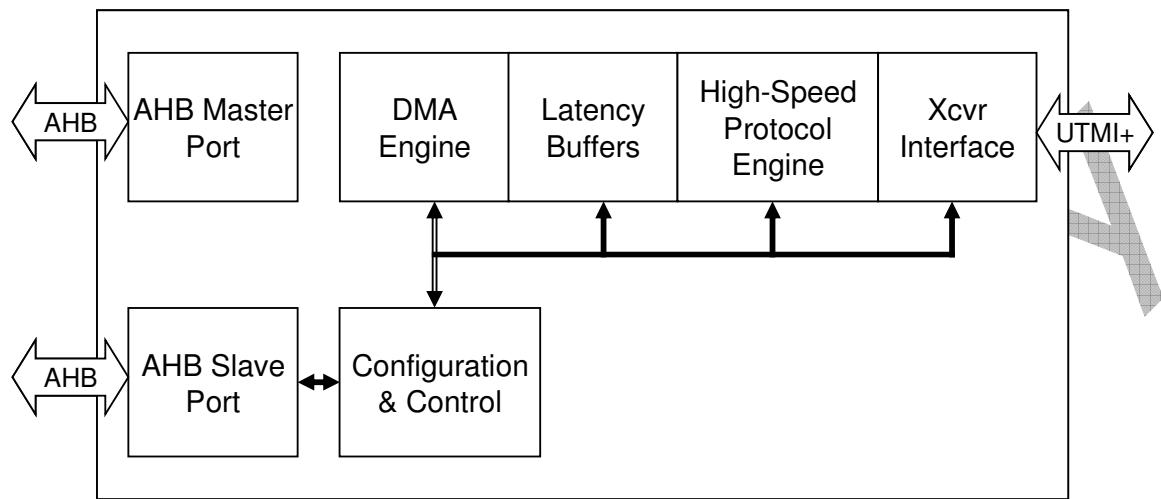
The USB2.0 HS OTG is designed to make efficient use of the system resources in an SOC design. The 32-bit system bus interface contains a chaining DMA engine that reduces the interrupt load on the application processor, and reduces the total system bus bandwidth that must be dedicated to servicing the USB interface requirements. By transferring the data to system memory at wire rates, the buffer memory requirement within the core is minimized.

Key Features

- Complies with the OTG specification supplement to the USB 2.0 protocol
- Compliant with INTEL EHCI-TT specification in host mode
- Programs as USB2.0 device only, USB2.0 host only, and OTG
- Supports high speed (480Mbps), full speed (12Mbps) and low-speed (1.5Mbps)
- 4 endpoints for USB device

Architecture

Block Diagram



Block Descriptions

AHB Master/Slave Port

AHB master and slave provide an interface to connect OTG functionality with AHB bus. CPU could program OTG registers through AHB slave port. DMA of OTG could access system memory through AHB master port.

Xcvr Interface

Xcvr interface provide UTMI+ signaling to transmit and receive data to PHY.

DMA Engine

DMA controller has state machines that are able to parse all of the data structures defined in this controller specification. In host mode, the data structures are from the EHCI specification and represent queues of transfers to be performed by the host controller, including the split transaction requests that allow an EHCI controller to direct packets to Low and Full speed devices. In device mode, the data structures designed to be similar to those in the EHCI specification are used to allow device responses to be queued for each of the active pipes in the device.

High-Speed Protocol Engine

Protocol Engine parses all the USB tokens and generates the response packets. It is responsible for all error checking, check field generation, formatting all the handshake, ping and data response packets on the bus, and for any signals that must be generated based on a USB based

timeframe. In host mode, the Protocol engine also generates all of the token packets required by the USB protocol.

Registers

Registers Summary

Identification Registers

Name	Offset	Size	Reset Value	Description
OTG_GPT0LD	0x0080	W	0x00000000	General purpose timer0 load register
OTG_GPT0CTRL	0x0084	W	0x00000000	General purpose timer0 control register
OTG_GPT1LD	0x0088	W	0x00000000	General purpose timer1 load register
OTG_GPT1CTRL	0x008C	W	0x00000000	General purpose timer1 control register
OTG_SBUSCFG	0x0090	W	0x00000007	System bus configuration register

Host/Device Capability Registers

Name	Offset	Size	Reset Value	Description
OTG_CAPLEN	0x0100	W	0x00000040	Capability register length
OTG_HCIVER	0x0102	W	0x00000100	Host interface version number
OTG_HCSPAR	0x0104	W	0x00010011	Host controller structural parameter
OTG_HCCPAR	0x0108	W	0x00000006	Host controller capability parameter
OTG_DCIVER	0x0120	W	0x00000001	Device interface version number
OTG_DCCPAR	0x0124	W	0x00000184	Device controller capability parameter

Host/Device Operational Registers

Name	Offset	Size	Reset Value	Description
OTG_USBCMD	0x0140	W	0x00080000	USB command register
OTG_USBSTS	0x0144	W	0x00000000	USB status register
OTG_USBINTR	0x0148	W	0x00000000	USB interrupt enable
OTG_FRINDEX	0x014C	W	0x00000000	USB frame index
OTG_FLBADDR	0x0154	W	0x00000000	Period list base address / Device address
OTG_NALADDR	0x0158	W	0x00000000	Next async. list address / Endp list address
OTG_TTCTRL	0x015C	W	0x00000000	TT control and status register
OTG_BURSTSZ	0x0160	W	0x00001010	Burst size register
OTG_TXTUNE	0x0164	W	0x00000000	Host transmit pre-buffer package

				tuning register
OTG_TXTTUNE	0x0168	W	0x00000000	Host TT transmit pre-buffer package tuning register
OTG_EPNAK	0x0178	W	0x00000000	Endpoint NAK register
OTG_EPNAKEN	0x017C	W	0x00000000	Endpoint NAK enable register
OTG_PORTSC	0x0184	W	0x10000000	Port control and status
OTG_OTGSC	0x01A4	W	0x00000020	OTG status and control
OTG_USBMODE	0x01A8	W	0x00000000	USB device mode register
OTG_EPSPSTAT	0x01AC	W	0x00000000	Endpoint setup status
OTG_EPPRIME	0x01B0	W	0x00000000	Endpoint initialization
OTG_EPFLUSH	0x01B4	W	0x00000000	Endpoint de-initialization
OTG_EPSTAT	0x01B8	W	0x00000000	Endpoint status register
OTG_EPCOMP	0x01BC	W	0x00000000	Endpoint complete register
OTG_EPCTRL0	0x01C0	W	0x00800080	Endpoint control 0 register
OTG_EPCTRL1	0x01C4	W	0x00000000	Endpoint control 1 register
OTG_EPCTRL2	0x01C8	W	0x00000000	Endpoint control 2 register
OTG_EPCTRL3	0x01CC	W	0x00000000	Endpoint control 3 register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

OTG_GPTxLD (x=0,1)

Address: Operational Base + offset (0x0080/0x0088)

General purpose timerx load register

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	RW	0x0	<p>GPTLD; General Purpose Timer Load Value.</p> <p>This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration.</p> <p>Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7.</p> <p>Note: Max value is 0xFFFFFFF or 16.777215 seconds.</p>

OTG_GPTxCTRL (x=0,1)

Address: Operational Base + offset (0x0084/0x008C)

General purpose timerx control register

Bit	Attr	Reset Value	Description
31	RW	0x0	GTPRUN; General Purpose Timer Run. 0: Timer stop

			1: Timer run
30	W	0x0	GPTRST; General Purpose Timer Reset. 0: No action 1: Load Counter Value Writing a one to this bit will reload the GPTCNT with the value in GPTLD.
29:25	-	-	Reserved
24	RW	0x0	GPTMODE; General Purpose Timer Mode. 0: One shot mode 1: Repeat mode
23:0	R	0x0	GPTCNT; General Purpose Timer Counter. This field is the value of the running timer.

OTG_SBUSCFG

Address: Operational Base + offset (0x0090)

System bus configuration register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	R	0x40	AHBBRST; AHB burst type of DMA master port: 000: INCR 001: INCR4 + SINGLE for less than 4-beat 010: INCR8 + INCR4/SINGLE for less than 8-beat 011: INCR16 + INCR8/NCR4/SINGLE for less than 16-beat 100: Reserved 101: INCR4 + INCR for less than 4-beat 110: INCR8 + INCR for less than 8-beat 111: INCR16 + INCR for less than 16-beat

OTG_CAPLEN

Address: Operational Base + offset (0x0100)

Capability register length

Bit	Attr	Reset Value	Description
7:0	R	0x40	Capability register length This register is used as an offset to add to register base to find the beginning of the Operational Register Space.

OTG_HCIVER

Address: Operational Base + offset (0x0102)

Host interface version number

Bit	Attr	Reset Value	Description

15:0	R	0x100	Host interface version number. This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.
------	---	--------------	---

OTG_HCSPAR

Address: Operational Base + offset (0x0104)

Host controller structural parameter

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:24	R	0x0	N_TT; Number of Transaction Translators. This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. This is a non-EHCI field to support embedded TT.
23:20	R	0x0	N_PTT; Number of Ports per Transaction Translator. This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller.
19:17	-	-	Reserved.
16	R	0x1	PI; Port Indicators. This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator.
15:12	R	0x0	NCC; Number of Companion Controller. This field indicates the number of companion controllers associated with this USB2.0 host controller. A zero in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported.
11:8	R	0x0	N_PCC; Number of Ports per Companion Controller. This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software.
7:5	-	-	Reserved.
4	R	0x1	PPC; Port Power Control. This field indicates whether the host controller implementation includes port power control.
3:0	R	0x1	N_PORT; Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller.

OTG_HCCPAR

Address: Operational Base + offset (0x0108)

Host controller capability parameter

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:8	R	0x0	EECP; EHCI Extended Capabilities Pointer.
7:4	R	0x0	IST; Isochronous Scheduling Threshold
3	-	-	Reserved.
2	R	0x1	ASP; Asynchronous Schedule Park Capability.
1	R	0x1	PFL; Programmable Frame List Flag.
0	R	0x0	ADC; 64-bit Addressing Capability. No 64-bit addressing capability is supported.

OTG_DCIVER

Address: Operational Base + offset (0x0120)

Device interface version number

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	R	0x1	The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

OTG_DCCPAR

Address: Operational Base + offset (0x0124)

Device controller capability parameter

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved.
8	R	0x1	HC; Host Capable. When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7	R	0x1	DC; Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6:5	-	-	Reserved.
4:0	R	0x4	DEN; Device Endpoint Number. This field indicates the number of endpoints built into the device controller. There are 4 endpoints in controller.

OTG_USBCMD

Address: Operational Base + offset (0x0140)

USB command register

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:16	RW	0x08	ITC; Interrupt Threshold Control

			<p>The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below.</p> <table> <thead> <tr> <th>Value</th><th>Maximum Interrupt Interval</th></tr> </thead> <tbody> <tr> <td>00h</td><td>Immediate (no threshold)</td></tr> <tr> <td>01h</td><td>1 micro-frame</td></tr> <tr> <td>02h</td><td>2 micro-frames</td></tr> <tr> <td>04h</td><td>4 micro-frames</td></tr> <tr> <td>08h</td><td>8 micro-frames (1ms)</td></tr> <tr> <td>10h</td><td>16 micro-frames (2ms)</td></tr> <tr> <td>20h</td><td>32 micro-frames (4ms)</td></tr> <tr> <td>40h</td><td>64 micro-frames (8ms)</td></tr> </tbody> </table>	Value	Maximum Interrupt Interval	00h	Immediate (no threshold)	01h	1 micro-frame	02h	2 micro-frames	04h	4 micro-frames	08h	8 micro-frames (1ms)	10h	16 micro-frames (2ms)	20h	32 micro-frames (4ms)	40h	64 micro-frames (8ms)
Value	Maximum Interrupt Interval																				
00h	Immediate (no threshold)																				
01h	1 micro-frame																				
02h	2 micro-frames																				
04h	4 micro-frames																				
08h	8 micro-frames (1ms)																				
10h	16 micro-frames (2ms)																				
20h	32 micro-frames (4ms)																				
40h	64 micro-frames (8ms)																				
15	RW	0x0	FS[2]; Frame List Size. FS[1:0] is defined in OTG_USBCMD[3:2]																		
14	RW	0x0	ATDTW; Add dTD TripWire (device mode only) This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when the state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.																		
13	RW	0x0	SUTW; Setup TripWire. (device mode only) This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off, then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists.																		
12	-	-	Reserved.																		
11	R or RW	0x0	ASPE; Asyn. Schedule Park Mode Enable. If the Asynchronous Park Capability bit in the OTG_HCCPAR register is a one, then this bit defaults to a 0x1 and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled.																		
10	-	-	Reserved.																		
9:8	R or	0x00	ASP; Async. Schedule Park Mode Count. If the Async. Park Capability bit in the OTG_HCCPAR																		

	RW		register is a one, then this field defaults to 0x03 and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x01 to 0x03. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior.
7	R	0x0	LR; Light Host/Device Controller Reset. Not Implemented. This field will always be "0".
6	RW	0x0	IAA; Interrupt on Async Advance Doorbell. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the OTG_USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the OTG_USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5	RW	0x0	ASE; Asynchronous Schedule Enable This bit controls whether the host controller skips processing the Asynchronous Schedule. Values meaning 0: Do not process the Async. Schedule. 1: Use the ASYNCLISTADDR register to access the Async. Schedule. Only the host controller uses this bit.
4	RW	0x0	PSE; Periodic Schedule Enable. This bit controls whether the host controller skips

			<p>processing the Periodic Schedule.</p> <p>0: Do not process the Periodic Schedule 1: Use the PERIODICLISTBASE register to Schedule</p> <p>Only the host controller uses this bit.</p>
3:2	RW or R	0x0	<p>FS[1:0]; Frame List Size</p> <p>This field is Read/Write only if Programmable Frame List Flag in the OTG_HCCPAR registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from OTG_USBCMD bits 15, 3 and 2.</p> <p>FS Values Meaning</p> <ul style="list-style-type: none"> 0001024 elements (4096 bytes) 001512 elements (2048 bytes) 010256 elements (1024 bytes) 011128 elements (512 bytes) 10064 elements (256 bytes) 10132 elements (128 bytes) 11016 elements (64 bytes) 1118 elements (32 bytes) <p>Only the host controller uses this field.</p>
1	RW	0x0	<p>RST; Controller Reset.</p> <p>Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register.</p> <p><u>Host Controller:</u></p> <p>When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the OTG_USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior.</p> <p><u>Device Controller:</u></p> <p>When software writes a one to this bit, the Device</p>

			Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the OTG_USBCMD Run/Stop bit should be set to 0.
0	RW	0x0	<p>RS; Run/Stop 1: Run. 0: Stop.</p> <p><u>Host Controller:</u> When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e. HCHalted in the OTG_USBSTS register is a one).</p> <p><u>Device Controller:</u> Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event.</p>

OTG_USBSTS

Address: Operational Base + offset (0x0144)

USB status register.

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them.

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.

25	RW	0x0	TI1; General Purpose Timer Interrupt 1. This bit is set when the counter in the OTG_GPT1CTRL (Non-EHCI) register transitions to zero. Writing a one to this bit will clear it.
24	RW	0x0	TI0; General Purpose Timer Interrupt 0. This bit is set when the counter in the OTG_GPT0CTRL (Non-EHCI) register transitions to zero. Writing a one to this bit will clear it.
23:20	-	-	Reserved.
19	RW	0x0	UPI; USB Host Periodic Interrupt. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero.
18	RW	0	UAI; USB Host Asynchronous Interrupt. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero.
17	-	-	Reserved.
16	R	0x0	NAKI; NAK Interrupt Bit. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are

			cleared.
15	R	0x0	AS; Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the OTG_USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0).
14	R	0x0	PS; Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0).
13	R	0x0	RCL; Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller.
12	R	0x0	HCH; HCHaIted. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller.
11:9	-	-	Reserved.
8	RW	0x0	SLI; DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a one. This bit is only cleared by software writing a 1 to it. Only used by the device controller.
7	RW	0x0	SRI; SOF Received.

			<p>When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125us in HS mode and will be synchronized to the actual SOF that is received.</p> <p>Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp.</p> <p>In host mode, this bit will be set every 125us and can be used by host controller driver as a time base.</p> <p>Software writes a 1 to this bit to clear it. This is a non-EHCI status bit.</p>
6	RW	0x0	<p>URI; USB Reset Received.</p> <p>When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit.</p> <p>Only used by the device controller.</p>
5	RW	0x0	<p>AAI; Interrupt on Async Advance.</p> <p>System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the OTG_USBCMD register. This status bit indicates the assertion of that interrupt source.</p> <p>Only used by the host controller.</p>
4	RW	0x0	<p>SEI; System Error.</p> <p>This bit will be set to "1" when an Error response is seen by the AHB master interface.</p>
3	RW	0x0	<p>FRI; Frame List Rollover.</p> <p>The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the OTG_USBCMD register) is 1024, the Frame Index Register rolls over every time OTG_FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every</p>

			time OTG_FHINDEX [12] toggles. Only used by the host controller.
2	RW	0x0	PCI; Port Change Detect. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when it detects resume signaling or the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible.
1	RW	0x0	UE1; USB Error Interrupt. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set
0	RW	0x0	UI; USB Interrupt (USBINT). This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

OTG_USBINTR

Address: Operational Base + offset (0x0148)

USB interrupt enable register.

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (OTG_USBSTS) still shows interrupt sources even if they are disabled by the OTG_USBINTR register, allowing polling of interrupt events by the software.

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25	RW	0x0	TIE1; When this bit is a one, and the GPTINT1 bit in the OTG_USBSTS register is a one, the controller will issue

			an interrupt. The interrupt is acknowledged by software clearing the GPTINT1 bit.
24	RW	0x0	TIE0; When this bit is a one, and the GPTINT0 bit in the OTG_USBSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT0 bit.
23:20	-	-	Reserved.
19	RW	0x0	UPIE; When this bit is a one, and the UPI bit in the OTG_EXTSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UPI bit.
18	RW	0x0	UAIE; When this bit is a one, and the UAI bit in the OTG_EXTSTS register is a one, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UAI bit.
17	-	-	Reserved.
16	R	0x0	NAKIE; When this bit is a one, and the NAKI bit in the OTG_EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the NAKI bit.
15:9	-	-	Reserved.
8	RW	0x0	SLE; DCSSuspend Interrupt Enable When this bit is a one, and the SLI bit in the OTG_USBSTS register transitions, the Controller will issue an interrupt. The interrupt is acknowledged by software writing a one to the SLI bit. Only used by the device controller.
7	RW	0x0	SRE; SOF Received Interrupt Enable. When this bit is a one, and the SRI bit in the OTG_USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the SRI bit.
6	RW	0x0	URE; USB Reset Received Interrupt Enable. When this bit is a one, and the URI bit in the OTG_USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the URI bit. Only used by the device controller.
5	RW	0x0	AAE; Interrupt on Async Advance Enable.

			When this bit is a one, and the AAI bit in the OTG_USBSTS register is a one, the Controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on AAI bit. Only used by the host controller.
4	RW	0x0	SEE; System Error Interrupt Enable. When this bit is a one, and the SEI bit in the OTG_USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the SSI bit.
3	RW	0x0	FRE; Frame List Rollover Interrupt Enable. When this bit is a one, and the FRI bit in the USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the FRI bit. Only used by the host controller.
2	RW	0x0	PCE; Port Change Detect Interrupt Enable When this bit is a one, and the PCI bit in the USBSTS register is a one, the Controller will issue an interrupt. The interrupt is acknowledged by software clearing the PCI bit.
1	RW	0x0	UEE; USB Error Interrupt When this bit is a one, and the UEI bit in the OTG_USBSTS register is a one, the Controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UEI bit in the OTG_USBSTS register.
0	RW	0x0	UE; USB Interrupt Enable. When this bit is a one, and the UI bit in the OTG_USBSTS register is a one, the Controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UI bit.

OTG_FRINDEX

Address: Operational Base + offset (0x014C)

USB frame index

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N:3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution, while bits [2:0] are used to indicate the current micro-frame number.

The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the OTG_USBCMD register.

This register must be written as a DWord. Byte writes produce-undefined results. This register

cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the OTG_FRINDEX[13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, OTG_FRINDEX[13:3] will be checked against the SOF marker. If OTG_FRINDEX[13:3] is different from the SOF marker, OTG_FRINDEX[13:3] will be set to the SOF value and OTG_FRINDEX [2:0] will be set to zero (i.e. SOF for 1 ms frame). If OTG_FRINDEX [13:3] is equal to the SOF value, OTG_FRINDEX [2:0] will be incremented (i.e. SOF for 125 us micro-frame).

Bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13:0	RW	0x0	<p>FRINDEX; Frame Index.</p> <p>The value, in this register, increments at the end of each time frame (e.g. micro-frame). Bits[N:3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index.</p> <p>The following illustrates values of N based on the value of the Frame List Size field in the OTG_USBCMD register, when used in host mode.</p>

OTG_FLBADDR

Address: Operational Base + offset (0x0154)

Frame list base-address

This register is shared between the host controller and the device controller operation. Writes must be DWord Writes

Host Controller (PERIODICLISTBASE)

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (OTG_FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Bit	Attr	Reset Value	Description
31:12	RW	0x0	<p>BASEADR; Base Address (Low).</p> <p>These bits correspond to memory address signals [31:12], respectively.</p> <p>Only used by the host controller.</p>
11:0	-	-	Reserved.

Device Controller (USB DEVICEADDR)

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor. The USBADRA is used to accelerate the SET_ADDRESS sequence by allowing the DCD to preset the USBADR register before the status phase of the SET_ADDRESS descriptor.

Bit	Attr	Reset Value	Description
31:25	RW	0x0	USBADR; Device Address. These bits correspond to the USB device address
24	RW	0x0	USBADRA; Device Address Advance. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
23:0	-	-	Reserved.

OTG_NALADDR

Address: Operational Base + offset(0x158)

Next async. list address

This register is shared between the host controller and the device controller operation.

Host Controller (ASYNCLISTADDR)

This 32-bit register contains the address of the next asynchronous queue head to be executed

by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Bit	Attr	Reset Value	Description
31:5	RW	0x0	ASYBASE; Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.
4:0	-	-	Reserved.

Device Controller (ENDPOINTLISTADDR)

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read. The memory structure referenced by this physical memory pointer is assumed 64-byte.

Bit	Attr	Reset Value	Description
31:11	RW	0x0	EPBASE; Endpoint List Pointer (Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). (i.e. one queue head per endpoint & direction.)
10:0	-	-	Reserved.

OTG_TTCTRL

Address: Operational Base + offset (0x015C)

Host controller embedded TT async. buffer status.

This register contains parameters needed for internal TT operations. This Register is not used in the device controller operation.

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30:24	RW	0x0	TTHA; Internal TT Hub Address Representation. This field is used to match against the Hub Address field in QH & siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in the QH/siTD
23:2	-	-	Reserved.
1	RW	0x0	TTAC; Embedded TT Async. Buffers Clear. This field will clear all pending transactions in the embedded TT Asynchronous Buffer(s). The clear will take as much time as necessary to

			clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	R	0x0	TTAS; Embedded TT Async Buffers Status This read only bit will be '1' if one or more transactions are being held in the embedded TT Asynchronous Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

OTG_BURSTSZ

Address: Operational Base + offset (0x0160)

Burst size register

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:8	RW	0x10	TXPBURST; Programmable TX Burst Length. This register represents the maximum length of a burst in 32- bit words while moving data from system memory to the USB bus. If field AHBBRST of register OTG_SBUSCFG (090h) is different from zero, this field TXPBURST will return the value of the INCRx length. Supported values are integer values from 4 to 128. It is recommended to set this value to a integer sub-multiple of VUSB_HS_TX_CHAN. Different values will not use all the available buffer space, preventing proper TX endpoint priming in stream disable mode (SDIS bit set to '1').
7:0	RW	0x10	RXPBURST; Programmable RX Burst Length This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory. If field AHBBRST of register OTG_SBUSCFG (090h) is different from zero, this field RXPBURST will return the value of the INCRx length. Supported values are integer values from 4 to 128. It is recommended to set this value to a integer sub-multiple of VUSB_HS_RX_DEPTH.

OTG_TXTUNE

Address: Operational Base + offset (0x0164)

Host transmit pre-buffer package tuning register.

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system. Definitions:

T0 = Standard packet overhead

T1 = Time to send data payload

Tff = Time to fetch packet into TX FIFO up to specified level.

Ts = Total Packet Flight Time (send-only) packet (Ts = T0 + T1)

Tp = Total Packet Time (fetch and send) packet

Tp = Tff + T0 + T1

Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure Tp remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < Ts then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a “back-off” event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHHEALTH (Tff) described below.

Bit	Attr	Reset Value	Description
31:22	-	-	Reserved.
21:16	RW	0x0	TXFIFOTHRES; FIFO Burst Threshold. This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in OTG_USBMODE register is set.
15:13	-	-	Reserved.
12:8	RW	0x0	TXSCHHEALTH; Scheduler Health Counter. This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7	-	-	Reserved.
6:0	RW	0x0	TXSCHOH; Scheduler Overhead. This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the

		<p>TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization.</p> <p>The time unit represented in this register is 1.267us when a device is connected in High- Speed Mode for OTG.</p> <p>The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG.</p>
--	--	--

OTG_TXTTUNE

Address: Operational Base + offset (0x0168)

Host TT transmit pre-buffer package tuning register.

Bit	Attr	Reset Value	Description
31:13	-	-	Reserved.
12:8	RW	0x0	TXTTSCHHEALTH; TT Scheduler Health Counter.
7:5	-	-	Reserved.
4:0	RW	0x0	TXTTSCHOH; TT Scheduler Overhead. Same description as TXSCHOH.

OTG_EPNAK

Address: Operational Base + offset (0x0178)

Endpoint NAK register.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19:16	RW	0x0	EPTN; TX Endpoint NAK. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. EPTN[3] – Endpoint #3 EPTN[2] – Endpoint #2 EPTN[1] – Endpoint #1 EPTN[0] – Endpoint #0
15:4	-	-	Reserved.
3:0	RW	0x0	EPRN; RX Endpoint NAK. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. EPRN[3] – Endpoint #3 EPRN[2] – Endpoint #2 EPRN[1] – Endpoint #1

			EPRN[0] – Endpoint #0
--	--	--	-----------------------

OTG_EPNAKEN

Address: Operational Base + offset (0x017C)

Endpoint NAK enable register.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19:16	RW	0x0	<p>EPTNE; TX Endpoint NAK Enable.</p> <p>Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set.</p> <p>EPTNE[3] – Endpoint #3 EPTNE[2] – Endpoint #2 EPTNE[1] – Endpoint #1 EPTNE[0] – Endpoint #0</p>
15:4	-	-	Reserved.
3:0	RW	0x0	<p>EPRNE; RX Endpoint NAK Enable.</p> <p>Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set.</p> <p>EPRNE[3] – Endpoint #3 EPRNE[2] – Endpoint #2 EPRNE[1] – Endpoint #1 EPRNE[0] – Endpoint #0</p>

OTG_PORTSC

Address: Operational Base + offset (0x0184)

Port control and status register.

This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are: no device connected; port disable.

If the port has port power control, this state remains until software applies power to the port by setting port power (PP) to one.

When in device mode, this register does not support power control and is only used for status port reset, suspend, and current connect status.

Bit	Attr	Reset Value	Description
31:28	-	-	Reserved.
27:26	R	0x0	<p>PSPD; Port Speed.</p> <p>This register field indicates the speed at which the port is operating.</p> <p>00 – Full Speed 01 – Low Speed</p>

			10 – High Speed 11 – Not connected																								
25	-	-	Reserved.																								
24	RW	0x0	PFSC; Port Force Full Speed Connect. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with a HS host, hub or device.																								
23	RW	0x0	PHCD; PHY Low Power Clock Disable. Writing this bit to a '1b' will disable the PHY clock. Writing a '0b' enables it. Reading this bit will indicate the status of the PHY clock.																								
22	-	-	Reserved.																								
21	RW	0x0	WKDS; Wake on Disconnect Enable. Writing this bit to a one enables the port to be sensitive to device disconnects as wake- up events. This field is zero if Port Power (PP) is '0' or in device mode. Only used in host mode.																								
20	RW	0x0	WKCN; Wake on Connect Enable. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power(PP) is '0' or in device mode. Only used in host mode.																								
19:16	RW	0x0	PTC; Port Test Control. Any other value than zero indicates that the port is operating in test mode. <table> <tr> <td>Value</td> <td>Specific Test</td> </tr> <tr> <td>0000b</td> <td>TEST_MODE_DISABLE</td> </tr> <tr> <td>0001b</td> <td>J_STATE</td> </tr> <tr> <td>0010b</td> <td>K_STATE</td> </tr> <tr> <td>0011b</td> <td>SE0 (host) / NAK (device)</td> </tr> <tr> <td>0100b</td> <td>Packet</td> </tr> <tr> <td>0101b</td> <td>FORCE_ENABLE_HS</td> </tr> <tr> <td>0110b</td> <td>FORCE_ENABLE_FS</td> </tr> <tr> <td>0111b</td> <td>FORCE_ENABLE_LS</td> </tr> <tr> <td>1000b</td> <td>Reserved</td> </tr> <tr> <td>...</td> <td>Reserved</td> </tr> <tr> <td>1111b</td> <td>Reserved</td> </tr> </table>	Value	Specific Test	0000b	TEST_MODE_DISABLE	0001b	J_STATE	0010b	K_STATE	0011b	SE0 (host) / NAK (device)	0100b	Packet	0101b	FORCE_ENABLE_HS	0110b	FORCE_ENABLE_FS	0111b	FORCE_ENABLE_LS	1000b	Reserved	...	Reserved	1111b	Reserved
Value	Specific Test																										
0000b	TEST_MODE_DISABLE																										
0001b	J_STATE																										
0010b	K_STATE																										
0011b	SE0 (host) / NAK (device)																										
0100b	Packet																										
0101b	FORCE_ENABLE_HS																										
0110b	FORCE_ENABLE_FS																										
0111b	FORCE_ENABLE_LS																										
1000b	Reserved																										
...	Reserved																										
1111b	Reserved																										

Refer to Chapter 7 of the USB Specification Revision 2.0

			<p>for details on each test mode.</p> <p>The FORCE_ENABLE_FS and FORCE_ENABLE_LS are extensions to the test mode support specified in the EHCI specification.</p> <p>Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed.</p> <p>Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point.</p> <p>Note: Low speed operations are not supported when in device mode.</p>									
15:13	-	-	Reserved.									
12	RW	0x0	<p>PP; Port Power.</p> <p>The function of this bit depends on the value of the Port Power Switching (PPC) field in the OTG_HCSPAR register. The behavior is as follows:</p> <table> <tr> <td>PPC 0b</td> <td>PP 0b</td> <td>Operation</td> </tr> <tr> <td>1b</td> <td>1b</td> <td>Read Only.</td> </tr> <tr> <td>1b 0b</td> <td>0b</td> <td>Read/Write.</td> </tr> </table> <p>A Controller in device mode does not have port power control switches.</p> <p>A Controller in host mode requires port power control switches.</p> <p>This bit represents the current setting of the switch ('0'=off, '1'=on). When power is not available on a port (i.e. PP equals to '0'), the port is non-functional and will not report attaches, detaches, etc.</p> <p>When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the Controller driver from '1' to '0' (removing power from the port).</p> <p>In device mode port power control is not necessary, thus PPC and PP = 0.</p>	PPC 0b	PP 0b	Operation	1b	1b	Read Only.	1b 0b	0b	Read/Write.
PPC 0b	PP 0b	Operation										
1b	1b	Read Only.										
1b 0b	0b	Read/Write.										
11:10	R	-	<p>LS; Line Status.</p> <p>These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The encoding of the bits</p>									

			<p>are:</p> <table> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>00b</td><td>SE0</td></tr> <tr> <td>10b</td><td>J-state</td></tr> <tr> <td>01b</td><td>K-state</td></tr> <tr> <td>11b</td><td>Undefined</td></tr> </tbody> </table> <p>In host mode, the use of linestate by the Controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS.</p> <p>In device mode, the use of linestate by the Controller driver is not necessary.</p>	Value	Meaning	00b	SE0	10b	J-state	01b	K-state	11b	Undefined
Value	Meaning												
00b	SE0												
10b	J-state												
01b	K-state												
11b	Undefined												
9	R	0x0	<p>HSP; High-Speed Port.</p> <p>When the bit is one, the port is in high-speed mode and if set to zero, the port is not in a high-speed mode.</p> <p>Note: HSP is redundant with PSPD but will remain in the design for compatibility.</p>										
8	RW	0x0	<p>PR; Port Reset.</p> <p>This field is zero if Port Power(PP) is '0'.</p> <p>Host mode: 1=Port is in Reset. 0=Port is not in Reset.</p> <p>When software writes '1' to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to '0' after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.</p> <p>Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the OTG_USBSTS register.</p>										
7	RW	0x0	<p>SUSP; Suspend.</p> <p>Host mode:</p> <p>1=Port in suspend state. 0=Port not in suspend state.</p> <p>Port Enabled bit and Suspend bit of this register define the port states as follows:</p>										

			<p>Bits [Port Enabled, Suspend] Port State</p> <table> <tr> <td>0x</td><td>Disable</td></tr> <tr> <td>10</td><td>Enable</td></tr> <tr> <td>11</td><td>Suspend</td></tr> </table> <p>When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to '1'. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB.</p> <p>The Controller when in host mode will unconditionally set this bit to zero when software sets the Force Port Resume bit to '0'. The Controller ignores a write of '0' to this bit.</p> <p>If software sets this bit to a '1' when the port is not enabled (i.e. Port Enabled bit is a '0') the results are undefined.</p> <p>This field is '0' if Port Power (PP) is '0' in host mode.</p> <p>Device mode: Read Only. 1=Port in suspend state. 0=Port not in suspend state.</p> <p>In device mode this bit is a read only status bit.</p>	0x	Disable	10	Enable	11	Suspend
0x	Disable								
10	Enable								
11	Suspend								
6	RW	0x0	<p>FPR; Force Port Resume. 1= Resume detected/driven. 0= No resume (K-state) detected/driven.</p> <p>Host mode: Software sets this bit to one to drive resume signaling. The Controller sets this bit to '1' if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a '1' because a J-to-K transition is detected, the Port Change Detect bit in the OTG_USBSTS register is also set to '1'. This bit will automatically change to '0' after the resume sequence is complete. This behavior is different from EHCI where the controller driver is required to set this bit to a '0' after the resume duration is timed in the driver.</p>						

			<p>Note that when the controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a '1'. This bit will remain a '1' until the port has switched to idle. Writing a '0' has no affect because the port controller will time the resume operation, clear the bit and the port control state switches to HS or FS idle.</p> <p>This field is '0' if Port Power (PP) is '0' in host mode. This bit is not-EHCI compatible.</p> <p>Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to '1' to drive resume signaling before clearing. The Controller will set this bit to '1' if a J- to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a '1' because a J-to-K transition detected, the Port Change Detect bit in the USBSTS register is also set to '1'.</p>						
5:4	-	-	Reserved.						
3	RW	0x0	<p>PEC; Port Enabled Change. If set to '1' indicates a Port Enabled/Disabled status change.</p> <p>Host mode: For the root hub, this bit gets set to a '1' only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a '1' to it. This field is '0' if Port Power (PP) is '0'.</p> <p>Device mode: The device port is always enabled. (This bit will be '0').</p>						
2	RW	0x0	<p>PE; Port Enabled.</p> <table> <tr> <td>Value</td> <td>Meaning</td> </tr> <tr> <td>1</td> <td>Enable</td> </tr> <tr> <td>0</td> <td>Disable</td> </tr> </table> <p>Host mode:</p>	Value	Meaning	1	Enable	0	Disable
Value	Meaning								
1	Enable								
0	Disable								

			<p>Ports can only be enabled by Controller as a part of the reset and enable. Software cannot enable a port by writing a '1' to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other controller and bus events.</p> <p>When the port is disabled ('0b'), downstream propagation of data is blocked except for reset.</p> <p>This field is '0' if Port Power (PP) is '0' in host mode.</p> <p>Device mode: The device port is always enabled. (This bit will be always '1').</p>
1	RW	0x0	<p>CSC; Connect Status Change. If set to '1' indicates a change in Current Connect Status (CCS).</p> <p>Host mode: Indicates a change has occurred in the port's Current Connect Status. The Controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a '1' to it.</p> <p>This field is '0' if Port Power(PP) is '0' in host mode.</p> <p>Device mode: This bit is undefined in device mode.</p>
0	R	0x0	<p>CCS; Current Connect Status.</p> <p>Host mode: Value Meaning 1Device is present on port. 0No device is present.</p> <p>This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit to be set.</p>

		<p>This field is '0' if Port Power (PP) is '0' in host mode.</p> <p>Device mode:</p> <table> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>1</td><td>Attached.</td></tr> <tr> <td>0</td><td>Not Attached.</td></tr> </tbody> </table> <p>A '1' indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the PSPD bits in this register. A '0' indicates that the device did not attach successfully or was forcibly disconnected by the software writing a '0' to the Run/Stop bit in the OTG_USBCMD register. It does not state the device being disconnected or suspended.</p>	Value	Meaning	1	Attached.	0	Not Attached.
Value	Meaning							
1	Attached.							
0	Not Attached.							

OTG_OTGSC

Address: Operational Base + offset (0x01A4)

OTG status and control register.

The Controller implements one On-The-Go (OTG) Status and Control register.

The OTGSC register has four sections:

OTG Interrupt enables	(Read/Write)
OTG Interrupt status	(Read/Write to Clear)
OTG Status inputs	(Read Only)
OTG Controls	(Read/Write)

The status inputs are debounced using a 1Msec time constant. Values on the status inputs that do not persist for more than 1Msec will not cause an update of the status input register, or will cause an OTG interrupt. This register only exists in an OTG implementation. All other controller implementations have this register reserved.

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30	RW	0x0	DPIE; Data Pulse Interrupt Enable. This bit enables the generation of an interrupt if bit DPIS is set.
29	RW	0x0	1msE; millisecond timer Interrupt Enable. This bit enables the generation of an interrupt if bit 1msS is set.
28	RW	0x0	BSEIE; B Session End Interrupt Enable. This bit enables the generation of an interrupt if bit BSEIS is set.
27	RW	0x0	BSVIE; B Session Valid Interrupt Enable. This bit enables the generation of an interrupt if bit BSVIS is set.
26	RW	0x0	ASVIE; A Session Valid Interrupt Enable. This bit enables the generation of an interrupt if bit ASVIS

			is set.
25	RW	0x0	AVVIE; A VBus Valid Interrupt Enable. This bit enables the generation of an interrupt if bit AVVIS is set.
24	RW	0x0	IDLE; USB ID Interrupt Enable. This bit enables the generation of an interrupt if bit IDIS is set.
23	-	-	Reserved.
22	RW	0x0	DPIS; Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when OTG_USBMODE.CM = Host ('11b') and OTG_PORTSC.PP = Off ('0b'). Software must write a '1' to clear this bit.
21	RW	0x0	1msS; 1 millisecond timer Interrupt Status. This bit is set once every millisecond. Software must write a '1' to clear this bit.
20	RW	0x0	BSEIS; B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software must write a '1' to clear this bit.
19	RW	0x0	BSVIS; B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a '1' to clear this bit.
18	RW	0x0	ASVIS; A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a '1' to clear this bit.
17	RW	0x0	AVVIS; Frame Index. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a '1' to clear this bit.
16	RW	0x0	IDIS; USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software must write a '1' to clear this bit.
15	-	-	Reserved.
14	R	0x0	DPS; Data Bus Pulsing Status. A '1' indicates data bus pulsing is being detected on the port.
13	R	0x0	1msT; 1 millisecond timer toggle. This bit toggles once per millisecond.
12	R	0x0	BSE; B Session End. Indicates VBus is below the B session end threshold.
11	R	0x0	BSV; B Session Valid.

			Indicates VBus is above the B session valid threshold.
10	R	0x0	ASV; A Session Valid. Indicates VBus is above the A session valid threshold.
9	R	0x0	AVV; A VBus Valid. Indicates VBus is above the A VBus valid threshold.
8	R	0x0	ID; USB ID. 0 = A device. 1 = B device.
7	RW	0x0	HABA; Hardware Assist B-Disconnect to A-connect. 0 = Disabled. 1 = Enable automatic B-disconnect to A-connect sequence.
6	RW	0x0	HADP; Hardware Assist Data-Pulse. If set, the hardware assist data pulsing sequence starts.
5	RW	0x1	IDPU; ID Pullup. This bit provides control over the ID pull-up resister. 0 = Off. 1 = On. When this bit is '0' the ID input will not be sampled.
4	RW	0x0	DP; Data Pulsing. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3	RW	0x0	OT; OTG Termination. This bit must be set when the Controller is in device mode. It controls the pulldown on DM.
2	RW	0x0	HAAR; Hardware Assist Auto-Reset. 0 = Disabled. 1 = Enable automatic reset after connect on host port.
1	RW	0x0	VC; VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0	RW	0x0	VD; VBUS Discharge. Setting this bit causes VBus to discharge through a resistor.

OTG_USBMODE

Address: Operational Base + offset (0x01A8)

USB device mode register.

Bit	Attr	Reset Value	Description
31: 16	-	-	Reserved.
15	RW	0x0	SRT; Shorten Reset Time When the Controller is in host mode, this bit enables a bypass of the Chirp J/K reset handshake, saving 6-7ms in simulation time for each reset sequence. This bit should

			only be used for initial system integration simulations, and should always be set to 0 for normal operation.
14:6			Reserved
5	RW	0x0	<p>VBPS; Vbus Power Select.</p> <p>Value Meaning</p> <p>0 OTGx_DRVVBUS pin output is '0' 1 OTGx_DRVVBUS pin output is '1'</p> <p>Only used in host mode.</p>
4	RW	0x0	SDIS; Stream Disable Mode.
3	RW	0x0	<p>SLOM;</p> <p>Value Meaning</p> <p>0Inactive 1Active</p> <p>Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB.</p> <p>Note1: Time duration to pre-fill the FIFO becomes significative when stream disable is active. See TXFILLTUNING and XTTFILLTUNING (MPH Only) to characterize the adjustments needed for the scheduler when using this feature.</p> <p>Note2: The use of this feature substantially limits of the overall USB performance that can be achieved</p> <p>Device mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode, when enabled, ensures that the RX and TX buffers are sufficient to contain an entire packet, so the usual double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems.</p> <p>Note: In High-Speed mode, all packets received will be responded to with a NYET handshake when stream disable is active.</p> <p>Setup Lockout Mode of device mode</p>

			<p>This bit controls behavior of the setup lock mechanism.</p> <p>Value Meaning</p> <p>0Setup Lockouts On. 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in OTG_USBCMD).</p>
2	RW	0x0	<p>ES; Endian Select.</p> <p>This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words.</p> <p>Value Meaning</p> <p>0Little Endian 1Big Endian</p>
1:0	RW	0x0	<p>CM; Controller Mode.</p> <p>Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability (OTG), the Controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RST bit in the OTG_USBCMD register before reprogramming this register.</p> <p>Value Meaning</p> <p>0Idle (in host/device mode). 01Reserved. 10Controller in device mode. 11Controller in host mode.</p>

OTG_EPSPSTAT

Address: Operational Base + offset (0x01AC)

Endpoint setup status register.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	<p>ENDPTSETUPSTAT; Setup Endpoint Status.</p> <p>For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from</p>

			Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. This register is only used in device mode.
--	--	--	---

OTG_EPPRIME

Address: Operational Base + offset (0x01B0)

Endpoint initialization register.

Bit	Attr	Reset Value	Description								
31:20	-	-	Reserved.								
19:16	RW	0x0	<p>PETB; Prime Endpoint Transmit Buffer.</p> <p>For each endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.</p> <p>Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.</p> <table> <tr> <td>PETB[3]</td> <td>– Endpoint #3</td> </tr> <tr> <td>PETB[2]</td> <td>– Endpoint #2</td> </tr> <tr> <td>PETB[1]</td> <td>– Endpoint #1</td> </tr> <tr> <td>PETB[0]</td> <td>– Endpoint #0</td> </tr> </table>	PETB[3]	– Endpoint #3	PETB[2]	– Endpoint #2	PETB[1]	– Endpoint #1	PETB[0]	– Endpoint #0
PETB[3]	– Endpoint #3										
PETB[2]	– Endpoint #2										
PETB[1]	– Endpoint #1										
PETB[0]	– Endpoint #0										
15:4	-	-	Reserved.								
3:0	RW	0x0	<p>PERB; Prime Endpoint Receive Buffer.</p> <p>For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.</p> <p>Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.</p>								

			PERB[3] – Endpoint #3 PERB[2] – Endpoint #2 PERB[1] – Endpoint #1 PERB[0] – Endpoint #0
--	--	--	--

OTG_EPFLUSH

Address: Operational Base + offset (0x01B4)

Endpoint de-initialization register.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19:16	RW	0x0	FETB; Flush Endpoint Transmit Buffer. Writing a one to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FETB[3] – Endpoint #3 FETB[2] – Endpoint #2 FETB[1] – Endpoint #1 FETB[0] – Endpoint #0
15:4	-	-	Reserved.
3:0	RW	0x0	FERB; Flush Endpoint Receive Buffer. Writing a one to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful. FERB[3] – Endpoint #3 FERB[2] – Endpoint #2 FERB[1] – Endpoint #1 FERB[0] – Endpoint #0

OTG_EPSTAT

Address: Operational Base + offset (0x01B8)

Endpoint status register.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19:16	R	0x0	ETBR; Endpoint Transmit Buffer Ready. One bit for each endpoint indicates status of the respective

			<p>endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the OTG_EPPRIME register. There will always be a delay between setting a bit in the OTG_EPPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the OTG_EPPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the OTG_EPFLUSH register.</p> <p>Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.</p> <table> <tr><td>ETBR[3]</td><td>– Endpoint #3</td></tr> <tr><td>ETBR[2]</td><td>– Endpoint #2</td></tr> <tr><td>ETBR[1]</td><td>– Endpoint #1</td></tr> <tr><td>ETBR[0]</td><td>– Endpoint #0</td></tr> </table>	ETBR[3]	– Endpoint #3	ETBR[2]	– Endpoint #2	ETBR[1]	– Endpoint #1	ETBR[0]	– Endpoint #0
ETBR[3]	– Endpoint #3										
ETBR[2]	– Endpoint #2										
ETBR[1]	– Endpoint #1										
ETBR[0]	– Endpoint #0										
15:4	-	-	Reserved.								
3:0	R	0x0	<p>ERBR; Endpoint Receive Buffer Ready.</p> <p>One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the OTG_EPPRIME register. There will always be a delay between setting a bit in the OTG_EPPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the OTG_EPPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the OTG_EPFLUSH register.</p> <p>Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.</p> <table> <tr><td>ERBR[3]</td><td>– Endpoint #3</td></tr> <tr><td>ERBR[2]</td><td>– Endpoint #2</td></tr> <tr><td>ERBR[1]</td><td>– Endpoint #1</td></tr> <tr><td>ERBR[0]</td><td>– Endpoint #0</td></tr> </table>	ERBR[3]	– Endpoint #3	ERBR[2]	– Endpoint #2	ERBR[1]	– Endpoint #1	ERBR[0]	– Endpoint #0
ERBR[3]	– Endpoint #3										
ERBR[2]	– Endpoint #2										
ERBR[1]	– Endpoint #1										
ERBR[0]	– Endpoint #0										

OTG_EPCOMP

Address: Operational Base + offset (0x01BC)

Endpoint complete register.

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved.

19:16	RW	0x0	<p>ETCE ; Endpoint Transmit Complete Event. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register.</p> <p>ETCE[3] – Endpoint #3 ETCE[2] – Endpoint #2 ETCE[1] – Endpoint #1 ETCE[0] – Endpoint #0</p>
15:4	-	-	Reserved.
3:0	RW	0x0	<p>ERCE; Endpoint Receive Complete Event. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register.</p> <p>ERCE[3] – Endpoint #3 ERCE[2] – Endpoint #2 ERCE[1] – Endpoint #1 ERCE[0] – Endpoint #0</p>

OTG_EPCTRL0

Address: Operational Base + offset (0x01C0)

Endpoint control 0 register.

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23	RW	0x0	<p>TXE; TX Endpoint Enable. 1 – Enabled Endpoint0 is always enabled.</p>
22:20	-	-	Reserved.
19:18	RW	0x0	<p>TXT; TX Endpoint Type 00 – Control Endpoint0 is fixed as a Control End Point.</p>
17	-	-	Reserved.
16	RW	0x0	<p>TXS; TX Endpoint Stall – Read/Write 0 – End Point OK 1 – End Point Stalled</p> <p>Software can write a one to this bit to force the endpoint to</p>

			return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request.
			After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated OTG_EPSPSTAT bit is cleared.
15:8	-	-	Reserved.
7	RW	0x0	RXE, RX Endpoint Enable 1 – Enabled Endpoint0 is always enabled.
6:4	-	-	Reserved.
3:2	RW	0x0	RXT; RX Endpoint Type 00 – Control Endpoint0 is fixed as a Control End Point.
1	-	-	Reserved.
0	RW	0x0	RXS; RX Endpoint Stall – Read/Write 0 – End Point OK. 1 – End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated OTG_EPSPSTAT bit is cleared.

OTG_EPCTRLx (x=1~3)

Address: Operational Base + offset (0x01C4/0x01C8/0x01CC)

Endpoint control x register.

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23	RW	0x0	TXE; TX Endpoint Enable 0 – Disabled 1 – Enabled An Endpoint should be enabled only after it has been configured.
22	W	0x0	TXR; TX Data Toggle Reset Write 1 – Reset PID Sequence Whenever a configuration event is received for this

			Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device
21	RW	0x0	<p>TXI; TX Data Toggle Inhibit 0 – PID Sequencing Enabled. 1 – PID Sequencing Disabled.</p> <p>This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.</p>
20	-	-	Reserved.
19:18	RW	0x0	<p>TXT; TX Endpoint Type 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt</p>
17	-	-	Reserved.
16	RW	0x0	<p>TXS; TX Endpoint Stall 0 – End Point OK 1 – End Point Stalled</p> <p>This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated OTG_EPSPSTAT bit is cleared.</p> <p>Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints.</p>
15:8	-	-	Reserved.
7	RW	0x0	<p>RXE; RX Endpoint Enable 0 – Disabled 1 – Enabled</p> <p>An Endpoint should be enabled only after it has been configured.</p>
6	W	0x0	<p>RXR; RX Data Toggle Reset Write 1 – Reset PID Sequence</p> <p>Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.</p>

5	RW	0x0	RXI; RX Data Toggle Inhibit 0 – Disabled 1 – Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4	-	-	Reserved.
3:2	RW	0x0	RXT; RX Endpoint Type 00 – Control 01 – Isochronous 10 – Bulk 11 – Interrupt
1	-	-	Reserved.
0	RW	0x0	RXS ; RX Endpoint Stall 0 – End Point OK. 1 – End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated OTG_EPSPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

This OTG core could be programmable into Host mode, Device mode, or OTG mode. Below describes the operation for three modes.

Host Mode Operation

The host mode operation of the core is near EHCI compatible with few minor differences documented in this section. The particulars of the deviations occur in the areas summarized here:

Embedded Transaction Translator – Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.

Embedded design interface.

Embedded Transaction Translator Function

The USB-HS Controller working in host mode supports directly connected full and low speed devices without requiring a companion controller by mimicking the capabilities of a USB 2.0 high speed hub transaction translator. Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125us using the transceiver clock as a reference clock.

Miscellaneous variations from EHCI

1. Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the OTG_PORTSC register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to the reset the device.
- Software shall write a '0' to the reset the device after 10 ms. This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device is now operational and at this point the port speed has been determined.

2. Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator (PSPD) has been added to PORTSC provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator (HSP) has been added to PORTSC to signify that the port

is in High-Speed vs. Full/Low Speed – This information is redundant with the 2-bit Port Speed indicator above.

Device Mode Operation

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pullup on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs. Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

1. Set Controller Mode in the USBMODE register to device mode.
 - Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.
 - Only needed to set when using an OTG core. A Device only core has this hardwired to Device.
2. Allocate and Initialize device queue heads in system memory.
 - Minimum: Initialize device queue heads 0 Tx & 0 Rx.
 - All device queue heads associated with control endpoints must be initialized before the control endpoint is enabled. Non-Control device queue heads must be initialized before the endpoint is used and not necessarily before the endpoint is enabled.
3. Configure ENDPOINTLISTADDR Pointer.
4. Enable the microprocessor interrupt associated with the USB-HS core.
 - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
5. Set Run/Stop bit to Run Mode.
 - After the Run bit is set, a device USB reset will occur. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the following Port State and Control section below.
 - Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.
 - It is also not necessary to initially prime Endpoint 0 because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

Port State and Control

1. Bus Reset

A bus reset is used by the host to initialize downstream devices. When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0 and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be canceled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.

Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.

Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFFFFFF to ENDPTFLUSH register.

Read the reset bit in the PORTSC register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare).

A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD register. A hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the PORTSC to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device.

In some applications, it may not be possible to enable one or more pipes while in FS mode.

Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.

2. Suspend/Resume

a. Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither.

Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

b. Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the

DCD is notified by an interrupt (assuming DC Suspend Interrupt is enabled). When the DCSuspend bit in the PORTSC is set to a '1', the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

Note: Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

c. Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port. Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the PORTSC while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

Note: Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

3. Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device. The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a control type data channel used for device discovery and enumeration. Other types of endpoints support by USB include bulk, interrupt, and isochronous. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB-HS device controller hardware supports the USB 2.0 maximum of 4 endpoints specified numbers.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation.

The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory.

4. Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the ENDPTCTRLx register. Each 32-bit ENDPTCTRLx is split into an upper and lower half. The lower half of ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization.

Field	Value
Data Toggle Reset	"1"
Data Toggle Inhibit	"0"
Endpoint Type	'00' – Control
	'01' – Isochronous
	'10' – Bulk
	'11' – Interrupt
Endpoint Stall	"0"

5. Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework (chapter 9). A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints and automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

Note: Any write to the ENDPTCTRLx register during operational mode must preserve the endpoint type field (i.e. perform a read-modify-write).

USB Packet	Endpoint STALL Bit	Effect on STALL Bit	USB Response
SETUP packet received by a non-control endpoint	N/A	None	STALL
IN/OUT/PING packet received by a non-control endpoint.	"1"	None	STALL
IN/OUT/PING packet received by a non-control endpoint.	"0"	None	ACK/NAK/NYET
SETUP packet received by a control endpoint.	N/A	Clear	ACK
IN/OUT/PING packet received by a control endpoint	"1"	None	STALL
IN/OUT/PING packet received by a control endpoint	"0"	None	ACK/NAK/NYET

6. Data toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe. For more information on data toggle, refer to the USB 2.0 specification.

a. Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the ENDPTCTRLx register. This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

b. Data Toggle Inhibit

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the data toggle Inhibit bit active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response

Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification. At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were designed so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3 (transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is designed in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "priming" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be designed properly use priming. Further, note that the term "flushing" is used to describe the action of clearing a packet that was queued for execution.

1. Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH). After the dTD is fetched, it will be stored in the

dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the

host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Since only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section Bandwidth and Latency Issues.

2. Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

3. Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical. All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0,

$$N = \text{INT}(\text{Number Of Bytes}/\text{Max. Packet Length}) + 1$$

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

With Zero Length Termination (ZLT) = 1,
 $N = \text{MAXINT}(\text{Number Of Bytes}/\text{Max. Packet Length})$

Bytes (dTd)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

Note: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints.

The ZLT bit in the dTD will operate as following on BULK and control transfers:

ZLT = 0, the default value, means that the zero length termination is active. With the ZLT option enabled, when the device is transmitting, the hardware will automatically append a zero length packet when the following conditions are true:

- The packet transmitted equals maximum packet length
- The dTD has exhausted the field Total Bytes

After this the dTD will be retired. When the device is receiving, if the last packet length received equal maximum packet length and the total bytes is zero, it will wait for a zero length packet from the host to retire the current dTD.

ZLT = 1, means the zero length termination is inactive. With the ZLT option disabled, when the device is transmitting, the hardware will not append any zero length packets. When receiving, it will not require a zero length packet to retire a dTD whose last packet was equal to the maximum packet length packet. The dTD is retired as soon as Total Bytes field goes to zero, or a short packet is received.

Each transfer is defined by one dTD, so the zero length termination is for each dTD.

In some software application cases, the logic transfer does not fit into just one dTD, so it does not make sense to add a Zero Length Termination packet each time a dTD is consumed. On those cases we recommend to turn off this ZLT feature and use software to generate the zero length termination.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. Total bytes in dTD will equal

zero when this occurs.

- A short packet (number of bytes < maximum packet length) was received. This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining.

From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.

- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

Note: All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

Below is the table for Interrupt/Bulk Endpoint Bus Response Matrix:

Token Type	Stall	Not Primed	Primed	Underflow	Overflow	Not Enabled
Setup	Ignore	Ignore	Ignore	N/A	N/A	BTO
In	STALL	NAK	Transmit	BS Error	N/A	BTO
Out	STALL	NAK	Receive +	N/A	NAK	BTO
			NYET/ACK			
Ping	STALL	NAK	ACK	N/A	N/A	BTO
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	BTO

BS Error – Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

4. Control Endpoint Operation Model

a. Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

Setup Package Handling

The setup lockout mechanism can be disabled and a tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted by an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

Disable Setup Lockout by writing '1' to Setup Lockout Mode (SLOM) in USBMODE. (Once at initialization).

Setup lockout is not necessary when using the tripwire as described below.

After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:

1. Write '1' to clear corresponding bit ENDPTSETUPSTAT.
2. Write '1' to Setup Tripwire (SUTW) in USBCMD register.
3. Duplicate contents of dQH. SetupBuffer into local software byte array.
4. Read Setup TripWire (SUTW) in USBCMD registers. (If set - continue; if cleared - goto 2)
5. Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.
6. Process setup packet using local software byte array copy and execute status/handshake phases.
7. Before priming for status/handshake phases ensure that ENDPTSETUPSTAT is '0'.

A poll loop should be used to wait until ENDPTSETUPSTAT transitions to '0' after step 1. above and before priming for the status/handshake phases.

The time from writing a '1' to ENDPTSETUPSTAT and reading back a '0' is very short (~1-2us) so a poll loop in the DCD will not be harmful.

Note: After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

b. Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by

reading the ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, i.e. the ENDPTPRIME bit goes to zero and the ENDPTSTATUS bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (ENDPTSTATUS) to enforce data coherency with the setup packet. Note: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints. Error handling of data phase packets is the same as bulk packets described previously.

c. Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase. The DCD must also perform the same checks of the ENDPTSETUPSTAT as described above in the data phase.

Note: The MULT field in the dQH must be set to “00” for bulk, interrupt, and control endpoints. Error handling of data phase packets is the same as bulk packets described previously.

d. Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

Token Type	Stall	Not Primed	Primed	Endpoint State	Under	Over	Not Enabled	Setup Lockout
Setup In	ACK STALL	ACK NAK	ACK Transmit	-flow N/A BS Error	-flow SYSERR N/A	BTO BTO	N/A	
Out	STALL	NAK	Receive +	N/A	NAK	BTO	N/A	
Ping Invalid	STALL Ignore	NAK Ignore	NYET/ACK ACK Ignore	N/A Ignore	N/A Ignore	BTO BTO	N/A Ignore	

BS Error – Force Bit Stuff Error

NYET/ACK – NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR – System error should never occur when the latency FIFOs are correctly sized and

the DCD is responsive.

5. Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes. Real time delivery by the device controller will be accomplished by the following:

- Exactly MULT Packets per (u)Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (u)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

Note: If the MULT field is set to more packets than present in the dTD to be transmitted, the controller will send zero length packets to all extra incoming IN tokens and report fulfillment error (transaction error) in current dTD. If more dTDs exist in memory the USB-HS controller will move to the next dTD to be transmitted in the next (u)frame. Because of this behavior it is recommended to always use the correct MULT matching the number of packets to be processed for a given dTD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (u)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (u)frame. Once an ISO transaction is started in a (u)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device

controller will force retire the ISOdTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software to discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the Transaction Error bit and the data is stored as usual for the application software to sort out.

TX Packet Retired:

- MULT counter reaches zero.
- Fulfillment Error [Transaction Error bit is set]

Packets Occurred > 0 AND # Packets Occurred < MULT

Note: For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field. If the

Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

RX Packet Retired:

- MULT counter reaches zero.
- Non-MDATA Data PID is received
- Overflow Error:

Packet received is > maximum packet length. [Buffer Error bit is set]

Packet received exceeds total bytes allocated in dTD. [Buffer Error bit is set]

- Fulfillment Error [Transaction Error bit is set]

Packets Occurred > 0 AND # Packets Occurred < MULT

- CRC Error [Transaction Error bit is set]

Note: For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (u)frame to (u)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (u)frames.

a. Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (u)frame number (FRINDEX register) can be used as a marker. To cause a packet transfer to occur at a specific (u)frame number [N], the DCD should interrupt on SOF during frame N-1. When the FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (u)frame N-1 so that the device controller will execute delivery during (u)frame N.

Caution: Priming an endpoint towards the end of (u)frame N-1 will not guarantee delivery in

(u)frame N. The delivery may actually occur in (u)frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

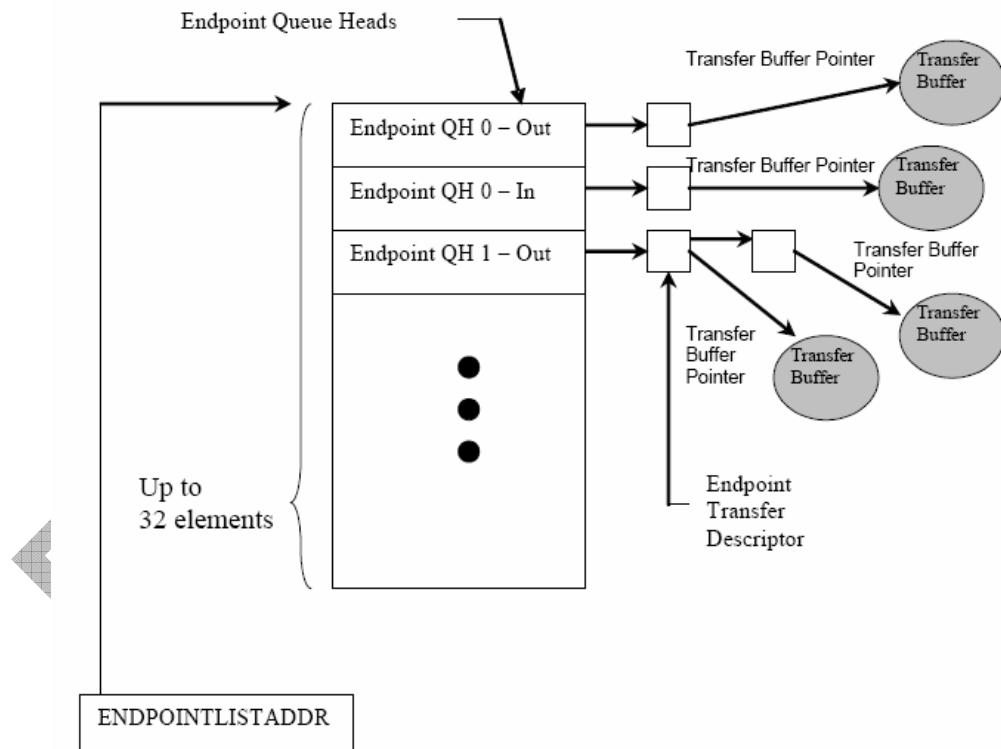
b. Isochronous Endpoint Bus Response Matrix

Token Type	Stall	Not Primed	Primed	Under-flow	Over-flow	Not Enabled
Setup In	STALL	STALL	STALL	N/A	N/A	BTO
	NULL Packet	NULL Packet	Transmit	BS Error	N/A	BTO
Out	Ignore	Ignore	Receive	N/A	Drop Package	BTO
Ping	Ignore	Ignore	Ignore	Ignore	Ignore	BTO
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	BTO

BS Error = Force Bit Stuff Error

NULL Packet = Zero Length Packet

Managing Queue Heads



The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in above figure. The even elements

in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors since pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multiplier, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

1. Queue Head Initialization

One pair of device queue heads must be initialized for each active endpoint. To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the USB Chapter 9 protocol. Note: In FS mode, the multiplier field can only be 1 for ISO endpoints.
- Write the next dTD Terminate bit field to '1'.
- Write the Active bit in the status field to '0'.
- Write the Halt bit in the status field to '0'.

Note: The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

2. Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8- byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

- a. Copy setup buffer contents from dQH - RX to software buffer.
- b. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

Note: The acknowledge must occur before continuing to process the setup packet.

Note: After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH – RX. Only the local software copy should be examined.

- c. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.

Note: It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

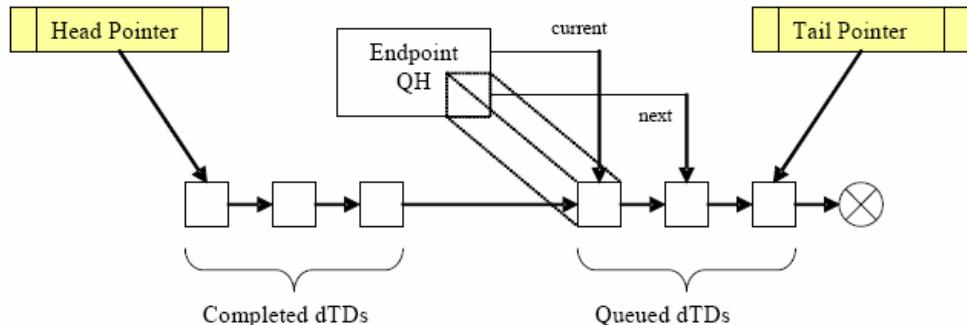
- d. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

Managing Transfers with Transfer Descriptors

1. Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

Note: To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.



2. Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer. Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to “00000”

Write the following fields:

- Initialize first 7 DWords to 0.
- Set the terminate bit to “1”.
- Fill in total bytes with transfer size.
- Set the interrupt on complete if desired.
- Initialize the status field with the active bit set to “1” and all remaining status bits set

to “0”.

- Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
- Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

3. Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty:

Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

Case 1: Link list is empty

1. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
2. Clear active & halt bit in dQH (in case set from a previous error).
3. Prime endpoint by writing ‘1’ to correct bit position in ENDPTPRIME.

Case 2: Link list is not empty

1. Add dTD to end of linked list.
2. Read correct prime bit in ENDPTPRIME – if ‘1’ DONE.
3. Set ATDTW bit in USBCMD register to ‘1’.
4. Read correct status bit in ENDPTPRIME. (store in tmp. variable for later)
5. Read ATDTW bit in USBCMD register.
- If ‘0’ goto 3.
- If ‘1’ continue to 6.
6. Write ATDTW bit in USBCMD register to ‘0’.
7. If status bit read in (4) is ‘1’ DONE.
8. If status bit read in (4) is ‘0’ then Goto Case 1: Step 1.

4. Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On

Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

Caution: Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

Active = 0
 Halted = 0
 Transaction Error = 0
 Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

5. Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one or more endpoints on a USB device reset or during a broken control transfer. There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

- Write a '1' to the corresponding bit(s) in ENDPTFLUSH.
- Wait until all bits in ENDPTFLUSH are '0'.

Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.

- Read ENDPTSTAT to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed: in very rare cases, a packet is in progress to the particular endpoint when commanded flush using ENDPTFLUSH. A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush by repeating steps 1-3 until each endpoint is successfully flushed.

6. Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Error	Direction	Package Type	Data Buffer Error Bit	Transaction Error bit
Overflow**	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1

ISO Fulfillment Error	Both	ISO	0
-----------------------	------	-----	---

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated.

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length. ** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct,
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (u)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (u)frame. During the "dead" (u)frame, the Device Controller reports error on the pipe and primes for the following frame.

Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

1. High-Frequency Interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Execution Order	Interrupt	Action
1a	USB Interrupt** - ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in section Managing Queue Heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol
1b	USB Interrupt** - ENDPTCOMPLETE	Handle completion of dTD as indicated in section Managing

2	SOF Interrupt	Queue Heads Action as deemed necessary by application. This interrupt may not have a use in all applications.
---	---------------	--

** It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

2. Low-Frequency Interrupts

The low frequency events include the following interrupts. These interrupt can be handled in any order since they don't occur often in comparison to the high-frequency interrupts.

Interrupt	Action
Port Change	Change software state information
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary
Reset Received	Change software state information. Abort pending transfers

3. Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core, free transfers buffers in progress and restart the DCD.

OTG Mode Operation

In the previous sections, device mode and for host mode behaviors are described. However, during OTG operations it is necessary to perform tasks independent of the controller mode. Note also from USBMODE register that the only way to transition the controller mode out of host or device mode is with the controller reset bit. Therefore, it is also necessary for the OTG tasks to be performed independent of a controller reset as well as independent of the controller mode.

To this end, the listed below are the register bits that are used for OTG operations, which are independent of the controller mode and are also not affected by a write to the reset bit in the USBCMD register:

- All Identification Registers
- All Device/Host Capability Registers
- OTGSC: All bits
- PORTSC:
 - Physical Interface Select
 - Physical Interface Serial Select
 - Physical Interface Data Width
 - Physical Interface Low Power
 - Physical Interface Wake Signals
 - Port Indicators
 - Port Power

Auto-Reset

When the HAAR is set to one, the host will automatically start a reset after a connect event. This shortcuts the normal process where software is notified of the connect event and starts the reset. Software will still receive notification of the connect event but should not write the reset bit when the HAAR is set. Software will be notified again after the reset is complete via the enable change bit in the PORTSC register which cause a port change interrupt.

This assist will ensure the OTG parameter TB_ACON_BSE0_MAX = 1ms is met.

Data-Pulse

Writing a one to HADP will start a data pulse of approximately 7ms in duration and then automatically cease the data pulsing. During the data pulse, the DP will be set and then cleared. This automation relieves software from accurately controlling the data-pulse duration. During the data pulse, the HCD can poll to see that the HADP and DP bit have returned low to recognize the completion or simply launch the data pulse and wait to see if a VBUS Valid interrupt occurs when the A-side supplies bus power.

This assist will ensure data pulsing meets the OTG requirement of > 5ms and < 10ms.

B-Disconnect to A-Connect

During HNP, the B-disconnect occurs from the OTG A_suspend state and within 3 ms, the A-device must enable the pullup on the DP leg in the A-peripheral state. When HABA is set, the Host Controller port is in suspend mode, and the device disconnects, then this hardware assist begins.

- Reset the OTG core.
- Write the OTG core into device mode.
- Write the device run bit to a ‘1’ and enable necessary interrupts including:
 - USB Reset Enable (URE); enables interrupt on usb bus reset to device
 - Sleep Enable (SLE); enables interrupt on device suspend
 - Port Change Detect Enable (PCE); enables interrupt on device connect

When software has enabled this hardware assist, it must not interfere during the transition

and should not write any register in the core until it gets an interrupt from the device controller signifying that a reset interrupt has occurred or at least first verify that the core has entered device mode. HCD/DCD must not activate the core soft reset at any time since this action is performed by hardware. During the transition, the software may see an interrupt from the disconnect and/or other spurious interrupts (i.e. SOF/etc.) that may or may not cascade and may be cleared by the soft reset depending on the software response time.

After the core has entered device mode by the hardware assist, the DCD must ensure that the

ENDPTLISTADDR is programmed properly before the host sends a setup packet. Since the end of the reset duration, which may be initiated quickly (a few microseconds) after connect, will require at a minimum 50 ms, this is the time for which the DCD must be ready to accept setup packets after having received notification that the reset has been detected or simply that the OTG is in device mode which ever occurs first.

In the case where the A-peripheral fails to see a reset after the controller enters device mode and engages the DP-pullup, the device controller interrupt the DCD signifying that a suspend has occurred.

This assist will ensure the parameter TA_BDIS_ACON_MAX = 3ms is met.

Chapter 14 SD/MMC/SDIO Host Controller

Overview

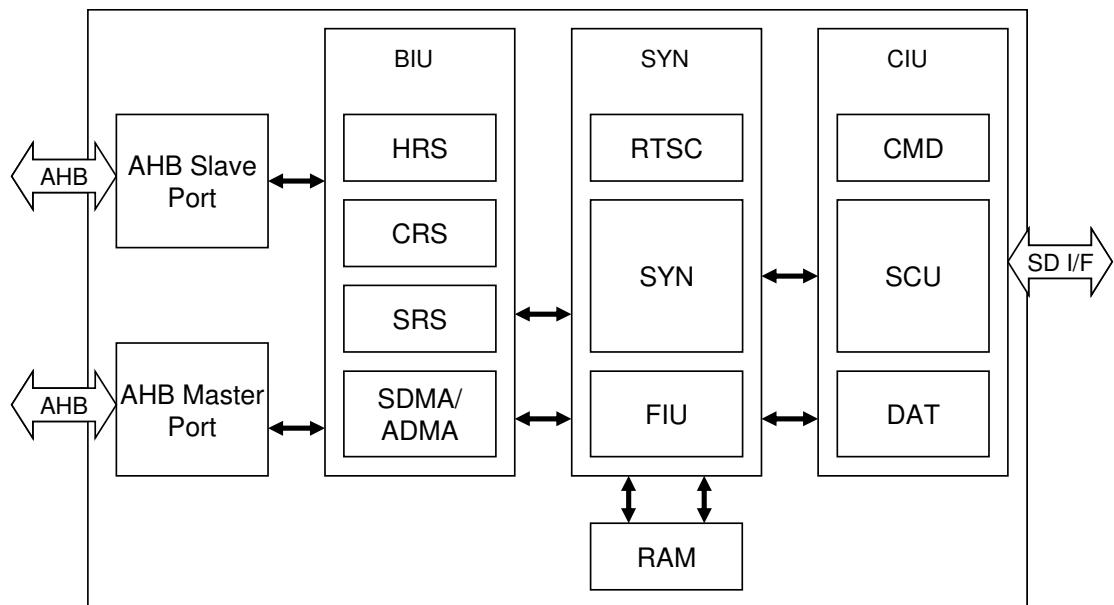
This controller is compatible with SDIO Host controller Specification ver 2.00. The host controller services SD Memory ver 2.00, MMC ver 4.2, and SDIO 2.00 type cards. It provides an interface with SD and MMC memory card for date storage and also provides an interface to extend function through SDIO interface.

Key Features

- SD Memory Card version 2.00 (including SDHC)
- SDIO Card version 2.00
- MMC Card version 4.2 (including MMCplus)
- SDIO Host Specification version 2.00
- SD1/SD4/MMC-8 modes of operation
- Suspend/Resume mechanism for SDIO cards
- Read Wait mechanism for SDIO cards

Architecture

Block Diagram



Block Descriptions

BIU

Bus Interface Unit. Contains several subcomponents working in the clk System clock

domain. It is responsible for communication with the host CPU. The SRS slave interface provides the access to the internal register spaces, including Slot Register Set (SRS), Common Register Set (CRS), and proprietary Host Register Set (HRS). The DMA master interface is optional and used for Direct Memory Access.

- SRS – Slot Register Set. Implements software-accessible registers which are independent for each slot.
- CRS – Common Register Set. Implements software-accessible registers which are common for all slots.
- HRS – Host Register Set. Implements software-accessible host configuration registers.
- DMA – Integrated DMA controller.

CIU

Card Interface Unit. Contains several subcomponents are working in the sdmclk clock domain. It is responsible for communication with the SD/SDIO/MMC cards using SD bus interface. It contains card clock dividers, Command/Response generation logic (CMD), and SD1/SD4 datapath logic (DAT). Most of the components are shared among all slots in order to reduce the area. Components that are independent for each slot are grouped in a Slot Control Unit (SCU).

- CMD – CMD line control module. Controls the command generation and response checking logic.
- DAT – DAT line control module. Controls the data transfer logic.
- SCU – Slot Control Unit. Contains logic which is independent for each slot.
 - Contains three additional subcomponents:

SYN

Cross Clock domain synchronization logic. Provides the reliable cross clock domain synchronization for all control paths inside the SDIO-HOST.

FIU

FIFO Interface Unit. Contains the data buffer control logic used for the data transactions. There can be two virtual buffers implemented inside on-chip RAM. One of them can be accessed by the BIU side, while the other can be accessed by the CIU at the same time.

RST

Reset Controller. Generates reset signals for every internal block of the SDIO-HOST. There are 4 reset factors implemented inside SDIO-HOST: hardware reset, software reset for all (clears all flip-flops except card detection logic), software reset for CMD (clears command/response logic), and software reset for DAT (clears the datapath logic).

Registers

The register set is divided into 3 spaces:

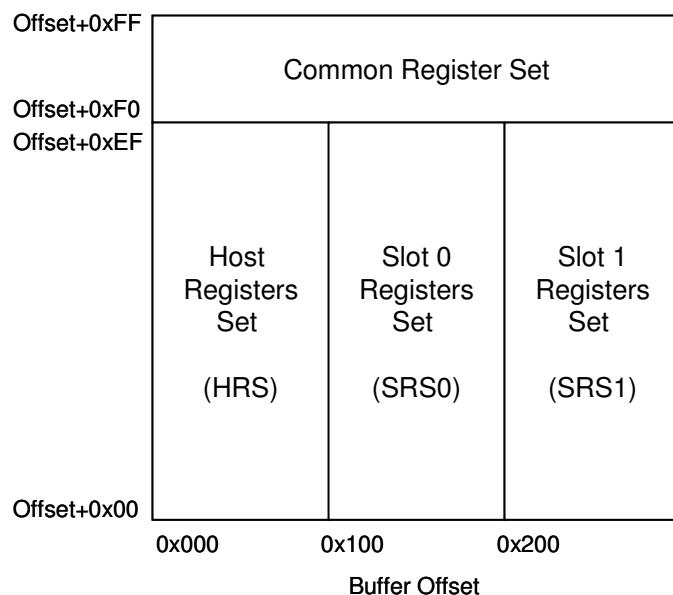
Host Register Set (HRS) – Contains registers specific for this design, and not covered by the SDIO Host Specification. It occupies the memory from address 0x000 to 0x0EF.

Slot Register Set (SRS) – Contains registers specific for a single card slot. There is a

separate register set per each card slot. The first slot (obligatory) occupies memory from address 0x100 to 0x1EF. The second slot occupies memory from 0x200 to 0x2EF. The SRS area is defined by the SDIO Host Specification.

Common Register Set (CRS) – Contains registers common for all card slots. It can be seen when accessing SDIO-HOST with any base offset address. This means that the CRS area may be accessed within 0x1F0-0x1FF, as well as within 0x2F0-0x2FF, and so on. The CRS area is defined by the SDIO Host Specification.

Below diagram shows register area organization.



Registers Summary

Host Register Set

Name	Offset	Size	Reset Value	Description
SDC_HRS0	0x0000	W	0x00030000	General information register
SDC_HRS1	0x0004	W	0x00000005	Debounce setting register
SDC_HRS2	0x0008	W	0x00000000	Bus setting register
SDC_HRS4	0x0010	W	0x00000000	HWInit SRS16 configuration for Slot #0
SDC_HRS6	0x0018	W	0x00000000	HWInit SRS18 configuration for Slot #0
SDC_HRS8	0x0020	W	0x00000000	HWInit SRS16 configuration for Slot #1
SDC_HRS10	0x0028	W	0x00000000	HWInit SRS18 configuration for Slot #1

SDC_HRS20	0x0050	W	0x00000000	CPRM information/setting register
SDC_HRS21	0x0054	W	0x001F0000	CPRM CBC setting register

Slot Register Set

Name	Offset	Size	Reset Value	Description
SDC_SRS0	0x0000	W	0x00000000	System address
SDC_SRS1	0x0004	W	0x00000000	Block count and size
SDC_SRS2	0x0008	W	0x00000000	Argument
SDC_SRS3	0x000C	W	0x00000000	Transfer mode and command information
SDC_SRS4	0x0010	W	0x00000000	Response #0
SDC_SRS5	0x0014	W	0x00000000	Response #1
SDC_SRS6	0x0018	W	0x00000000	Response #2
SDC_SRS7	0x001C	W	0x00000000	Response #3
SDC_SRS8	0x0020	W	0x00000000	Buffer data port
SDC_SRS9	0x0024	W	0x01FA0000	Present state
SDC_SRS10	0x0028	W	0x00000000	Host control settings #0
SDC_SRS11	0x002C	W	0x00000000	Host control settings #1
SDC_SRS12	0x0030	W	0x00000000	Interrupt status
SDC_SRS13	0x0034	W	0x00000000	Interrupt status enable
SDC_SRS14	0x0038	W	0x00000000	Interrupt signal enable
SDC_SRS15	0x003C	W	0x00000000	Auto CMD12 Error status
SDC_SRS16	0x0040	W	0x07FA32B2	Capabilities
SDC_SRS18	0x0048	W	0x001F0F08	Maximum current capabilities
SDC_SRS20	0x0050	W	0x00000000	Force event
SDC_SRS21	0x0054	W	0x00000000	ADMA error status
SDC_SRS22	0x0058	W	0x00000000	ADMA system address

Common Register Set

Name	Offset	Size	Reset Value	Description
SDC_CSR63	0x00FC	W	0x00010000	Host version/interrupt status

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

SDC_HRS0

Address: Operational Base + offset (0x0000)

HRS0 General Information Register

Bit	Attr	Reset Value	Description
31:18	-	-	Reserved.
17	R	0x1	S1AV; Slot #1 available. 1 = slot is available. 0 = slot is not available.

16	R	0x1	S0AV; Slot #0 available. 1 = slot is available. 0 = slot is not available.
15:1	-	-	Reserved.
0	RW	0x0	SWR; Software reset. When set to 1, the entire SDIO-HOST is reset. The difference between SWR and SRS11.SRFA is that the latter bit is dedicated to the single slot, while the SWR resets all flip-flops in every slot. After completing the reset operation, SWR bit is automatically cleared. It takes some time to complete the reset operation, so the software should always wait until SWR=0, and continue the other operations only when SWR=0.

SDC_HRS1

Address: Operational Base + offset (0x0004)

HRS1 Debounce Setting Register

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
23:0	RW	0x5	DP; Debounce Period. Defines the number of clk clock cycles used for the debouncing circuit which detects card detection events. The debounce period is equal to: $DP * HCLK \text{ Period}$ Typically, DP value should be chosen to obtain the period of 20ms.

SDC_HRS2

Address: Operational Base + offset (0x0008)

HRS2 Bus Setting Register

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.

3:0	RW	0x0	PBL; Programmable Burst Length. Defines the maximum number of beats within single DMA burst.																										
			<table> <thead> <tr> <th>PBL</th> <th>Burst Length</th> </tr> </thead> <tbody> <tr><td>0001</td><td>1</td></tr> <tr><td>0010</td><td>2</td></tr> <tr><td>0011</td><td>4</td></tr> <tr><td>0100</td><td>8</td></tr> <tr><td>0101</td><td>16</td></tr> <tr><td>0110</td><td>32</td></tr> <tr><td>0111</td><td>64</td></tr> <tr><td>1000</td><td>128</td></tr> <tr><td>1001</td><td>256</td></tr> <tr><td>1010</td><td>512</td></tr> <tr><td>1011</td><td>1024</td></tr> <tr><td>Others</td><td>2048</td></tr> </tbody> </table>	PBL	Burst Length	0001	1	0010	2	0011	4	0100	8	0101	16	0110	32	0111	64	1000	128	1001	256	1010	512	1011	1024	Others	2048
PBL	Burst Length																												
0001	1																												
0010	2																												
0011	4																												
0100	8																												
0101	16																												
0110	32																												
0111	64																												
1000	128																												
1001	256																												
1010	512																												
1011	1024																												
Others	2048																												

SDC_HRSx (x=4, 6, 8, 10)

Address: Operational Base + offset (0x0010, 0x0018, 0x0020, 0x0028)

HWInit SRS16/SRS18 Configuration

Registers in this area work as "mirror" versions of SRS16/SRS18 registers defined by the SDIO Host Specification. The HRS registers are "write-only", while SDC_SRS16/SDC_SRS18 corresponding to them are "read-only". The value written to the HRS field updates the corresponding SRS value.

The purpose is to allow changing the SRS16/SRS18 values dynamically during the normal operation, while preserving the compatibility with SDIO Host Specification. The SDIO Host Specification requires SDC_SRS16/18 registers to be read-only.

The software can change the SDC_HRS4-18 values according to the target system parameters during the system power-up procedure.

Bit	Attr	Reset Value	Description
31:0	RW	0x0	The value written to the HRS field updates the corresponding SDC_SRS value. Read operations on the SDC_HRS registers are ignored.

SDC_HRS20

Address: Operational Base + offset (0x0050)

HRS20 CPRM Information/Settings Register

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31	R	0x0	CPRM_BUSY; CPRM busy 1 = CPRM is busy. 0 = CPRM is ready to execute next command. (Read Clear)																														
30:24	-	-	Reserved.																														
23:16	R	0x0	CPRM_CCI_OUT; Copy Control Information Output The CCI calculated during the Title Key decryption. (Read Clear)																														
15:8	RW	0x0	CPRM_CCI_IN; Copy Control Information Input The CCI that is to be encrypted together with a random Title Key.																														
7	RW	0x0	CPRM_EN; CPRM enable 1 = Enables CPRM operation (All card read/write are affected). 0 = CPRM does not affect any operation.																														
6	RW	0x0	CPRM_HALT; CPRM halt 1 = Halts the operation of CRPM module (Only in case of MKB processing and CBC enabled transfers). 0 = normal CPRM operation.																														
5:4	-	-	Reserved.																														
3:0	RW	0x0	CPRM_CMD; CPRM command The index of CPRM command. <table> <tr> <td>CRPM_CMD</td> <td>Command mnemonic</td> </tr> <tr> <td>1</td> <td>PROCESS_MKB</td> </tr> <tr> <td>2</td> <td>PROCESS_MKB_EXT</td> </tr> <tr> <td>3</td> <td>CALC_KMU</td> </tr> <tr> <td>4</td> <td>AKE</td> </tr> <tr> <td>5</td> <td>SET_BIND_ID</td> </tr> <tr> <td>8</td> <td>GEN_KT</td> </tr> <tr> <td>9</td> <td>CALC_KT</td> </tr> <tr> <td>A</td> <td>GEN_KT_BIND</td> </tr> <tr> <td>B</td> <td>CALC_KT_BIND</td> </tr> <tr> <td>C</td> <td>DECRYPT_USER</td> </tr> <tr> <td>D</td> <td>ENCRYPT_USER</td> </tr> <tr> <td>E</td> <td>DECRYPT_PROT</td> </tr> <tr> <td>F</td> <td>ENCRYPT_PROT</td> </tr> <tr> <td>Others</td> <td>NOP</td> </tr> </table>	CRPM_CMD	Command mnemonic	1	PROCESS_MKB	2	PROCESS_MKB_EXT	3	CALC_KMU	4	AKE	5	SET_BIND_ID	8	GEN_KT	9	CALC_KT	A	GEN_KT_BIND	B	CALC_KT_BIND	C	DECRYPT_USER	D	ENCRYPT_USER	E	DECRYPT_PROT	F	ENCRYPT_PROT	Others	NOP
CRPM_CMD	Command mnemonic																																
1	PROCESS_MKB																																
2	PROCESS_MKB_EXT																																
3	CALC_KMU																																
4	AKE																																
5	SET_BIND_ID																																
8	GEN_KT																																
9	CALC_KT																																
A	GEN_KT_BIND																																
B	CALC_KT_BIND																																
C	DECRYPT_USER																																
D	ENCRYPT_USER																																
E	DECRYPT_PROT																																
F	ENCRYPT_PROT																																
Others	NOP																																

SDC_HRS21

Address: Operational Base + offset (0x0054)

HRS21 CPRM CBC Settings Register

Bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20:16	RW	0x1F	SD_BIND_BLOCK; CBC block size. If SD_BIND_BLOCK = 11111b the whole file is considered as a single CBC block; otherwise the file is encrypted/decrypted in CBC mode with a block size equal to 64*2SD_BIND_BLOCK bytes. Each block starts a new CBC chain.
15:4	-	-	Reserved.
3:0	RW	0x0	SD_BIND_HEADER; Unencrypted header size. If SD_BIND_HEADER = 0000b, no header; otherwise the device passes first 64*2SD_BIND_HEADER -1 bytes (header) without processing.

SDC_SRS0

Address: Operational Base + offset (0x0000)

SRS0 System Address Register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	SDMA SA; SDMA System Address. Memory address for the DMA transactions. It should be set by the software before sending command which uses SDMA, or after SDMA interrupt (SRS12.DMAINT). The software should not read or write SDMA SA when the SDMA transaction is in progress. After SDMA is stopped (finishing data transfer command or DMA interrupt), SDMA SA points to the next system address of the next contiguous data position.

SDC_SRS1

Address: Operational Base + offset (0x0004)

SRS1 System Address Register

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:16	RW	0x0	<p>BCCT; Blocks Count For Current Transfer. Contains the number of data blocks to be transferred. It is decremented after each block transfer if Block Count is enabled (SRS3.BCE = 1). If SRS3.BCE = 0 then the value written to BCCT is ignored by the SDIO-HOST. Setting the BCCT to 0 results in no data blocks being transferred. This register should be accessed only when no data transfer is in progress. During the data transfer, read operations on this register may return an invalid value, and write operations are ignored. This register is enabled when Block Count Enable in the SRS3 register is set to 1 and is valid only for multiple block transfers.</p> <table> <thead> <tr> <th>BCCT</th><th>Number of Blocks</th></tr> </thead> <tbody> <tr> <td>0000h</td><td>0</td></tr> <tr> <td>0001h</td><td>1</td></tr> <tr> <td>.....</td><td>.....</td></tr> <tr> <td>FFFFh</td><td>65535</td></tr> </tbody> </table>	BCCT	Number of Blocks	0000h	0	0001h	1	FFFFh	65535
BCCT	Number of Blocks												
0000h	0												
0001h	1												
.....												
FFFFh	65535												
15	-	-	Reserved.										

14:12	RW	0x0	<p>HDMABB; Host DMA Buffer Boundary. Contains the size of the contiguous buffer in the system memory. When the SDMA reaches the buffer boundary, it stops the SDMA transaction and generates the DMA Interrupt (SRS12.DMAINT). After the DMA Interrupt, the software should write new SDMA System Address (SRS0) or set Continue Request (SRS10.CR) in order to resume the SDMA transaction. If Continue Request is used without changing SRS0, then the SDMA starts at the next address after the buffer boundary.</p> <table> <thead> <tr> <th>HDMABB</th><th>Buffer Boundary</th></tr> </thead> <tbody> <tr> <td>000b</td><td>4KB</td></tr> <tr> <td>001b</td><td>8KB</td></tr> <tr> <td>010b</td><td>16KB</td></tr> <tr> <td>011b</td><td>32KB</td></tr> <tr> <td>100b</td><td>64KB</td></tr> <tr> <td>101b</td><td>128KB</td></tr> <tr> <td>110b</td><td>256KB</td></tr> <tr> <td>111b</td><td>512KB</td></tr> </tbody> </table>	HDMABB	Buffer Boundary	000b	4KB	001b	8KB	010b	16KB	011b	32KB	100b	64KB	101b	128KB	110b	256KB	111b	512KB
HDMABB	Buffer Boundary																				
000b	4KB																				
001b	8KB																				
010b	16KB																				
011b	32KB																				
100b	64KB																				
101b	128KB																				
110b	256KB																				
111b	512KB																				
11:0	RW	0x0	<p>TBS; Transfer Block Size. Specifies the block size for block data transfers. Values ranging from 1 up to the maximum buffer size can be set. TBS can be accessed only if no data transfer is in progress. Read operations during the data transfer may return an invalid value, and write operations are ignored. TBS should never exceed the FIFO buffer size physically implemented inside the SDIO-HOST. The FIFO buffer size is 512 Bytes</p>																		

SDC_SRS2

Address: Operational Base + offset (0x0008)

SRS2 Argument Register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	ARG; Command argument. Contains bits [39:8] of command argument.

SDC_SRS3

Address: Operational Base + offset (0x000C)

SRS3 Transfer Mode & Command Information Register

The SRS3 register contains the information needed by the SDIO-HOST to initiate the transaction on the SD interface. The software will check the Present State Register (SRS9) before writing the SRS3. Writing the SRS3.CI (Command Index) field triggers the actual command generation.

The upper part of SRS3 [31:16] should be written only if SRS9.CICMD=0 (i.e. no command transmission is already in progress). Additionally, the command that uses DAT line can only be sent if SRS9.CIDAT=0 (i.e. DAT line is not used at a given time). Commands that use DAT line include commands with BUSY response (SFR3.RTS=11b) and commands with data transfer (SFR3.DPS=1).

Note that the upper part of SRS3 is not hardware-protected, so it is the responsibility of the software to decide if the write operation is allowed at a given time.

The lower part of SRS3 [15:0] should be written only if SRS9.CIDAT=0. The lower part of SRS3 is hardware-protected from writing when SRS9.CIDAT=1.

Bit	Attr	Reset Value	Description										
31:30	-	-	Reserved.										
29:24	RW	0x0	<p>CI; Command Index. Contains an index of the command to be sent. The index should be in range 0-63, corresponding directly to CMD0-CMD63 (or ACMD0-ACMD63) commands as shown below:</p> <table> <tbody> <tr> <td>CI</td> <td>Command Mnemonic</td> </tr> <tr> <td>000000b</td> <td>CMD0/ACMD0</td> </tr> <tr> <td>000001b</td> <td>CMD1/ACMD1</td> </tr> <tr> <td>.....</td> <td></td> </tr> <tr> <td>111111b</td> <td>CMD63/ACMD63</td> </tr> </tbody> </table> <p>Writing the Command Index triggers the actual command generation. This field should be written only when Command Inhibit CMD bit is 0 in Present State Register (SRS9).</p> <p>To check the list of available commands, please refer to the appropriate Card specifications (SD, SDIO, or MMC</p>	CI	Command Mnemonic	000000b	CMD0/ACMD0	000001b	CMD1/ACMD1		111111b	CMD63/ACMD63
CI	Command Mnemonic												
000000b	CMD0/ACMD0												
000001b	CMD1/ACMD1												
.....													
111111b	CMD63/ACMD63												

			Specifications).																		
23:22	RW	0x0	<p>CT; Command Type. There are 4 command types:</p> <table> <tr> <td>CT</td> <td>Command Type</td> </tr> <tr> <td>11b</td> <td>Abort Command.</td> </tr> <tr> <td>10b</td> <td>Used when the software wants to stop the current data transfer (read or write data transfer). After sending an Abort Command, the software will also issue the software reset.</td> </tr> <tr> <td>01b</td> <td>Suspend Command.</td> </tr> <tr> <td>00b</td> <td>Can be used by the software to release the SD bus.</td> </tr> <tr> <td></td> <td>Resume Command.</td> </tr> <tr> <td></td> <td>Used to restart the transaction previously suspended by the Suspend Command.</td> </tr> <tr> <td></td> <td>Normal Command.</td> </tr> <tr> <td></td> <td>Covers all the remaining commands.</td> </tr> </table>	CT	Command Type	11b	Abort Command.	10b	Used when the software wants to stop the current data transfer (read or write data transfer). After sending an Abort Command, the software will also issue the software reset.	01b	Suspend Command.	00b	Can be used by the software to release the SD bus.		Resume Command.		Used to restart the transaction previously suspended by the Suspend Command.		Normal Command.		Covers all the remaining commands.
CT	Command Type																				
11b	Abort Command.																				
10b	Used when the software wants to stop the current data transfer (read or write data transfer). After sending an Abort Command, the software will also issue the software reset.																				
01b	Suspend Command.																				
00b	Can be used by the software to release the SD bus.																				
	Resume Command.																				
	Used to restart the transaction previously suspended by the Suspend Command.																				
	Normal Command.																				
	Covers all the remaining commands.																				
21	RW	0x0	<p>DPS; Data Present Select. Should be set to 1 for commands which transfer data (i.e. read or write data from the card using DAT line). Should be 0 for all other commands, including:</p> <ul style="list-style-type: none"> - Commands using only CMD line - Commands with busy (SRS3.RTS=11b) - Resume type Commands (SRS3.CT=10b) 																		
20	RW	0x0	<p>CICE; Command Index Check Enable. When set to 1, the SDIO-HOST checks if the Command index field of the response is equal to the SRS3.CI value. When 0, the check is not performed and Command index field of the response is ignored.</p>																		
19	RW	0x0	CRCCE; Command CRC Check Enable.																		

			<p>When set to 1, the SDIO-HOST checks if the CRC field of the response is valid.</p> <p>When 0, the CRC check is disabled and the CRC field of the response is ignored.</p> <p>The CRC check should be disabled for responses which do not contain an actual CRC value (some responses contain all 1's in place of the CRC field), and enabled for all other kinds of responses.</p>										
18	-	-	Reserved.										
17:16	RW	0x0	<p>RTS; Response Type Select.</p> <p>Defines the expected response length as below:</p> <table> <tr> <td>RTS</td><td>Response Type</td></tr> <tr> <td>00b</td><td>No response</td></tr> <tr> <td>01b</td><td>136-bit response</td></tr> <tr> <td>10b</td><td>48-bit response</td></tr> <tr> <td>11b</td><td>48-bit response with BUSY</td></tr> </table> <p>Every command implies one of the response types listed above. To check the response type corresponding to a given command, please refer to the appropriate Card specifications (SD, SDIO, or MMC Specifications).</p>	RTS	Response Type	00b	No response	01b	136-bit response	10b	48-bit response	11b	48-bit response with BUSY
RTS	Response Type												
00b	No response												
01b	136-bit response												
10b	48-bit response												
11b	48-bit response with BUSY												
15:6	-	-	Reserved.										
5	RW	0x0	<p>MSBS; Multi/Single Block Select.</p> <p>When set to 1, indicates multi block data transfer. When 0, indicates single block transfer.</p> <p>This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS9). When SRS9.CIDAT=1, all writes to this field are ignored.</p>										
4	RW	0x0	<p>DTDS; Data Transfer Direction Select.</p> <p>Selects between read and write transactions for commands which transfer data (i.e. with SRS3.DPS=1).</p> <p>Should be set to 1 for data read operations (data read from the card to the host).</p> <p>Should be 0 for data write operations (data write from the host to the card).</p> <p>For commands which do not transfer data (SRS3.DPS=0), DTDS is treated as a don't care by the SDIO-HOST.</p> <p>This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS9). When SRS9.CIDAT=1, all writes to this field are ignored.</p>										
3	-	-	Reserved.										
2	RW	0x0	ACE; Auto CMD12 Enable.										
			When set to 1, the Host Controller issues CMD12										

			automatically when last block of multi-block transfer is completed. Will be 0 for all commands which do not require using CMD12 to stop data transfer. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS9). When SRS9.CIDAT=1, all writes to this field are ignored.
1	RW	0x0	BCE; Block Count Enable. When set to 1, the SFR1.BCCT Block Count is automatically decremented after each data block transferred on DAT line. When 0, block counting is disabled, and SFR1.BCCT retains its value. This is useful in executing an infinite transfer. The software would use an explicit ABORT type command to stop the infinite transaction. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS9). When SRS9.CIDAT=1, all writes to this field are ignored.
0	RW	0x0	DMAE; DMA Enable. When set to 1, it enables DMA functionality. DMA can be enabled only if it is supported as indicated in the DMA Support in the SFR16 register. If DMA is not supported, this bit is treated as a don't care bit. This field is hardware-protected by Command Inhibit DAT bit in Present State Register (SRS9). When SRS9.CIDAT=1, all writes to this field are ignored.

SDC_SRS4/SDC_SRS5/SDC_SRS6/SDC_SRS7

Address: Operational Base + offset (0x0010/0x0014/0x0018/0x001C)

SRS4~ SRS7 Response Register

This SRS4 ~ SRS7 registers store the response returned by the card. The mapping of the actual card response and the SRS4 ~ SRS7 contents depends on the type of response. The type of response is determined by the SRS3.RTS field (Response Type) for all user-defined commands. The separate case is the Auto-CMD12 response (called R1b in the SD Memory Specification). Auto-CMD12 response is handled by the SDIO-HOST automatically and goes to the SRS7 register regardless of the SRS3.RTS value.

Below table describes Response from Card Mapped to the SRS Response Register fields.

Type of Response	SRS3. RTS	Card Response	SRS Response Bits
Auto-CMD12	-	R[39:8]	RESP[127:96]
No response	00b	-	-

136-bit	01b	R[127:8]	RESP[119:0]	SRS7, SRS6, SRS5, SRS4
48-bit	10b	R[39:8]	RESP[31:0]	SRS4
48-bit with BUSY	11b	R[39:8]	RESP[31:0]	SRS4

SDC_SRS4

Bit	Attr	Reset Value	Description
31:0	R	0x0	RESP0; (Read clear) [31:0] of command response (RESP).

SDC_SRS5

Bit	Attr	Reset Value	Description
31:0	R	0x0	RESP1; (Read clear) [63:32] of command response (RESP).

SDC_SRS6

Bit	Attr	Reset Value	Description
31:0	R	0x0	RESP2; (Read clear) [95:64] of command response (RESP).

SDC_SRS7

Bit	Attr	Reset Value	Description
31:0	R	0x0	RESP3; (Read clear) [127:96] of command response (RESP).

SDC_SRS8

Address: Operational Base + offset (0x0020)

SRS8 Buffer Data Port Register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	BDP; Buffer Data Port. Used to access data blocks for non-DMA transfers. 8-bit, 16-bit, or 32-bit access to SRS8 is possible with the following restrictions: <ul style="list-style-type: none"> - Only sequential contiguous access in Little Endian mode is possible. For example, if the software accesses SRS8(7..0), then the next transfer will access SRS8.(15:8). No byte skipping is allowed. - Each new block will start at the least significant byte of SRS8, that is SRS8(7:0). - If the block size is not a multiple of 32-bits, and the software accesses SRS8 using 32-bit words, then the excess bytes of the last word are ignored. This allows the software driver to use only 32-bit data

			transfers regardless of the block size.	
Below is the table with all transfers (Byte enable patterns) which are allowed on SRS8:				
Transfer Width	be(3) BDP[31: 24]	be(2) BDP[23: 16]	be(1) BDP[15: 8]	be(0) BDP[7:0]
32-bit	1	1	1	1
16-bit	0	0	1	1
16-bit	1	1	0	0
8-bit	0	0	0	1
8-bit	0	0	1	0
8-bit	0	1	0	0

SDC_SRS9

Address: Operational Base + offset (0x0024)

SRS9 Present State Register

Bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	R	0x0	CMDSL; CMD Line Signal Level. The value is equal to the actual signal level on CMD line of the SD interface (SDCx_CMD). Is useful for debugging purposes.
23:20	R	0x0	DATSL; DAT[3:0] Line Signal Level. The value is equal to the actual signal level on DAT pins of the SD interface (SDCx_DATA): SFR9.23 – SDCx_DATA[3] pin level. SFR9.22 – SDCx_DATA[2] pin level. SFR9.21 – SDCx_DATA[1] pin level. SFR9.20 – SDCx_DATA[0] level. Is useful for debugging purposes.
19	R	0x0	WPSL; Write Protect Switch Pin Level. The value is equal to the actual signal level on Write Protect pin of the SD interface (SDCx_WP). 1 - Write operation is enabled. 0 - Write operation is disabled.
18	R	0x0	CDL; Card Detect Pin Level. The value is equal to the inverted signal level on Card Detect pin of the SD interface (SDCx_CD). 1 – Card is inserted. 0 – No card is inside the slot. Debouncing is not performed on CDL, therefore the use of Card Inserted (SFR9.CI) bit is recommended during normal work. CDL bit is useful only for debugging purposes.

17	R	0x0	CSS; Card State Stable. Indicates if Card Detect Pin Level (CDL) is stable. 1 – CDL value is stable. 0 – CDL is not stable (during card insertion/removal or during the reset). Is useful for debugging purposes.
16	R	0x0	CI; Card Inserted. Indicates if the card is inserted inside the slot. 1 – Card is inserted. 0 – No card is inside the slot. Unlike CDL, value of CI bit is guaranteed to be stable (i.e. de-bouncing is performed on this bit). Use of this bit is recommended during the normal operation of SDIO-HOST.
15:12	-	-	Reserved.
11	R	0x0	BRE; Buffer Read Enable. (Read clear) Shows the current data buffer (SRS8) state during non-DMA data read transfers. 1 – Valid data can be read from the data buffer. 0 – No valid data inside the data buffer. After reading the entire data block, this changes to 0.
10	R	0x0	BWE; Buffer Write Enable. (Read clear) Shows the current data buffer state during non-DMA data write transfers. 1 – Data can be written to the data buffer. 0 – Data cannot be written. After reading the entire data block, this changes to 0.
9	R	0x0	RTA; Read Transfer Active. (Read clear) Indicates the status of the read data transfer. 1 – Data read transfer is in progress. 0 – No read transfer is in progress. The SDIO-HOST sets RTA to 1 after sending the read command, or after restarting the read transfer by the Continue Request (SRS10.CR). This is cleared by the hardware after the last block of the read transfer, or after stopping the read transfer by the Stop at Block Gap Request (SRS10.SBGR). In both cases, the entire data has to be read by the system from the data buffer before clearing RTA bit. In other words, RTA=0 means that the entire data is already transferred to the system, and internal data buffer is empty. In the case of ADMA transfers, the end of read transfer is

			designated by the END flag in a descriptor.
8	R	0x0	<p>WTA; Write Transfer Active. (Read clear) Indicates the status of the write data transfer.</p> <p>1 – Data write transfer is in progress. 0 – No write transfer is in progress.</p> <p>The SDIO-HOST sets WTA to 1 after sending the write command, or after restarting the write transfer by the Continue Request (SRS10.CR).</p> <p>This is cleared by the hardware after the last block of the write transfer, or after stopping the write transfer by the Stop at Block Gap Request (SRS10.SBGR).</p> <p>In both cases, the entire data has to be transferred to the card from the internal data buffer before clearing WTA bit. In other words, RTA=0 means that the entire data is already transferred to the card, and CRC response for the last data block is already received.</p> <p>In the case of ADMA transfers, the end of write transfer is designated by the END flag in a descriptor.</p>
7:3	-	-	Reserved.
2	R	0x0	<p>DAT Line Active. (Read clear) Indicates if the DAT lines of SD interface are currently in use.</p> <p>1 – DAT lines are active (in use). 0 – DAT lines are released (not in use).</p> <p>This is set when Read or Write Transfer bits are active (SRS9.RTA=1 or SRS9.WTA=1), or if the card indicates busy state on the DAT lines.</p> <p>The card can become busy immediately after the write operation, or after command which requires "response with busy".</p> <p>Falling edge of this bit (change from 1 to 0) directly triggers Transfer Complete bit (SRS12.TC).</p>
1	R	0x0	<p>CIDAT; Command Inhibit DAT. (Read clear) Indicates if the SDIO-HOST can issue a command which uses DAT line. Commands which use DAT line include write and read data commands and commands with busy response.</p> <p>1 – Command using DAT line cannot be sent. 0 – Command using DAT line can be sent.</p>

			<p>When CIDAT=1 then the SRS3.(15..0) is write-protected. The software can write SRS3.(15..0) only when CIDAT=0.</p> <p>The behavior of this bit is the same as the behavior of SRS9.DATLA bit (i.e. value of DATLA is equal to the value of CIDAT).</p>
0	R	0x0	<p>CICMD; Command Inhibit CMD. (Read clear) Indicates if the SDIO-HOST can issue a command.</p> <p>1 – Command cannot be sent. 0 – Command can be sent.</p> <p>If this bit is 0, indicates the CMD line is not in use and the Host Controller can issue an SD command using the CMD line. This bit is set immediately after the SRS3.CI is written, indicating start of command transmission. This bit is cleared when the command response is received.</p> <p>Even if the Command Inhibit DAT is set to 1, commands using only the CMD line can be issued if the Command Inhibit CMD is 0.</p>

SDC_SRS10

Address: Operational Base + offset (0x0028)

SRS10 Host Control Settings #0 Register

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26	RW	0x0	WCR; Wakeup Event Enable On SD Card Removal. When set to 1, enables wakeup event via Card Removal assertion in the SRS12 register.
25	RW	0x0	WCI; Wakeup Event Enable On Card Inserted. When set to 1, enables wakeup event via Card Insertion assertion in the SRS12 register.
24	RW	0x0	WCINT; Wakeup Event Enable On Card Interrupt. When set to 1, enables wakeup event via Card Interrupt assertion in the SRS12 register.
23:20	RW	0x0	Reserved..
19	RW	0x0	IBG; Interrupt At Block Gap. When set to 1, enables interrupt detection at the block gap for a multiple block transfer. This bit is valid only in SD4 mode. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0.
18	RW	0x0	RWC; Read Wait Control. When set to 1, enables read wait control. The read wait function is optional for SDIO cards. If the card does not support read wait, this bit would never be set to 1; otherwise, DAT line conflict may occur.
17	RW	0x0	CR; Continue Request.

			<p>When set to 1, restarts the transfer previously stopped using the Stop At Block Gap.</p> <p>The software will clear SRS10.SBGR (Stop At Block Gap) bit before setting the Continue Request. When SRS10.SBGR=1, then all write operations to Continue Request are ignored. Clearing SRS10.SBGR can be done before or simultaneously with writing the Continue Request.</p> <p>Continue Request bit is cleared automatically by the SDIO-HOST when SRS9.DATLA (Dat Line Active) changes from 0 to 1, indicating the actual restart of the transfer.</p>
16	RW	0x0	<p>SBGR; Stop At Block Gap Request.</p> <p>When set to 1, orders the stop executing read and write transaction at the next block gap for non- DMA, SDMA and ADMA transfers. The software will maintain SBGR=1 until the current transfer is complete (typically by waiting for SRS12.TC - Transfer Complete bit). After Transfer Complete event, the software will clear SBGR back to 0.</p> <p>In the case of the read transfer, the SDIO-HOST stops after the next data block received from the card. This uses the ReadWait mechanism if it is enabled by SRS10.RWC, or stops the card clock (SDCx_CLK) if ReadWait is disabled.</p> <p>In the case of the write transfer, SDIO-HOST stops after the last block written to the data buffer. The SDIO-HOST sends all data already written to the data buffer before stopping the transfer.</p>
15:9	-	-	Reserved.
8	RW	0x0	<p>BP; SD Bus Power.</p> <p>When BP= 1, the SD device is powered. The state of this bit directly drives SDCx_PWR pin of the SDIO-HOST. Additionally, when BP=0, SDIO-HOST stops driving CMD, DAT, and SDCLK lines by setting: cmd_s#_en = 0 dat_s#_en = 0 and sdclk_s# = 0</p> <p>The SDIO-HOST clears BP to 0 automatically when card is removed from the slot (i.e. after high to low transition on SDCx_CD pin). This is to provide the "hot removal support".</p>

7	RW	0x0	<p>CDSS; Card Detect Signal Selection This bit selects active signal source for card detection bits and procedures.</p> <table> <thead> <tr> <th>Value</th><th>Active Source</th></tr> </thead> <tbody> <tr> <td>1</td><td>CDTL (SRS10.6) bit (for test purpose)</td></tr> <tr> <td>0</td><td>SDCx_CD pin (for normal operation)</td></tr> </tbody> </table>	Value	Active Source	1	CDTL (SRS10.6) bit (for test purpose)	0	SDCx_CD pin (for normal operation)				
Value	Active Source												
1	CDTL (SRS10.6) bit (for test purpose)												
0	SDCx_CD pin (for normal operation)												
6	RW	0x0	<p>CDTL; Card Detect Test Level This bit can be used for test purposes. When CDSS bit is set to 1, the value of this bit is used instead of the input signal SDCx_CD.</p> <table> <thead> <tr> <th>Value</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>1</td><td>Card Inserted</td></tr> <tr> <td>0</td><td>No Card</td></tr> </tbody> </table> <p>Please note that the active state of this bit is the opposite of the active state of the input signal.</p>	Value	Meaning	1	Card Inserted	0	No Card				
Value	Meaning												
1	Card Inserted												
0	No Card												
5	-	-	Reserved.										
4:3	RW	0x0	<p>DMASEL; DMA Select. Selects the DMA operating mode. Use of selected DMA is determined by DMA Enable bit in SRS3 register.</p> <table> <thead> <tr> <th>DMASEL</th><th>Select DMA Mode</th></tr> </thead> <tbody> <tr> <td>00b</td><td>SDMA</td></tr> <tr> <td>01b</td><td>ADMA</td></tr> <tr> <td>01b</td><td>ADMA2 (32-bit address)</td></tr> <tr> <td>11b</td><td>Reserved.</td></tr> </tbody> </table>	DMASEL	Select DMA Mode	00b	SDMA	01b	ADMA	01b	ADMA2 (32-bit address)	11b	Reserved.
DMASEL	Select DMA Mode												
00b	SDMA												
01b	ADMA												
01b	ADMA2 (32-bit address)												
11b	Reserved.												
2	RW	0x0	<p>HSE; High Speed Enable. Selects the operating mode.</p> <p>1 – High Speed mode selected. The SDIO-HOST outputs CMD and DAT lines on the rising edge of SDCLK clock. 0 – Default mode selected. The SDIO-HOST outputs CMD and DAT lines on the falling edge of SDCLK clock.</p> <p>The maximum SD clock frequency is defined as 0-25MHz in the default mode, and 0-50MHz in the High Speed mode.</p>										
1	RW	0x0	<p>DTW; Data Transfer Width. 1 – SD4 (4-bit) mode selected. 0 – SD1 (1-bit) mode selected.</p>										
0	RW	0x0	<p>LEDC; LED Control. State of this bit directly drives led pin of the SDIO- HOST in order to control the external LED diode.</p> <p>LEDC=1 will switch LED on. LEDC=0 will switch it off.</p> <p>The software will switch LED on to caution the user not to remove the card while the transfer is in progress.</p>										

SDC_SRS11

Address: Operational Base + offset (0x002C)

SRS11 Host Control Settings #1 Register

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26	RW	0x0	<p>SRDAT; Software Reset For DAT Line. When set to 1, resets the logic connected with the data path, including data buffers and the DMA logic.</p> <p>The following registers and bits are cleared: SFR8 register:</p> <ul style="list-style-type: none"> ● Buffer is cleared. <p>SFR9 register:</p> <ul style="list-style-type: none"> ● Buffer Read Enable ● Buffer Write Enable ● Read Transfer Active ● Write Transfer Active ● DAT Line Active ● Command Inhibit DAT <p>SFR10 register:</p> <ul style="list-style-type: none"> ● Continue Request ● Stop At Block Gap Request <p>SFR12 register:</p> <ul style="list-style-type: none"> ● Buffer Read Ready ● Buffer Write Ready ● DMA Interrupt ● Block Gap Event ● Transfer Complete <p>After completing the reset operation, SRDAT bit is automatically cleared. It takes some time to complete the reset operation, so the software will wait until SRDAT=0, and continue the other operations only when SRDAT=0.</p>
25	RW	0x0	<p>SRCMD; Software Reset For CMD Line. When set to 1, resets the logic connected with the command/response generation and checking.</p> <p>The following registers and bits are cleared: SRS9 register:</p> <ul style="list-style-type: none"> ● Command Inhibit CMD <p>SRS12 register:</p> <ul style="list-style-type: none"> ● Command Complete <p>After completing the reset operation, SRCMD bit is automatically cleared. It takes some time to complete the reset operation, so the software will wait until SRCMD=0, and continue the other operations only when SRCMD=0.</p>
24	RW	0x0	SRFA; Software Reset For All.

			<p>When set to 1, the entire SDIO-HOST is reset.</p> <p>After completing the reset operation, SRFA bit is automatically cleared. It takes some time to complete the reset operation, so the software will wait until SRFA=0, and continue the other operations only when SRFA=0.</p> <p>Additionally, after performing Software reset For All, the software will reset and reinitialize all cards inserted to the host.</p>																				
23:20	-	-	Reserved.																				
19:16	RW	0x0	<p>DTCV; Data Timeout Counter Value.</p> <p>This value determines the interval by which DAT line timeouts are detected.</p> <table> <thead> <tr> <th>DTCV</th> <th>Timeout Interval</th> </tr> </thead> <tbody> <tr> <td>1111b</td> <td>Reserved.</td> </tr> <tr> <td>1110b</td> <td>tsdmclk x 2²⁷</td> </tr> <tr> <td>1101b</td> <td>tsdmclk x 2²⁶</td> </tr> <tr> <td>.....</td> <td>.....</td> </tr> <tr> <td>0001b</td> <td>tsdmclk x 2¹⁴</td> </tr> <tr> <td>0000b</td> <td>tsdmclk x 2¹³</td> </tr> </tbody> </table> <p>This interval can be computed as below: Where tsdmclk is the SDC clock period.</p> <p>Refer to the Data Timeout Error in the SFR12 register for information on factors which generate data timeouts. When setting this register, prevent inadvertent timeout events by clearing the Data Timeout Error Status Enable (in the SFR12 register).</p>	DTCV	Timeout Interval	1111b	Reserved.	1110b	tsdmclk x 2 ²⁷	1101b	tsdmclk x 2 ²⁶	0001b	tsdmclk x 2 ¹⁴	0000b	tsdmclk x 2 ¹³						
DTCV	Timeout Interval																						
1111b	Reserved.																						
1110b	tsdmclk x 2 ²⁷																						
1101b	tsdmclk x 2 ²⁶																						
.....																						
0001b	tsdmclk x 2 ¹⁴																						
0000b	tsdmclk x 2 ¹³																						
15:8	RW	0x0	<p>SDCLKFS; SDCLK Frequency Select.</p> <p>This register is used to select the SDCLK frequency, i.e. frequency of the card clock (SDCx_CLK pin). It contains the divider for the base SDC clock.</p> <p>Only the following settings are allowed:</p> <table> <thead> <tr> <th>SDCLKFS</th> <th>SDCLK Frequency</th> </tr> </thead> <tbody> <tr> <td>80h</td> <td>fsdmclk / 256</td> </tr> <tr> <td>40h</td> <td>fsdmclk / 128</td> </tr> <tr> <td>20h</td> <td>fsdmclk / 64</td> </tr> <tr> <td>10h</td> <td>fsdmclk / 32</td> </tr> <tr> <td>08h</td> <td>fsdmclk / 16</td> </tr> <tr> <td>04h</td> <td>fsdmclk / 8</td> </tr> <tr> <td>02h</td> <td>fsdmclk / 4</td> </tr> <tr> <td>01h</td> <td>fsdmclk / 2</td> </tr> <tr> <td>00h</td> <td>fsdmclk / 1</td> </tr> </tbody> </table>	SDCLKFS	SDCLK Frequency	80h	fsdmclk / 256	40h	fsdmclk / 128	20h	fsdmclk / 64	10h	fsdmclk / 32	08h	fsdmclk / 16	04h	fsdmclk / 8	02h	fsdmclk / 4	01h	fsdmclk / 2	00h	fsdmclk / 1
SDCLKFS	SDCLK Frequency																						
80h	fsdmclk / 256																						
40h	fsdmclk / 128																						
20h	fsdmclk / 64																						
10h	fsdmclk / 32																						
08h	fsdmclk / 16																						
04h	fsdmclk / 8																						
02h	fsdmclk / 4																						
01h	fsdmclk / 2																						
00h	fsdmclk / 1																						

			Where fsdmclk is the SDC clock frequency. The value of this register can be changed only when SRS11.SDCE (SD Clock Enable) = 0.
7:3	-	-	Reserved.
2	RW	0x0	SDCE; SD Clock Enable. When set to 1, SDCx_CLK clock is enabled. When cleared to 0, SDCx_CLK clock is stopped. The SDIO-HOST clears SDCE automatically when card is removed from the slot (i.e. after the high to low transition on SDCx_CD pin). Also, the SDCx_CLK clock will be stopped by the software when changing the clock divider (i.e. SDCE bit will be cleared before writing SRS11.SDCLKFS).
1	R	0x0	ICS; Internal Clock Stable. (Read clear) When read as 1, indicates that the clock on sdmclk pin of the SDIO-HOST is stable after setting SRS11.ICE to 1. When read as 0, indicates that the clock is not stable yet (for example, the external PLL that generates the clock is not yet locked). The value of ICS is equal to the actual signal level on ics pin of the SDIO-HOST. The user would connect ics to the external PLL if required. Otherwise, ics would be connected directly to the ice output of the SDIO-HOST.
0	RW	0x0	ICE; Internal Clock Enable. State of this bit controls ice pin of the SDIO-HOST in order to control the external clock generator (for example, PLL). The ICE bits of every slot are logically OR-ed together and then drive the ice pin of SDIO-HOST. This means that the ice pin = 0 only when ICE bits = 0 for every slot implemented inside SDIO-HOST. The ice pin = 1 if at least one of the ICE bits is set to 1. When set to 0, the clock on sdmclk pin can be stopped externally. If the sdmclk is stopped, then SDIO-HOST goes to a very low power state. SDIO-HOST registers are still able to be read and written, even if the clock on sdmclk is stopped. Setting of the ICE bit does not affect card detection. This means that the card detection works even if the clock on sdmclk is stopped.

SDC_SRS12

Address: Operational Base + offset (0x0030)

SRS12 Interrupt Status Register

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.

25	RW	0x0	EADMA; ADMA Error Generated when the error occurs during ADMA read or write transfer. To resolve the cause of the error, the state of the ADMA engine at error occurrence is saved in ADMA Error Status register, and the address of the descriptor processed at error occurrence is provided in ADMA System Address register.
24	RW	0x0	EAC12; Auto CMD12 Error. Generated when the error occurs during Auto-CMD12 command transmission. It indicates one of the following situations: 1. Detecting that one of the bits in SRS15 register has changed from 0 to 1. 2. Auto CMD12 is not executed due to the previous command error.
23	RW	0x0	ECL; Current Limit Error. Generated when SDIO-HOST is not supplying power to SD card due to some failure. The value is equal to the actual signal level on cle pin of the SDIO-HOST.
22	RW	0x0	EDEB; Data End Bit Error. When set to 1, indicates detecting 0 at the end bit position of read data which uses the DAT line, or at the end bit position of the Write CRC Status.
21	RW	0x0	EDCRC; Data CRC Error. When set to 1, indicates 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".
20	RW	0x0	EDT; Data Timeout Error. When set to 1, indicates detecting one of the following timeout conditions: 1. Busy timeout for the response with busy. 2. Busy timeout after Write CRC status. 3. Write CRC Status timeout. 4. Read data timeout.
19	RW	0x0	ECI; Command Index Error. When set to 1, indicates that Index error occurs in the command response.
18	RW	0x0	ECEB; Command End Bit Error. When set to 1, indicates detecting that the end bit of a command response is 0.
17	RW	0x0	ECCRC; Command CRC Error When set to 1, indicates that command CRC error has occurred.

16	RW	0x0	ECT; Command Timeout Error When set to 1, indicates that no response was returned within 64 SDCLK cycles from the end bit of the command.
15	R	0x0	EINT; Error Interrupt. If any of the bits in range SRS12(31..16) is set, this bit is also set. The software can check for an error by reading this single bit first.
14:9	-	-	Reserved.
8	R	0x0	CINT; Card Interrupt. (Read clear) Indicates the card interrupt. CINT is not sampled by the card clock, so the interrupt can be detected even with SD clock stopped (SRS11.SDCE = 0). Also, CINT is not cleared by writing 1. Instead, the software will clear the source of an interrupt inside the card. After detecting the Card Interrupt, the software will stop further interrupt detection by clearing SRS13.CINTE to 0. Then, the software will clear the interrupt source inside the card by using the appropriate commands. For the details, please refer to the given SDIO Card Specification. After clearing the interrupt source, the card will stop to drive the interrupt signal to the SDIO-HOST. Finally, when the interrupt service routine is finished, the interrupt detection can be enabled by setting SRS13.CINT back to 1.
7	RW	0x0	CR; Card Removal. Generated when the SRS9.CI bit changes from 1 to 0, indicating card removal. When read as 1, indicates that the card was removed from the slot. When read as 0, indicates that the card state is stable (still inserted or removed) or that the debouncing is in progress.
6	RW	0x0	CIN; Card Insertion. Generated when the SRS9.CI bit changes from 0 to 1, indicating card insertion. When read as 1, indicates that the card was inserted to the slot. When read as 0, indicates that the card state is stable (still inserted or removed) or that the debouncing is in progress.
5	RW	0x0	BRR; Buffer Read Ready. Generated when the SRS9.BRE changes from 0 to 1, indicating that the data buffer can be read by the software.
4	RW	0x0	BWR; Buffer Write Ready. Generated when the SRS9.BWE changes from 0 to 1,

			indicating that the data buffer can be written by the software.
3	RW	0x0	DMAINT; DMA Interrupt. In SDMA mode, DMA interrupt is generated when the Host Controller detects the Host SDMA Buffer boundary. In ADMA mode, DMA interrupt is generated when the INT flag is set in a currently serviced ADMA descriptor. Note: Other DMA interrupt factors could be added in future releases of the SDIO-HOST.
2	RW	0x0	BGE; Block Gap Event. Generated when the read/write transaction is stopped at a block gap as the result of setting SRS10.SBGR to 1.
1	RW	0x0	TC; Transfer Complete. Generated when the transfer which uses the DAT line is complete. Transfers which use the DAT line include the read/write transfers and commands with a busy response. In the case of the read transfer, TC indicates that the entire data was transferred from the card to the host system (i.e. the SDIO-HOST FIFO is empty after reading the last data block). In the case of the write transfer, TC indicates that the entire data was transferred from the SDIO-HOST to the card (i.e. the SDIO-HOST FIFO is empty after writing the last data block), and the card accepted the data (busy signal released after the last block). In the case of the command with a busy response, TC indicates that the busy signal is released after the response.
0	RW	0x0	CC; Command Complete. Generated when the end bit of the response is received, except the response for Auto-CMD12 command. Auto-CMD12 command does not generate CC.

SDC_SRS13

Address: Operational Base + offset (0x0034)

SRS13 Interrupt Status Enable Register

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25	RW	0x0	EADMA SE; ADMA Error status enable. 1 – enabled 0 – masked
24	RW	0x0	EAC12 SE; Auto CMD12 Error status enable.

			1 – enabled 0 – masked
23	RW	0x0	ECL SE; Current Limit Error status enable. 1 – enabled 0 – masked
22	RW	0x0	EDEB SE; Data End Bit Error status enable. 1 – enabled 0 – masked
21	RW	0x0	EDCRC SE; Data CRC Error status enable. 1 – enabled 0 – masked
20	RW	0x0	EDT SE; Data Timeout Error status enable. 1 – enabled 0 – masked
19	RW	0x0	ECI SE; Command Index Error status enable. 1 – enabled 0 – masked
18	RW	0x0	ECEB SE; Command End Bit Error status enable. 1 – enabled 0 – masked
17	RW	0x0	ECCRC SE; Command CRC Error status enable. 1 – enabled 0 – masked
16	RW	0x0	ECT SE; Command Timeout Error status enable. 1 – enabled 0 – masked
15:9	-	-	Reserved.
8	RW	0x0	CINT SE; Card Interrupt status enable. 1 – enabled 0 – masked
7	RW	0x0	CR SE; Card Removal status enable. 1 – enabled 0 – masked
6	RW	0x0	CIN SE; Card Insertion status enable. 1 – enabled 0 – masked
5	RW	0x0	BRR SE; Buffer Read Ready status enable. 1 – enabled 0 – masked
4	RW	0x0	BWR SE; Buffer Write Ready status enable. 1 – enabled 0 – masked
3	RW	0x0	DMAINT SE; DMA Interrupt status enable. 1 – enabled 0 – masked

2	RW	0x0	BGE SE; Block Gap Event status enable. 1 – enabled 0 – masked
1	RW	0x0	TC SE; Transfer Complete status enable. 1 – enabled 0 – masked
0	RW	0x0	CC SE; Command Complete status enable. 1 – enabled 0 – masked

SDC_SRS14

Address: Operational Base + offset (0x0038)

SRS14 Interrupt Enable Register

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25	RW	0x0	EDMA IE; ADMA Error interrupt enable. 1 – enabled 0 – masked
24	RW	0x0	EAC12 IE; Auto CMD12 Error interrupt enable. 1 – enabled 0 – masked
23	RW	0x0	ECL IE; Current Limit Error interrupt enable. 1 – enabled 0 – masked
22	RW	0x0	EDEB IE; Data End Bit Error interrupt enable. 1 – enabled 0 – masked
21	RW	0x0	EDCRC IE; Data CRC Error interrupt enable. 1 – enabled 0 – masked
20	RW	0x0	EDT IE; Data Timeout Error interrupt enable. 1 – enabled 0 – masked
19	RW	0x0	ECI IE; Command Index Error interrupt enable. 1 – enabled 0 – masked
18	RW	0x0	ECEB IE; Command End Bit Error interrupt enable. 1 – enabled 0 – masked
17	RW	0x0	ECCRC IE; Command CRC Error interrupt enable. 1 – enabled 0 – masked
16	RW	0x0	ECT IE; Command Timeout Error interrupt enable. 1 – enabled 0 – masked

15:9	-	-	Reserved.
8	RW	0x0	CINT IE; Card Interrupt interrupt enable. 1 – enabled 0 – masked
7	RW	0x0	CR IE; Card Removal interrupt enable. 1 – enabled 0 – masked
6	RW	0x0	CIN IE; Card Insertion interrupt enable. 1 – enabled 0 – masked
5	RW	0x0	BRR IE; Buffer Read Ready interrupt enable. 1 – enabled 0 – masked
4	RW	0x0	BWR IE; Buffer Write Ready interrupt enable. 1 – enabled 0 – masked
3	RW	0x0	DMAINT IE; DMA Interrupt interrupt enable. 1 – enabled 0 – masked
2	RW	0x0	BGE IE; Block Gap Event interrupt enable. 1 – enabled 0 – masked
1	RW	0x0	TC IE; Transfer Complete interrupt enable. 1 – enabled 0 – masked
0	RW	0x0	CC IE; Command Complete interrupt enable. 1 – enabled 0 – masked

SDC_SRS15

Address: Operational Base + offset (0x003C)

SRS15 Interrupt Status Register

This register identifies the reason for the Auto CMD12 error. The software will check SRS15 after detecting Auto CMD12 error (SRS12.EAC12 = 1).

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7	R	0x0	CNIACE; Command Not Issued By Auto CMD12 Error. (Read clear) When read as 1, means that the command was not executed by the SDIO-HOST due to the previous Auto CMD12 error. When SDIO-HOST detects any error during Auto CMD12, then all further command generation attempts are blocked. The software reset sequence is needed for recovery.

6:5	-	-	Reserved.
4	R	0x0	ACIE; Auto CMD12 Index Error. (Read clear) When read as 1, means that Command Index error occurred in the Auto CMD12 response.
3	R	0x0	ACEBE; Auto CMD12 End Bit Error. (Read clear) When read as 1, indicates that the end bit of the Auto CMD12 response is 0.
2	R	0x0	ACCE; Auto CMD12 CRC Error. (Read clear) When read as 1, indicates a CRC error in the Auto CMD12 response.
1	R	0x0	ACTE; Auto CMD12 Timeout Error. (Read clear) When read as 1, indicates that no response is returned within 64 SDCx_CLK cycles from the end bit of the Auto CMD12. If this bit is set to 1, the other error status bits (D04-D02) are meaningless.
0	R	0x0	ACNE; Auto CMD12 Not Executed. (Read clear) When set to 1, means the SDIO-HOST cannot issue Auto CMD12 due to some error. If this bit is set to 1, other error status bits (D04-D01) are meaningless.

SDC_SRS16

Address: Operational Base + offset (0x0040)

SRS16 Capabilities Register

This register defines the characteristics and the specific features of the SDIO-HOST.

Bit	Attr	Reset Value	Description
31:29	-	-	Reserved.
28	R	0x0	64BS; 64-bit System Bus Support This bit is always set to 0 to indicate that core does not support 64-bit system bus.
27	-	-	Reserved.
26	R	0x1	VS18; Voltage Support 1.8V. 1 – 1.8V supported 0 – 1.8V not supported
25	R	0x1	VS30; Voltage Support 3.0V. 1 – 3.0V supported 0 – 3.0V not supported
24	R	0x1	VS33; Voltage Support 3.3V. 1 – 3.3V supported 0 – 3.3V not supported
23	R	0x1	SRS; Suspend/Resume Support. 1 – Suspend/Resume is enabled 0 – Suspend/Resume disabled
22	R	0x1	DMAS; DMA Support. 1 – DMA (SDMA mode) supported

			0 – DMA (SDMA mode) not supported This bit defines whether the DMA is supported in the given slot or not. There is a separate IMPLEMENT_DMA generic parameter which defines whether the DMA is physically implemented inside SDIO-HOST. If DMAS is set to 1 for any card slot, then the IMPLEMENT_DMA would also be set to 1.
21	R	0x1	HSS; High Speed Support. 1 – High Speed mode supported 0 – High Speed mode not supported
20	R	0x1	ADMA1S; ADMA1 Support. 1 – ADMA1 mode supported 0 – ADMA1 mode not supported
19	R	0x1	ADMA2S; ADMA2 Support. 1 – ADMA2 mode supported 0 – ADMA2 mode not supported
18	-	-	Reserved.
17:16	R	0x10	MBL; Max Block Length. This value indicates the maximum block size that can be transferred by the SDIO-HOST. Three sizes can be defined as indicated below: The physical buffer size is 2048 Bytes. Therefore, the Maximum Block Size defined by MBL will always be less or equal to the physical buffer size.
15:14	-	-	Reserved.
13:8	R	0x32	BCSDCLK; Base Clock Frequency For SD Clock. Defines the base (maximum) clock frequency for the SD Clock in 1MHz units. The base clock is the clock supplied on sdmclk pin of the SDIO-HOST. The maximum clock frequency supported is between 10MHz to 63MHz.
7	R	0x1	TCU; Timeout Clock Unit. Defines the frequency unit for the SRS16.TCF. 0 – kHz 1 – MHz
6	-	-	Reserved.

5:0	R	0x32	<p>TCF; Timeout Clock Frequency. Defines the base clock frequency used to detect Data Timeout Error. The SRS16.TCU bit determines the unit used.</p> <table> <thead> <tr> <th>TCF Value</th><th>Timeout Clock Frequency</th></tr> </thead> <tbody> <tr> <td>TCU = 0</td><td>TCU = 1</td></tr> <tr> <td>11111b</td><td>63KHz</td></tr> <tr> <td>11110b</td><td>62KHz</td></tr> <tr> <td>.....</td><td>.....</td></tr> <tr> <td>000001b</td><td>1KHz</td></tr> <tr> <td>000000b</td><td>Reserved.</td></tr> <tr> <td></td><td>1MHz</td></tr> <tr> <td></td><td>Reserved.</td></tr> </tbody> </table>	TCF Value	Timeout Clock Frequency	TCU = 0	TCU = 1	11111b	63KHz	11110b	62KHz	000001b	1KHz	000000b	Reserved.		1MHz		Reserved.
TCF Value	Timeout Clock Frequency																				
TCU = 0	TCU = 1																				
11111b	63KHz																				
11110b	62KHz																				
.....																				
000001b	1KHz																				
000000b	Reserved.																				
	1MHz																				
	Reserved.																				

SDC_SRS18

Address: Operational Base + offset (0x0048)

SRS18 Maximum Current Capabilities Register

This register defines the maximum current capability for each voltage range.

Bit	Attr	Reset Value	Description														
31:24	-	-	Reserved.														
23:16	R	0x1F	<p>MC18; Maximum Current for 1.8V</p> <table> <thead> <tr> <th>MC18</th> <th>Maximum Current</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved.</td> </tr> <tr> <td>1</td> <td>4 mA</td> </tr> <tr> <td>2</td> <td>8 mA</td> </tr> <tr> <td>3</td> <td>12 mA</td> </tr> <tr> <td>.....</td> <td>.....</td> </tr> <tr> <td>255</td> <td>1020 mA</td> </tr> </tbody> </table>	MC18	Maximum Current	0	Reserved.	1	4 mA	2	8 mA	3	12 mA	255	1020 mA
MC18	Maximum Current																
0	Reserved.																
1	4 mA																
2	8 mA																
3	12 mA																
.....																
255	1020 mA																
15:8	R	0xF	<p>MC30; Maximum Current for 3.0V</p> <table> <thead> <tr> <th>MC30</th> <th>Maximum Current</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved.</td> </tr> <tr> <td>1</td> <td>4 mA</td> </tr> <tr> <td>2</td> <td>8 mA</td> </tr> <tr> <td>3</td> <td>12 mA</td> </tr> <tr> <td>.....</td> <td>.....</td> </tr> <tr> <td>255</td> <td>1020 mA</td> </tr> </tbody> </table>	MC30	Maximum Current	0	Reserved.	1	4 mA	2	8 mA	3	12 mA	255	1020 mA
MC30	Maximum Current																
0	Reserved.																
1	4 mA																
2	8 mA																
3	12 mA																
.....																
255	1020 mA																

7:0	R	0x8	MC33; Maximum Current for 3.3V MC33 0 1 2 3 255	Maximum Current Reserved. 4 mA 8 mA 12 mA 1020 mA
-----	---	-----	--	---

SDC_SRS20

Address: Operational Base + offset (0x0050)
SRS20 Event Trigger Register

This register is implemented to allow user to force the host to report erroneous status bits. Writing the bit with 1 will be reflected in Interrupt status register -SRS12 (if corresponding Interrupt Status Enable bit is set in SRS13 register) or in AutoCMD12 Error status register (SRS15). Reading this bit returns 0.

Bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25	W	-	EADMA FE; Force ADMA Error Event.
24	W	-	EAC12 FE; Force Auto CMD12 Error Event.
23	W	-	ECL FE; Force Current Limit Error Event.
22	W	-	EDEB FE; Force Data End Bit Error Event.
21	W	-	EDCRC FE; Force Data CRC Error Event.
20	W	-	EDT FE; Force Data Timeout Error Event.
19	W	-	ECI FE; Force Command Index Error Event.
18	W	-	ECEB FE; Force Command End Bit Error Event.
17	W	-	ECCRC FE; Force Command CRC Error Event.
16	W	-	ECT FE; Force Command Timeout Error Event.
15:8	-	-	Reserved.
7	W	-	CNIACE FE; Force Command Not Issued By Auto CMD12 Error Event.
6:5	-	-	Reserved.
4	W	-	ACIE FE; Force Auto CMD12 Index Error Event.
3	W	-	ACEBE FE; Force Auto CMD12 End Bit Error Event.
2	W	-	ACCE FE; Force Auto CMD12 CRC Error Event.
1	W	-	ACTE FE; Force Auto CMD12 Timeout Error Event.
0	W	-	ACNE FE; Force Auto CMD12 Not Executed Event.

SDC_SRS21

Address: Operational Base + offset (0x0054)
SRS21 Event Trigger Register

When ADMA Error interrupt occurs, the state of this register reflects the state of the ADMA engine saved when the error occurred.

Bit	Attr	Reset Value	Description										
31:3	-	-	Reserved.										
2	RW	0x0	<p>ADMAL; ADMA Length Mismatch Error. This bit is set when:</p> <ol style="list-style-type: none"> 1. Total data length specified in ADMA descriptors is different from that specified by the Block Count and Block Length fields (if Block Count Enable is set). 2. Total data length cannot be divided by the block length (if Block Count Enable is not set). 										
1:0	RW	0x0	<p>ADMAES; ADMA Error State. The value of this field reflects the state of the ADMA state machine. The possible values are:</p> <table> <thead> <tr> <th>ADMAES</th> <th>State</th> </tr> </thead> <tbody> <tr> <td>00b</td> <td>ST STOP – ADMA stopped</td> </tr> <tr> <td>01b</td> <td>ST FDS – Fetching descriptor</td> </tr> <tr> <td>10b</td> <td>Not used</td> </tr> <tr> <td>11b</td> <td>ST TRF – Transfer date</td> </tr> </tbody> </table>	ADMAES	State	00b	ST STOP – ADMA stopped	01b	ST FDS – Fetching descriptor	10b	Not used	11b	ST TRF – Transfer date
ADMAES	State												
00b	ST STOP – ADMA stopped												
01b	ST FDS – Fetching descriptor												
10b	Not used												
11b	ST TRF – Transfer date												

SDC_SRS22

Address: Operational Base + offset (0x0058)

SRS22 ADMA System Address Register

The ADMA System Address register contains the physical descriptor address used in ADMA transfers.

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:0	RW	0x0	<p>ADMA SA; ADMA System Address The registers holds the physical address of the currently processed ADMA descriptor. The Host Driver should will set this register with the descriptors table base address before it starts the ADMA transfers. The Host Driver should not write this register while the data transfer is active. While the ADMA engine is processing the descriptors list the ADMASA value is always incremented to point the next descriptor to be fetched.</p> <p>If the ADMA Error occurs, the register holds the descriptor address depending on the ADMA Error State (SRS21.ADMAES) register value, as listed in the table below:</p> <table> <thead> <tr> <th>ADMAES</th><th>State</th></tr> </thead> <tbody> <tr> <td>00b</td><td>Points next of the error descriptor</td></tr> <tr> <td>01b</td><td>Points of error descriptor</td></tr> <tr> <td>10b</td><td>Not used</td></tr> <tr> <td>11b</td><td>Points next of the error descriptor</td></tr> </tbody> </table>	ADMAES	State	00b	Points next of the error descriptor	01b	Points of error descriptor	10b	Not used	11b	Points next of the error descriptor
ADMAES	State												
00b	Points next of the error descriptor												
01b	Points of error descriptor												
10b	Not used												
11b	Points next of the error descriptor												

SDC_CRS63

Address: Operational Base + offset (0x00FC)

CRS63 Interrupt Status Register

Bit	Attr	Reset Value	Description
31:2	R	0x0	<p>VVN; Vendor Version Number. This field is reserved for the vendor version number. The Host Driver would not use this status.</p>
23:16	R	0x1	<p>SVN; Specification Version Number. This field identifies the Host Controller Specification Version. SVN would be 01h for current version of the SDIO-HOST, indicating compatibility with SDIO Host Specification Version 2.0.</p>
15:2	-	-	Reserved.
1:0	R	0x0	<p>ISES; Interrupt Signal For Each Slot. These status bits indicate the logical OR of Interrupt Signal and Wakeup Signal for each slot. (Read clear)</p> <p>CRS63.0 – slot #0 interrupt status CRS63.1 – slot #1 interrupt status</p>

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Clock and Reset

SD Clock Control

The SD card clock is initially turned off after the reset. This clock is turned on only when a card is inserted into the slot. This specific sequence is needed to enable the SD clock: First, the software has to enable the sdmclk master clock (called also "internal clock"), if not already enabled. The sdmclk is enabled by setting SRS11.ICE bit (Internal Clock Enable), and then checking SRS11.ICS (Internal Clock Stable) until it reads 1. When the internal clock is stable, the SD clock can be turned on by setting SRS11.SDCE.

The SDIO-HOST automatically turns off the SD clock by clearing SRS11.SDCE after detecting high to low transition on SDCx_CD pin (Card Detect). This is because the SD clock would be turned off when there is no card inside the slot. The software can also turn the SD clock off manually when no transaction is in progress by clearing SRS11.SDCE.

A specific sequence is also needed when changing the SD clock frequency. Before changing the SD frequency, the software has to clear the SRS11.SDCE bit, if previously set. Then it is possible to change the frequency by writing an appropriate value to SRS11.SDCLKFS (SD Clock Frequency Select). After writing SRS11.SDCLKFS, the software can restart the SD clock by setting SRS11.CDCE to 1.

Hardware Reset

For a proper hardware reset sequence, the reset signal would be asserted for at least 8 clock cycles of both clk and sdmclk clocks (slower clock would be used when defining the actual reset sequence length). Both clocks would be running at the reset time.

Software Reset

Software reset can be performed at any time by setting one of the following bits: HSR01.SWR (Software Reset) – Resets all internal logic. This reset is common for all slots and works like the hardware reset. Refer to the HRS01 register description for details.

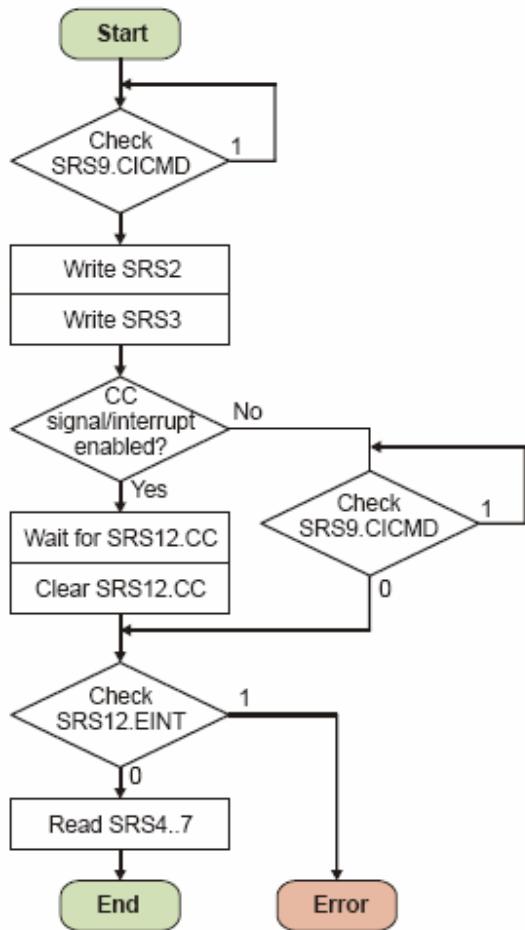
SRS11.SRFA (Software Reset For All) – Resets all internal logic for a given slot. Refer to the SRS11 register description for details.

SRS11.SRCMD (Software Reset For CMD) – Resets the part of the design responsible for command generation and response checking for a given slot. Refer to the SRS11 register description for details.

SRS11.SRDAT (Software Reset For DAT) – Resets the part of the design responsible for data transfers for a given slot. Refer to the SRS11 register description for the details.

SD Transaction Generation

Commands Which Do Not Use DAT Line



Commands in this category include all commands other than commands which transfer data through DAT line or commands with a BUSY response.

Before sending any command, the software has to verify that no other command/response cycle is currently in progress by checking SRS9.CICMD bit (Command Inhibit CMD).

When SRS9.CICMD = 0, then the CMD line is idle and the software is free to send another command. First, the software writes an appropriate command argument, 32-bit field in SRS2. Then it writes SRS3 with an appropriate command configuration. Writing SRS3.CI (Command Index) field directly triggers the command generation/response checking process.

When command generation/response checking is in progress, the SRS9.CICMD is set to 1. After finishing the entire command/response cycle on the SD interface, SRS9.CICMD is automatically cleared by the SDIO-HOST. High to low transition on SRS9.CICMD triggers the Command Complete interrupt (SRS12.CC) if it is enabled (SRS13.CC_SE & SRS14.CC_IE enable bits).

At this time, the command/response cycle is complete and the software can read the card response from SRS.4 – SRS.7. The software can also check for possible response error status bits (SRS12.ECI, SRS12.ECEB, SRS12.ECRC, SRS12.ECT).

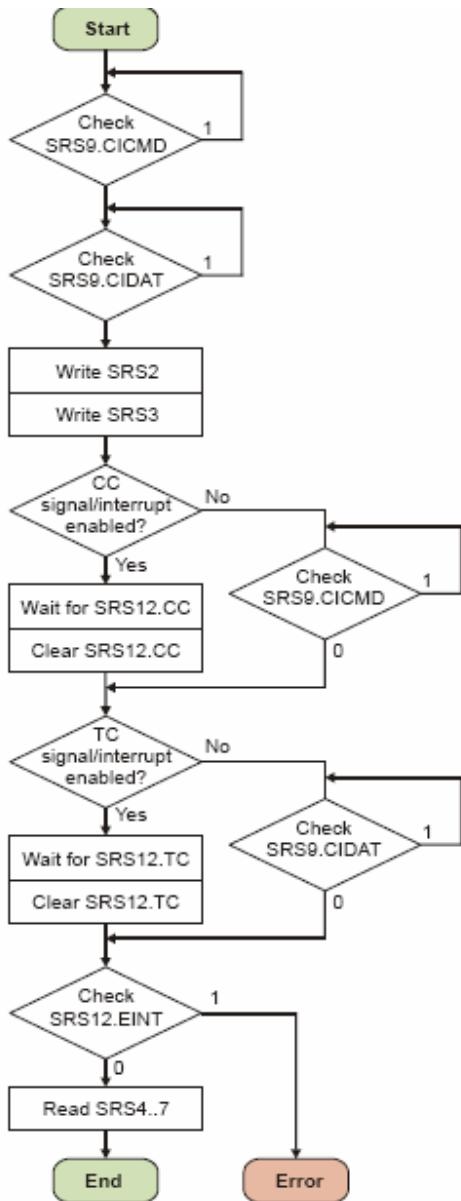
After finishing with the command, the software clears the Command Complete Interrupt (SRS12.CC) by writing 1 to this bit.

Commands with BUSY

Commands in this category use the DAT line for BUSY signaling. The procedure is very similar to the procedure of sending the commands which do not use DAT line. The difference is that, after checking SRS9.CICMD bit (Command Inhibit CMD), the software also checks SRS9.CIDAT (Command Inhibit DAT). The command with BUSY can only be sent if both SRS9.CICMD and SRS9.CIDAT are cleared. The only exception is the ABORT type command, that is, a command with SRS3.CT (Command Type) = 11b. The ABORT command can be sent even if SRS9.CIDAT = 0. It is possible, since the ABORT commands are used for breaking the other transaction that uses DAT line.

When command generation/response checking is in progress, the SRS9.CICMD is set to 1. When sending the command on CMD, SDIO-HOST also sets SRS9.CIDAT to 1. After finishing the command/response cycle on the CMD line, SRS9.CICMD is automatically cleared by the SDIO-HOST. High to low transition on SRS9.CICMD triggers the Command Complete interrupt (SRS12.CC) if it is enabled (SRS13.CC_SE & SRS14.CC_IE enable bits). The software will clear SRS12.CC manually by writing logical 1.

After the command/response cycle on CMD, the card can send the BUSY status using the DAT line. SRS9.CIDAT remains set until the BUSY is released. High to low transition on SRS9.CIDAT triggers the Transfer Complete interrupt (SRS12.TC) if it is enabled (SRS13.TC_SE & SRS14.TC_IE enable bits). The software will wait until the BUSY is released by looking at the SRS9.CIDAT or, alternatively, by waiting for SRS12.TC interrupt. After the Transfer Complete event, the SRS12.TC will be cleared by the software. At this time, the entire command/response cycle is complete.



Commands Which Transfer Data on the DAT Line without DMA

Commands in this category use the DAT line for a data transfer. Such commands include single- and multi-block read/write data transfers. The read transfer term indicates that the data is read from the card to the SDIO-HOST. The write transfer indicates that the data is written to the card by the SDIO-HOST.

The software first checks the Command Inhibit CMD (SRS9.CICMD) and Command Inhibit DAT (SRS9.CIDAT) before doing anything else, exactly as in the case of the commands with BUSY. The data transfer command can only be started when SRS9.CICMD = SRS9.CIDAT = 0.

The data transfer requires the appropriate configuration setup in SRS1 register: SRS1.BCCT (Block Count for Current Transfer), and SRS1.TBS (Transfer Block Size). Also SRS3.BCE is set for the transfer with a known number of data blocks, or cleared for

the infinite data transfer.

A few further steps are common for all command types. The software writes an appropriate command argument in SRS2. Then it writes SRS3 with the command configuration. SRS3.DPS (Data Present Select) is set to 1. SRS3.DTDS (Data Transfer Direction Select) is 1 for the read transfers, or 0 for the write transfers. SRS3.DMAE is cleared, since the DMA is not used.

When command generation/response checking is in progress, the SRS9.CICMD is set to 1. After finishing the entire command/response cycle on the SD interface, SRS9.CICMD is automatically cleared by the SDIO-HOST. High to low transition on SRS9.CICMD triggers the Command Complete interrupt (SRS12.CC) if it is enabled. The software will clear SRS12.CC manually by writing logical 1.

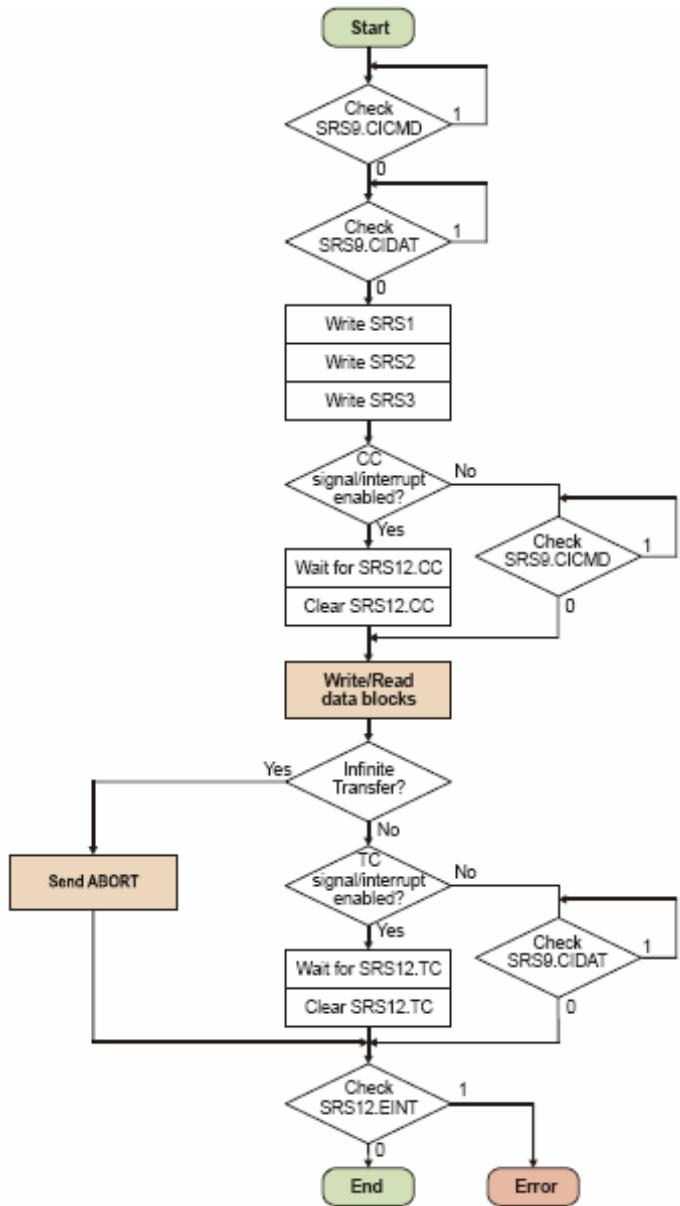
Then the software will process the data blocks.

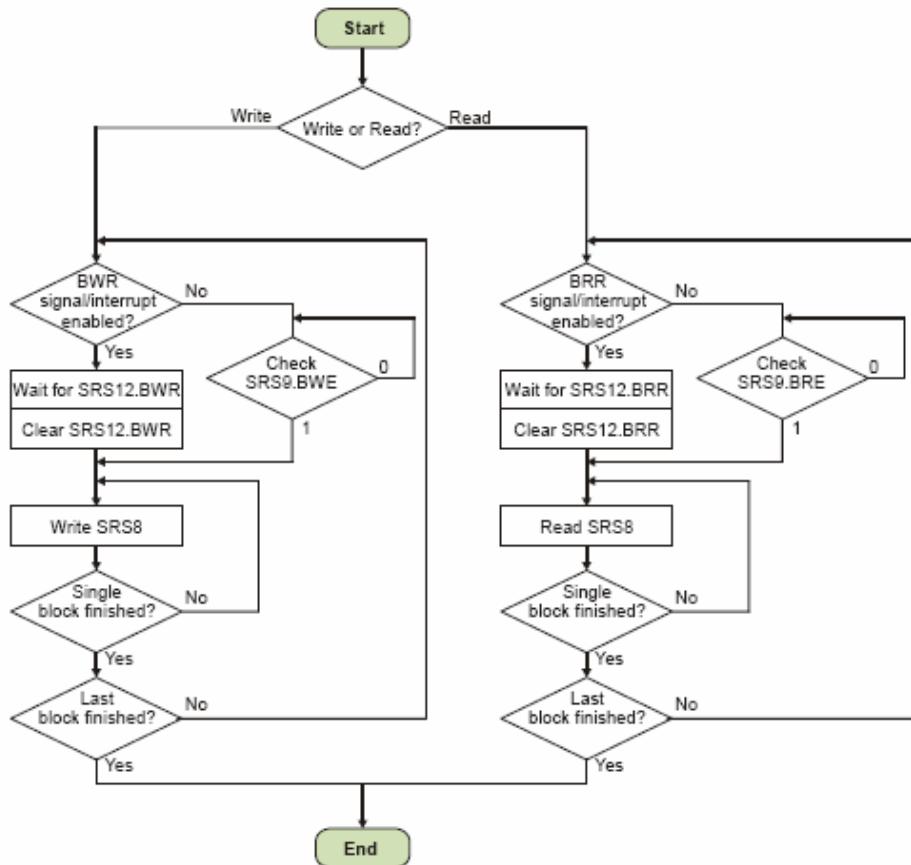
In case of the write transfer, the data would be written to SRS8 (Data Buffer) on a block by block basis. SRS9.BWE (Buffer Write Enable) can be used to check if there is a free FIFO buffer inside SDIO-HOST. SRS9.BWE = 1 means that there is enough space to write the entire block. Alternatively, the software can use SRS12.BWR (Buffer Write Ready) interrupt that is triggered whenever the FIFO buffer becomes ready for the transfer.

In case of the read transfer, the data would be read from SRS8 (Data Buffer) on a block by block basis. SRS9.BRE (Buffer Read Enable) indicates that the entire block can be read from the FIFO. Alternatively, the SRS12.BRR (Buffer Read Ready) is generated whenever the SRS9.BRE changes from 0 to 1, indicating the new data block ready in the FIFO.

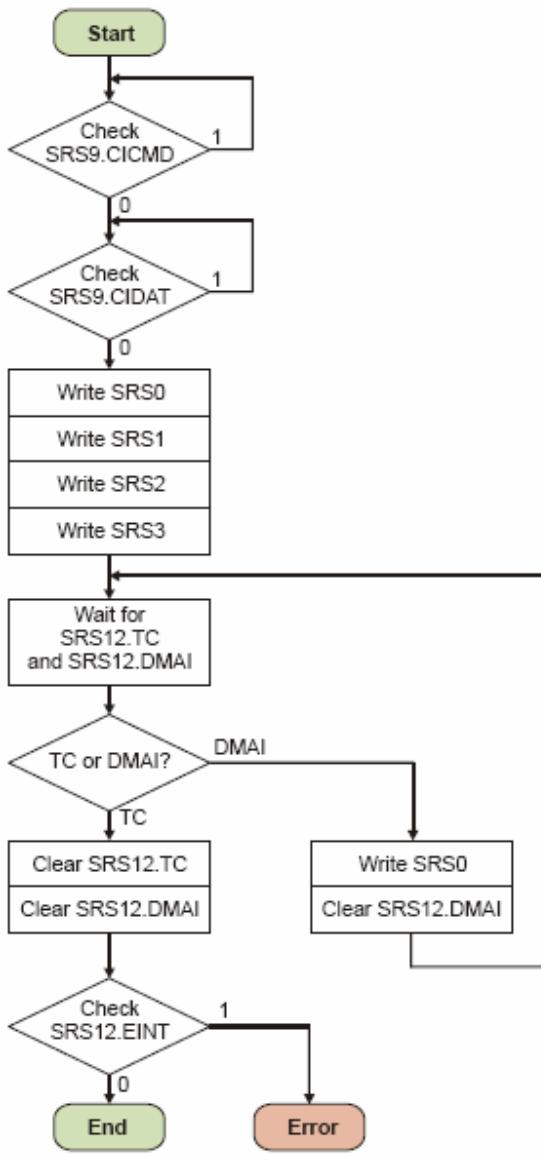
The software will process all data blocks as indicated by the SRS1.BCCT (Block Count for Current Transfer). During the entire data transfer, SRS9.CIDAT (Command Inhibit DAT) remains set. SRS9.CIDAT is cleared by the SDIO-HOST after the last block transfers to/from the FIFO. High to low transition on SRS9.CIDAT triggers the Transfer Complete interrupt (SRS12.TC) if it is enabled (SRS13.TC_SE & SRS14.TC_IE enable bits).

The software will wait until the busy is released by looking at the SRS9.CIDAT or, alternatively, by waiting for SRS12.TC interrupt. After the Transfer Complete event, the SRS12.TC will be cleared by the software. The only exception is for the infinite data transfers. The transfer is infinite when SRS3.BCE (Block Count Enable) = 0. In this case, the block count is not known to the SDIO-HOST, and the software has to break the transfer manually by performing the Abort procedure. This Abort procedure is described in a separate section.





Commands Which Transfer Data on the DAT Line Using DMA



This entire procedure is based on the previous procedure (for commands which do not use DMA). There are some differences in the setup:

SRS0.SA (System Address) is written with the memory address of the first data block. All data blocks are ready in the system memory (will contain valid data in case of write transfer, or will be reserved for the read transfer).

SRS1.HDMABB (Host DMA Address Buffer Boundary) is written with the appropriate value (this value depends on the system memory and bus architecture).

SRS3.DMAE (DMA Enable) bit is set to 1.

SRS3.BCE (Block Count Enable) is set to 1. This is because the infinite transfers are not possible when using DMA.

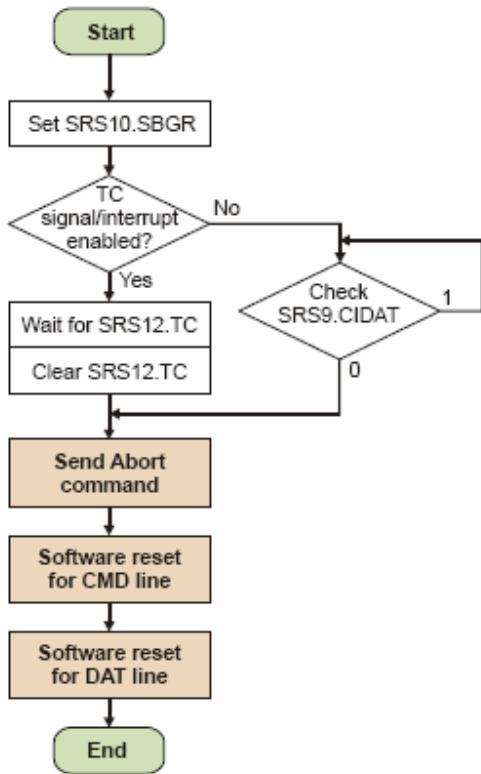
SRS13.DINT_SE (DMA Interrupt Signal Enable) is set to 1.

When DMA is enabled, the software does not have to process the data buffer manually. The SDIO-HOST will transfer the data between the system and the internal FIFO

automatically, block by block. After transferring the last data block, the SDIO-HOST clears SRS9.CIDAT (Command Inhibit DAT), and generates SRS12.TC (Transfer Complete Interrupt) if it is enabled.

During the data transaction, the SDIO-HOST also generates SRS12.DINT (DMA Interrupt) when crossing the boundary defined in SRS1.HDMABB (Host DMA Buffer Boundary). In such a case, the software would clear SRS1.DINT by writing 1, and restart the DMA operation by writing the SRS0.SA register. The SDIO-HOST restarts the DMA transfer from the current SRS0.SA value.

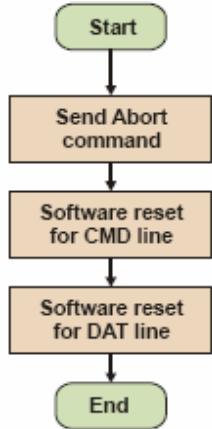
Synchronous Abort



The synchronous abort is the standard and recommended way of breaking the data transaction. Abort is used, for example, for stopping an infinite data transaction. The software sets SRS10.SBGR (Stop at Block Gap Request) to stop the transfer at the gap between two consecutive data blocks. When stopping, the SDIO-HOST clears SRS9.CIDAT (Command Inhibit DAT), and generates SRS12.TC (Transfer Complete Interrupt) if it is enabled. So the software simply waits for a Transfer Complete, and then clears the Transfer Complete status.

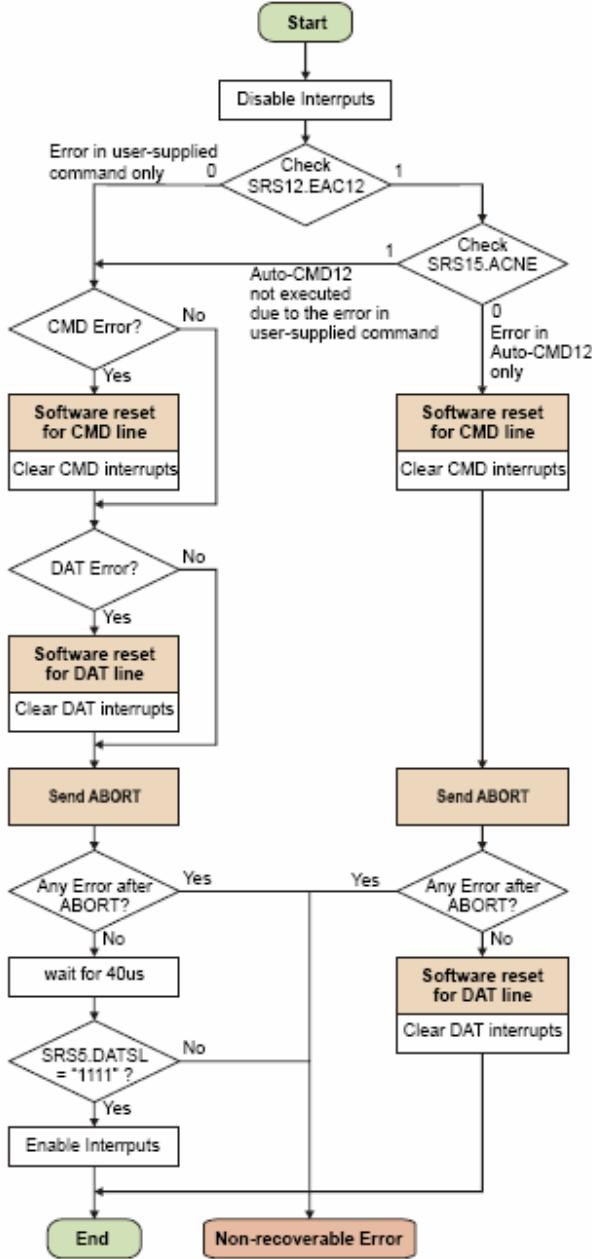
The next step is to send the ABORT type command to the card. The procedure of sending such command is described in the "Command with BUSY" section. The ABORT type command is indicated by SRS3.CT (Command Type) = 11b.

Finally, the software is reset for both CMD and DAT lines. The software performs the reset by setting SRS11.SRCMD and SRS11.SRDAT with 1. Synchronous Abort procedure is complete when both SRS11.SRCMD and SRS11.SRDAT are cleared by SDIO-HOST.

Asynchronous Abort

This entire procedure is based on the previous procedure (Synchronous Abort). The only difference is that the Stop at Block Gap Request is not used in this case. Asynchronous Abort can be issued at any time when SRS3.CICMD (Command Inhibit CMD) is cleared.

Error Recovery



The Error recovery procedure is executed by the driver when detecting any error during the previously described procedures. First, the software checks SRS12.EAC bit, which indicates if the error is related to the Auto-CMD12 command generation. There are two possible reasons for Auto-CMD12 error: either an error occurred in the Auto-CMD12 itself, or an error occurred in a previously sent command, and SDIO-HOST was not able to send Auto-CMD12 at all. These two cases can be distinguished by reading SRS15.ACNE (Auto-CMD12 Command Not Executed).

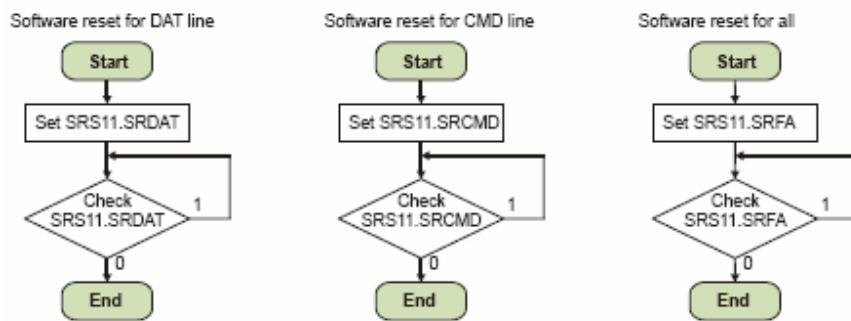
If an error is caused by the user-supplied command (i.e. not in Auto-CMD12), then the

software would perform software reset based on the SRS12 Error status. If at least one of SRS12.(19..16) bits is set indicating CMD line error, then the software would perform Software Reset for the CMD line. If at least one of SRS12.(23..20) bits is set indicating DAT line error, then the software would perform Software Reset for DAT line.

After the CMD/DAT line resets, the software sends Abort command. If any additional error occurs during Abort command, then the error is considered a non-recoverable. If the Abort command completes without any error, then the software would wait for about 40us and check the DAT line bits by reading SRS5.DATSL. DAT line bits should be all 1's, unless the error is non-recoverable.

In the case of Auto-CMD12 error, the procedure is very similar to the procedure described above. The difference is that the Software reset for DAT line is performed after sending the ABORT command.

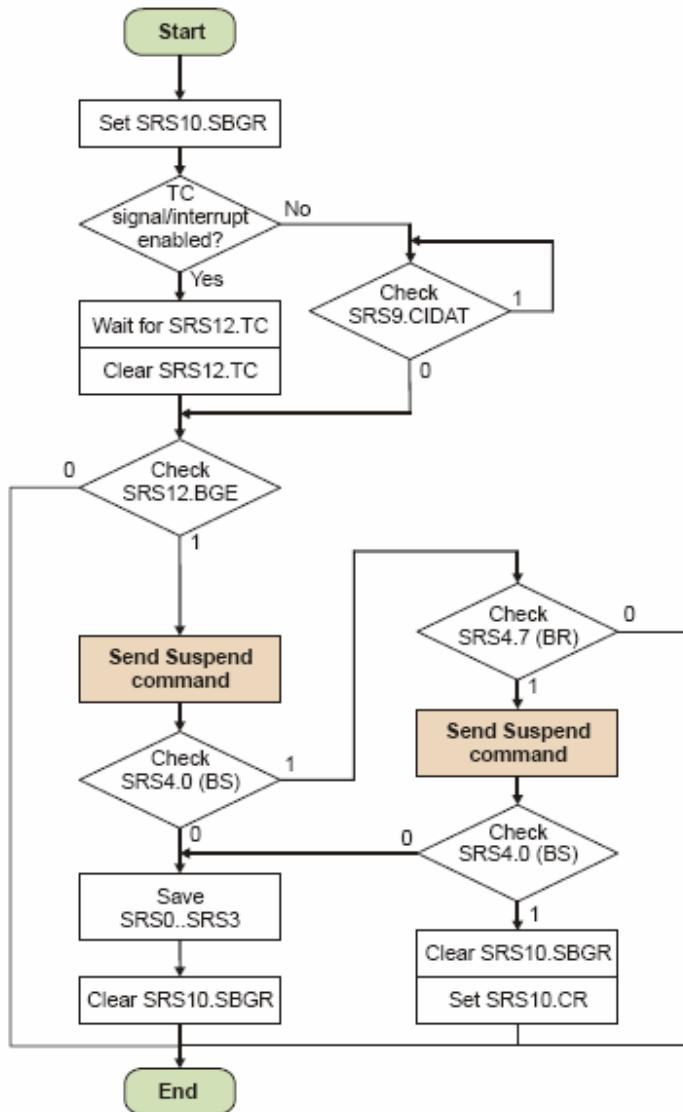
In the case of a non-recoverable error, the software driver would turn the power of the card off by clearing SRS10.BP bit, and then reinitialize both SDIO-HOST and the card.



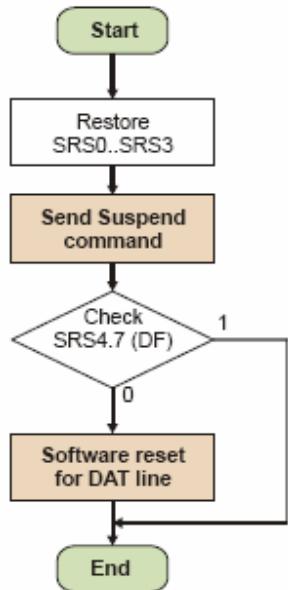
Suspend/Resume Mechanism

The suspend/resume mechanism is used for SDIO Cards with multiple functions or by the Combo Cards, i.e. SDIO cards with both IO and memory devices. The suspend procedure instructs the currently handled function to cut off from the bus. Then the software can access another (typically, higher priority) function. The suspended function can be restarted again by using the resume procedure.

Suspend Sequence



Resume Sequence



Multi-Slot Operation

The SDIO-HOST can implement up to 4 slots. Each slot has separate SRS register spaces and can be controlled independently. This means that the multi-slot operation is transparent for the software driver developer.

However, the user should note that all slots share the same data path, including FIFO, DMA, and command/response logic inside SDIO-HOST. This means that only a single command can be executed at a given time. If the software instructs SDIO-HOST to send commands (all kinds of commands, including commands which transfer data) for more than one slot at the same time, then the commands are sent to each slot in sequential order. All slots have the same priority, and round-robin arbitration is used.

DMA Operation

The SDIO-HOST controller supports 3 DMA modes:

SDMA – algorithm defined in SD Host Controller Specification Version 1.00

ADMA1 – algorithms defined in SD Host Controller Specification Version 2.00

● **ADMA2** – algorithms defined in SD Host Controller Specification Version 2.00; the core supports 32-bit addressing mode

The DMA mode for current transfer is selected via SRS10.DMASEL register and can be different for each consecutive data transfer. The Host Driver can change DMA mode when neither the Write Transfer Active (SRS9.WTA) nor the Read Transfer Active (SRS9.RTA) status bit is set.

The DMA transfer in each mode can be stopped by setting Stop at the Block Gap Request bit (SRS10.SBGR). The ADMA transfers can be restarted only by setting Continue Request bit (SRS10.CR), while SDMA transfers can be restarted by setting Continue Request bit or writing a new address to SDMA System Address register

(SRS0.SDMASA).

If an error occurs, the Host Driver will abort the DMA transfer in each mode by setting Software Reset for DAT Line (SRS11.SRDAT) and issuing CMD12 command, if a multiple block transfer is executing.

SDMA – Single-operation DMA Mode

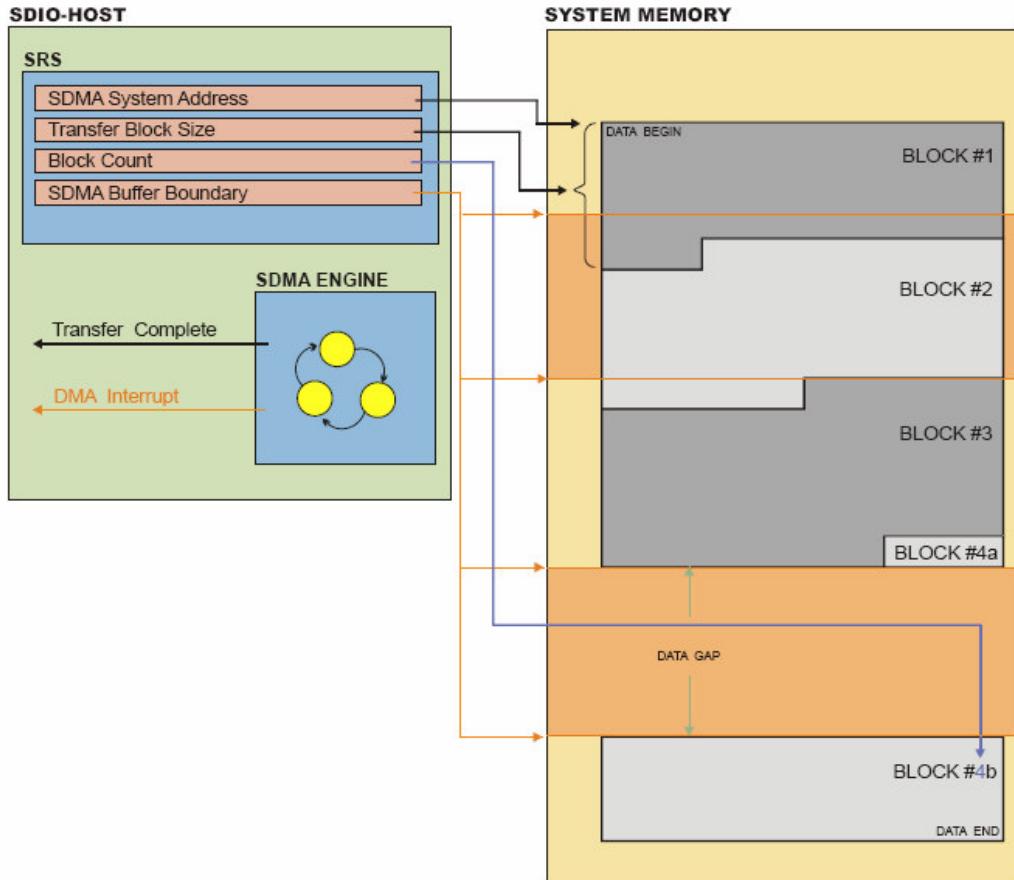
The single-operation DMA mode uses SDIO-HOST registers to describe the data transfer. The SDMA System Address (SRS0) register defines the base address of the data block. The length of the data transfer is defined by the Block Count (SRS1.BCCNT) and Transfer Block Size (SRS1.TBS) values. There is no limitation on the SDMA System Address value – the data block can start at any address.

The SDMA engine waits at every boundary specified in the Host SDMA Buffer Boundary (SRS1.HDMABB) register. The SDIO-HOST controller generates the DMA interrupt once the buffer boundary is reached, stops the current transfer and requests the Host Driver to update the SDMA System Address register, or order the transfer to continue.

When the SDMA engine stops at the buffer boundary, the SDMA System Address register points the next system address of the next data position to be transferred. The SDMA engine restarts the transfer when the uppermost byte of the SDMA System Address register is written or the SRS10.CR bit is set.

The SDMA engine neither uses nor affects the ADMA Error Status (SRS21) and ADMA System Address (SRS22) registers.

Below is the block diagram of SDMA



ADMA1 – Advanced DMA Mode Version 1 (4KByte boundary based DMA)

The Advanced DMA Mode Version 1 (ADMA1) uses the Descriptors List to describe data transfers. The SDIO-HOST registers only point to the base address of the Descriptors List; the base addresses and sizes of the data pages are defined inside the descriptors.

The ADMA1 mode was defined in the draft SD Controller Specification Version 2.00, but was replaced by the ADMA2 in the final release of the specification. However, to provide backward compatibility, the ADMA1 remained as the supplementary ADMA mechanism and is described in Appendix C of the final specification. Even though the SDIO-HOST supports the ADMA1 mode for data transferring, use of the ADMA2 mode (that has no 4KByte boundary limitations) is strongly recommended where possible.

1. ADMA1 Data Page

When in ADMA1 mode, the SDIO-HOST transfers data from data pages. A page is a block of valid data that is defined by a single ADMA1 Descriptor. Each ADMA1 descriptor can define only one data page; the starting address of the data page must be aligned to the 4KB boundary. The size of each data page is arbitrary – it depends on neither the previous nor the successive page size; it can also differ from the SD card

transfer block size (SRS1.TBS).

At the start of the ADMA engine, the default page size is set to 4KBytes. The size of the data page can be changed using the SET type descriptor.

2. ADMA1 Descriptors Table

The ADMA1 engine transfers are configured in a Descriptor List. The base address for this list is set in the ADMA System Address register (SRS22) whether it is read or write transfer. The ADMA1 Descriptor List consists of a number of 32-bit descriptors of different functions.

Each descriptor can:

Perform transfer of data page

Set the data page length

Link next descriptor address to an arbitrary memory location

Additionally, each descriptor has a set of flags, as follows:

VAL – denoting a valid ADMA1 descriptor

END – denoting last descriptor on the Descriptor List

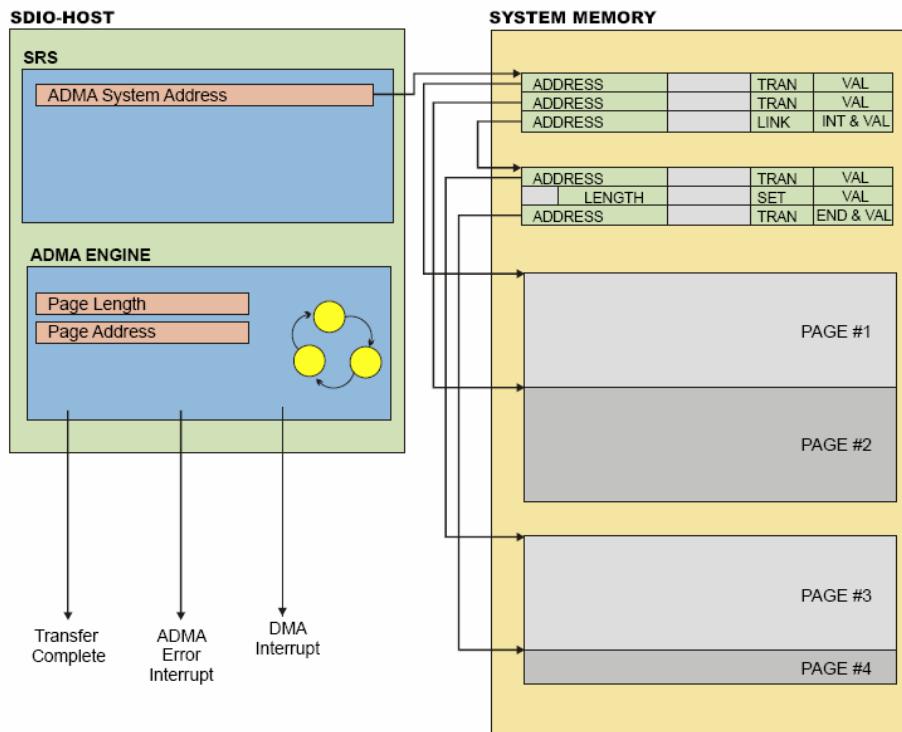
INT – denoting the ADMA interrupt requesting when the current transfer is completed

The table below defines the ADMA1 descriptor fields and their functions:

Bit	Symbol	Description
31:12	ADDRESS/ LENGTH	The field contains data page address, data page length, or next descriptor list address, depending on the descriptor type.
		When the descriptor is type TRAN, the field contains the 20 most significant bits of the page address – the page must be 4KB aligned in this mode (the less significant 12 bits of the address are always 0).
		When the descriptor is type SET, the field bits 27 to 12 contain the length (in bytes) of the subsequent pages. Please note that the default 4KB page size is set at every start of the ADMA1 engine.
11:6	-	When the descriptor type is LINK, the field contains the address for the next Descriptor List. The field contains the 20 most significant bits of the address – the next Descriptor List base address must be 4KB aligned in this mode. Reserved.

5:4	ACT	The field defines the type of descriptor.
		Value Type Command
	00	NOP No operation – go to next descriptor on the list.
	01	SET Set data page length and go to next descriptor on the list.
	10	TRAN Transfer data from the pointed page and go to next descriptor on the list.
	11	LINK Go to the next descriptor list
3	-	Reserved.
2	INT	When this bit is set, the ADMA interrupt is generated when the ADMA1 engine finishes processing the descriptor.
1	END	When this bit is set, it signals termination of the transfer and generates Transfer Complete Interrupt when this transfer is completed.
0	VAL	When this bit is set, it indicates the valid descriptor on a list. When this bit is cleared, the ADMA Error Interrupt is generated and the ADMA1 engine stops processing the Descriptor List. This prevents ADMA1 engine runaway due to improper descriptors.

The figure below shows a schematic of the ADMA1 process:



ADMA2 – Advanced DMA Mode Version 2

The Advanced DMA Mode Version 2 (ADMA2) uses the Descriptors List to describe data transfers. The SDIO-HOST registers only point to the base address of the Descriptors List; the base addresses and sizes of the data pages are defined inside the descriptors.

The ADMA2 mode was defined in the SD Controller Specification Version 2.00 and is the recommended ADMA mode. The SDIO-HOST supports a 32-bit address version of ADMA2 mode.

1. ADMA1 Data Page

When in ADMA2 mode, the SDIO-HOST transfers data from data pages. A page is a block of valid data that is defined by a single ADMA2 Descriptor. Each ADMA2 descriptor can define only one data page; the starting address of the data page must be aligned to the 4B boundary (the 2 least significant bits set to 0). The size of each data page is arbitrary – it depends on neither the previous nor the successive page size; it can also be different from the SD card transfer block size (SRS1.TBS).

2. ADMA1 Descriptors Table

The ADMA2 engine transfers are configured in a Descriptor List. The base address for this list is set in the ADMA System Address register (SRS22) whether it is a read or write transfer. The ADMA2 Descriptor List consists of a number of 64-bit descriptors of different functions.

Each descriptor can:

Perform transfer of data page of specified size

Link next descriptor address to an arbitrary memory location

Additionally, each descriptor has a set of flags, as follows:

VAL – denoting a valid ADMA1 descriptor

END – denoting last descriptor of the Descriptor List

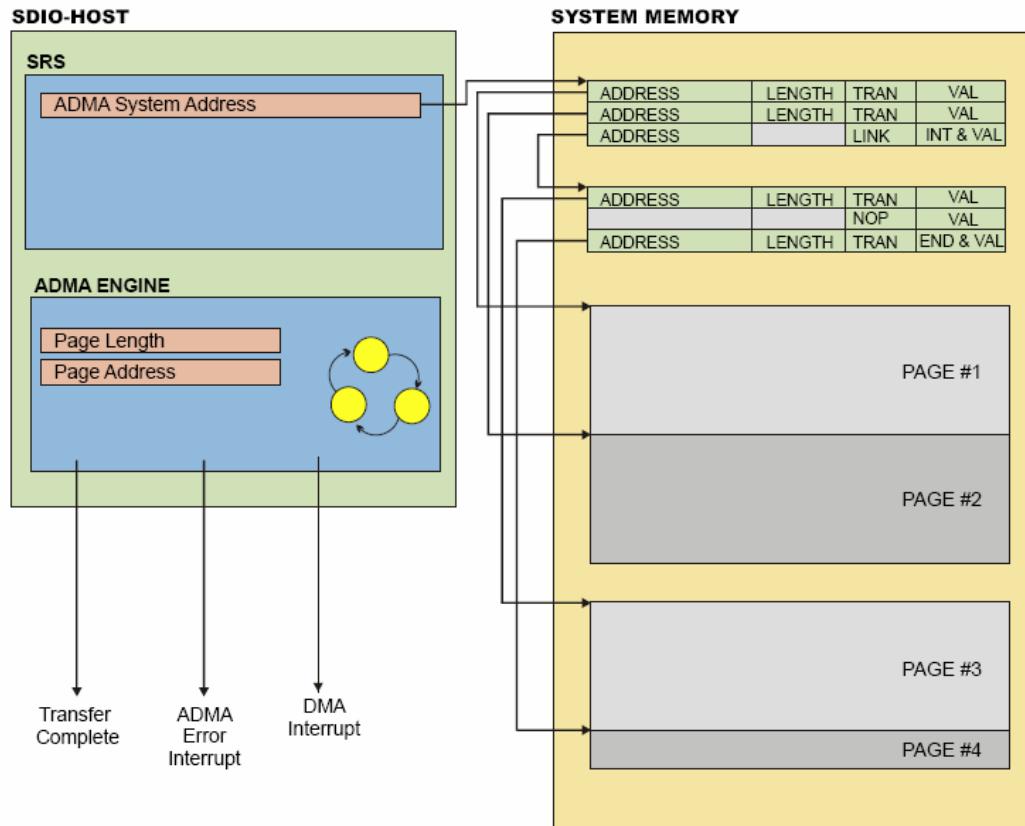
INT - denoting the ADMA interrupt requesting when the current transfer is complete

The table below defines the ADMA2 descriptor fields and their functions:

Bit	Symbol	Description
63:32	ADDRESS	The field contains data page address or next descriptor list address depending on the descriptor type. When the descriptor is type TRAN, the field contains the page address.
31:16	LENGTH	When the descriptor type is LINK, the field contains address for the next Descriptor List. The field contains data page length in bytes. 15:6 - Reserved.

5:4	ACT	The field defines the type of descriptor.		
		Value	Type	Command
		00	NOP	No operation – go to next descriptor on the list.
		01	-	Reserved.
		10	TRAN	Transfer data from the pointed page and go to next descriptor on the list.
		11	LINK	Go to the next descriptor list Reserved.
3	-			
ADMA1D.2	INT	When this bit is set, the ADMA interrupt is generated when the ADMA1 engine completes processing of the descriptor.		
ADMA1D.1	END	When this bit is set, it signals termination of the transfer and generates Transfer Complete Interrupt when this transfer is completed.		
ADMA1D.0	VAL	When this bit is set, it indicates the valid descriptor on a list. When this bit is cleared, the ADMA Error Interrupt is generated and the ADMA2 engine stops processing the Descriptor List. This prevents ADMA2 engine runaway due to improper descriptors.		

The figure below shows a schematic of the ADMA2 process:



Chapter 15 UART (16550)

Overview

The UART is an APB slave performing:

- Serial-to-parallel conversion on data received from a peripheral device
- Parallel-to-serial conversion on data transmitted to the peripheral device

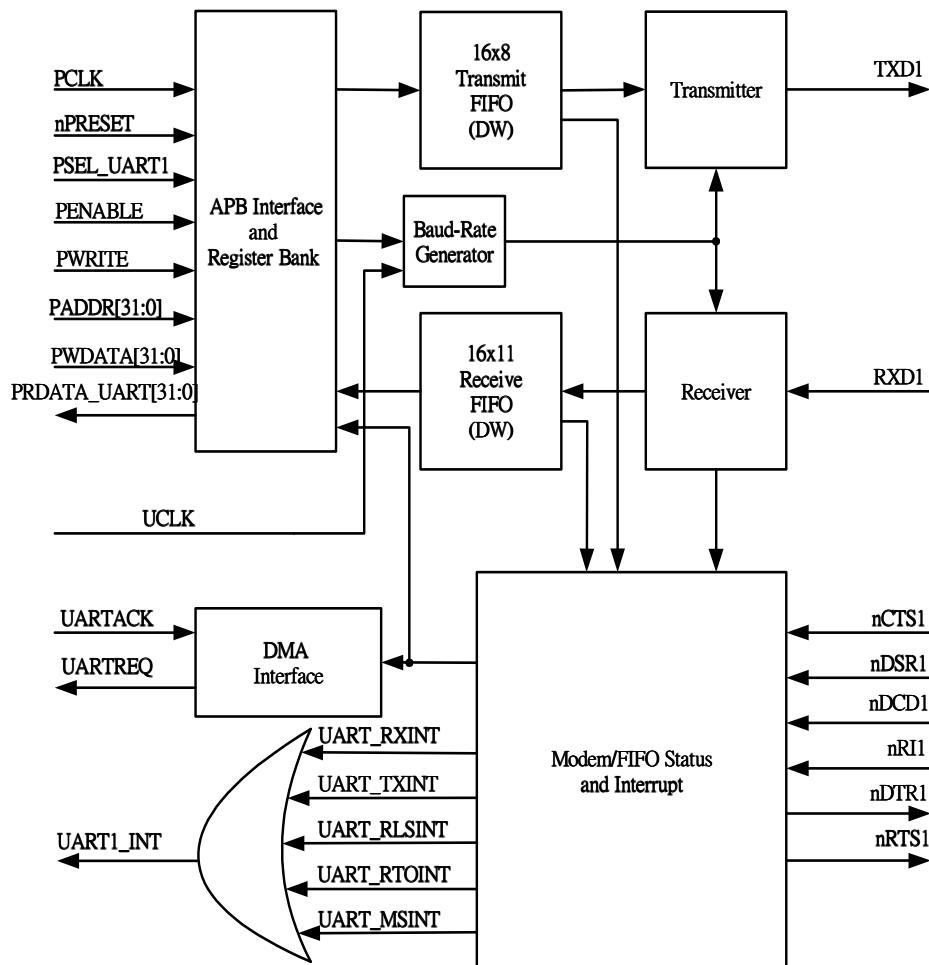
The CPU reads and writes data and control/status information through the APB interface. The transmitting and receiving paths are buffered with internal FIFO memories enabling up to 16-bytes to be stored independently in both transmit and receive modes. A baud rate generator can generate a common transmit and receive internal clock input. The baud rates will depend on the internal clock frequency. The UART will also provide transmit, receive and exception interrupts to system. A DMA interface is implemented for improving the system performance.

Key Features

- Compliance to AMBA APB specification
- Supports up to 115.2Kbps baud-rate
- Separate transmit and receive FIFO buffers (16 x 8) to reduce CPU interrupts
- Programmable baud rate generator. This enables division of the internal clock by (1 ~ 65535 x 16) and generates an internal x16 clock
- Standard asynchronous communication bits (start, stop and parity). These are added prior to transmission and removed on reception
- Independent masking of transmit FIFO, receive FIFO, and receive timeout and error condition interrupts
- False start bit detection
- Line break generation and detection
- Fully-programmable serial interface characteristics:
 - Data can be 5, 6, 7, 8 bits
 - Even, odd, stick or no-parity bit generation and detection
 - 1 or 1.5 or 2 stop bit generation
 - Baud rate generation

Architecture

Block Diagram



Block Descriptions

The UART consists of APB Slave interface for UART register configuration, The transfer data is buffered into transmit FIFO and transfer out by transmitter, and the received data is received by receiver and buffered into Receive FIFO, UART also support DMA request and interrupt to improve the system performance.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
UART_RBR	0x0000	W	0x00000000	Receiver FIFO output.

UART_THR	0x0000	W	-	Transmit FIFO input.
UART_IER	0x0004	W	0x00000000	Enable/Mask interrupts generated by the UART.
UART_IIR	0x0008	W	0x000000C1	Get interrupt information.
UART_FCR	0x0008	W	0x000000C0	Control FIFO options.
UART_LCR	0x000C	W	0x00000003	Line Control register.
UART_MCR	0x0010	W	0x00000000	Modem Controls register.
UART_LSR	0x0014	W	0x00000060	Line Status information.
UART_MSR	0x0018	W	0x00000000	Modem Status.

In addition, there are 2 Clock Divisor registers that together form one 16-bit.

The registers can be accessed when the 7th (DLAB) bit of the Line Control Register is set to '1'. At this time the above registers at addresses 0x00 and 0x04 can't be accessed.

Name	Offset	Size	Reset Value	Description
UART_DLL	0x0000	W	0x00000000	The LSB of the divisor latch.
UART_DLH	0x0004	W	0x00000000	The MSB of the divisor latch.

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

UART_RBR

Address: Operational Base + offset (0x0000)

Receive Buffer Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Any data words received by the UART from the serial link are accessed by reading this register. Bit 0 in the LSR line status register can be used to check if all received bytes have been read. This bit will change to zero if no more bytes are present

UART_THR

Address: Operational Base + offset (0x0000)

Transmitter holding register

Bit	Attr	Reset Value	Description
31:0	W	-	This register is used to buffer outgoing characters. Bit 5 in the LSR, line status register can be used to check if new information must be written to THR. The value 1 indicates that the register is empty. More than one character can be written to the transmitter holding register when the bit signals an empty state.

UART_IER

Address: Operational Base + offset (0x0004)

Interrupt enable register allows enabling and disabling interrupt generation by the UART.

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3	RW	0x0	Modem status Interrupt ‘0’ – disabled ‘1’ – enabled
2	RW	0x0	Receiver line status interrupt ‘0’ – disabled ‘1’ – enabled
1	RW	0x0	Transmitter holding register empty interrupt ‘0’ – disabled ‘1’ – enabled
0	RW	0x0	Received data available interrupt ‘0’ – disabled ‘1’ – enabled

UART_IIR

Address: Operational Base + offset (0x0008)

Interrupt identification register enables the programmer to retrieve what is the current highest priority pending interrupt.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:6	R	0x3	For compatibility reason. (FIFO enabled)
5:4	R	0x0	Reserved.
3:1	R	0x0	Indicates what the current highest priority pending interrupt is.
0	R	0x1	Indicates that an interrupt is pending or not. ‘0’ – an interrupt is pending. ‘1’ – no interrupt is pending.

The following table displays the list of possible interrupts along with the bits they enable, priority, and their source and reset control.

Bit 3	Bit 2	Bit 1	Priority	Interrupt Type	Interrupt Source	Interrupt Clear Control
0	1	1	1 st	Receiver line status	Parity, Overrun or Framing errors or Break Interrupt.	Reading the Line Status Register.
0	1	0	2 nd	Receive Data available	FIFO trigger level reached.	FIFO trigger level un-reached.
1	1	0	3 rd	Timeout indication	There's at least 1 character in the FIFO but no character has been input to the FIFO or read from it for the last 4 Char times.	Reading from the FIFO (Receiver Buffer Register).
0	0	1	4 th	Transmitter holding register empty	Transmitter Holding Register Empty.	Writing to the Transmitter Holding Register or reading the IIR or LSR.
0	0	0	5 th	Modem status	nCTS, nDSR, nRI or nDCD.	Reading the Modem status register.

UART_FCR

Address: Operational Base + offset (0x0008)

FIFO control register allows selection of the FIFO trigger level (the number of bytes in FIFO required to enable the Received Data Available interrupt). In addition, the FIFOs can be cleared using this register.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:6	W	0x3	Define the Receiver FIFO Interrupt trigger level. 0x0 – 1 byte 0x1 – 4 bytes 0x2 – 8 bytes 0x3 – 14 bytes
5:4	-	-	Reserved
3	W	0x0	Hardware request to DMA. '0' – disable. '1' – enable.
2	W	0x0	Writing a '1' to bit 2 clears the Transmitter FIFO and resets its logic. The shift register is not cleared, i.e. transmitting of the current character continues.
1	W	0x0	Writing a '1' to bit 1 clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, i.e. receiving of the current character continues.
0	W	0x0	Reserved.

UART_LCR

Address: Operational Base + offset (0x000C)

The line control register allows the specification of the format of the asynchronous data communication used. A bit in the register also allows access to the Divisor Latches, which define the baud rate. Reading from the register is allowed to check the current settings of the communication.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7	RW	0x0	Divisor Latch Access bit. '1' – The divisor latches can be accessed. '0' – The normal registers are accessed.
6	RW	0x0	Break Control bit. '1' – the serial out is forced into logic '0' (break state). '0' – break is disabled.
5	RW	0x0	Stick Parity bit. '0' – Stick Parity disabled. '1' - If bits 3 and 4 are logic '1', the parity bit is transmitted and checked as logic '0'. If bit 3 is '1' and bit 4 is '0' then the parity bit is transmitted and checked as '1'.
4	RW	0x0	Even Parity select. '0' – Odd number of '1' is transmitted and checked in each word (data and parity combined). In other words, if the data has an even number of '1' in it,

			then the parity bit is '1'. '1' – Even number of '1' is transmitted in each word.
3	RW	0x0	Parity Enable. '0' – No parity. '1' – Parity bit is generated on each outgoing character and is checked on each incoming one.
2	RW	0x0	Specify the number of generated stop bits. '0' – 1 stop bit. '1' – 1.5 stop bits when 5-bit character length selected and 2 bits otherwise Note that the receiver always checks the first stop bit only.
1:0	RW	0x3	Select number of bits in each character. 0x0 – 5 bits. 0x1 – 6 bits. 0x2 – 7 bits. 0x3 – 8 bits.

UART_MCR

Address: Operational Base + offset (0x0010)

The modem control register allows transferring control signals to a modem connected to the UART.

Bit	Attr	Reset Value	Description
31:5	-	-	Reserved.
4	RW	0x0	Loopback mode. '0' – normal operation. '1' – loopback mode. When in loopback mode, the Serial Output Signal (TXD) is set to logic '1'. The signal of the transmitter shift register is internally connected to the input of the receiver shift register. The following connections are made: nDTR → nDSR nRTS → nCTS Out1 → nRI Out2 → nDCD
3	RW	0x0	Out2. In loopback mode, connected to Data Carrier Detect (nDCD) input.
2	RW	0x0	Out1. In loopback mode, connected Ring Indicator (nRI) signal input.
1	RW	0x0	Request To Send (nRTS) signal control. '0' – nRTS is '1' '1' – nRTS is '0'
0	RW	0x0	Data Terminal Ready (nDTR) signal control. '0' – nDTR is '1' '1' – nDTR is '0'

UART_LSR

Address: Operational Base + offset (0x0014)

Line status register

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7	R	0x0	‘1’ – At least one parity error, framing error or break indications have been received and are inside the FIFO. The bit is cleared upon reading from the register. ‘0’ – Otherwise.
6	R	0x1	Transmitter Empty indicator. ‘1’ – Both the transmitter FIFO and transmitter shift register are empty. The bit is cleared upon reading from the register or upon writing data to the transmit FIFO. ‘0’ – Otherwise
5	R	0x1	Transmit FIFO is empty. ‘1’ – The transmitter FIFO is empty. Generates Transmitter Holding Register Empty interrupt. The bit is cleared in the following cases: The LSR has been read, the IIR has been read or data has been written to the transmitter FIFO. ‘0’ – Otherwise.
4	R	0x0	Break Interrupt (BI) indicator. ‘1’ – A break condition has been reached in the current character. The break occurs when the line is held in logic 0 for a time of one character (start bit + data + parity + stop bit). In that case, one zero character enters the FIFO and the UART waits for a valid start bit to receive next character. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt. ‘0’ – No break condition in the current character.
3	R	0x0	Framing Error (FE) indicator. ‘1’ – The received character at the top of the FIFO did not have a valid stop bit. The UART core tries re-synchronizing by assuming that the bit received was a start bit. Of course, generally, it might be that all the following data is corrupt. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt. ‘0’ – No framing error in the current character
2	R	0x0	Parity Error (PE) indicator. ‘1’ – The character that is currently at the top of the FIFO has been received with parity error. The bit is cleared upon reading from the register. Generates Receiver Line Status interrupt. ‘0’ – No parity error in the current character
1	R	0x0	Overrun Error (OE) indicator. ‘1’ – If the FIFO is full and another character has been received in the receiver shift register. If another character is starting to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. The bit is cleared upon reading from the register. Generates Receiver Line

			Status interrupt. '0' – No overrun state.
0	R	0x0	Data Ready (DR) indicator. '0' – No characters in the FIFO. '1' – At least one character has been received and is in the FIFO. Reset when all data has read out from FIFO.

UART_MSR

Address: Operational Base + offset (0x0018)

The register displays the current state of the modem control lines. Also, four bits also provide an indication in the state of one of the modem status lines. These bits are set to '1' when a change in corresponding line has been detected and they are reset when the register is being read.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7	R	0x0	Complement of the nDCD input or equals to Out2 (MCR[3]) in loopback mode.
6	R	0x0	Complement of the nRI input or equals to Out1 (MCR[2]) in loopback mode.
5	R	0x0	Complement of the nDSR input or equals to nDTR (MCR[0]) in loopback mode.
4	R	0x0	Complement of the nCTS input or equals to nRTS (MCR[1]) in loopback mode.
3	R	0x0	Delta Data Carrier Detect (DDCD) indicator '1' – The nDCD line has changed its state.
2	R	0x0	Delta Ring Indicator (DRI) detector '1' – The nRI line has changed its state. Note: The definition is not as same as Trailing Edge of Ring Indicator (TERI) detector.
1	R	0x0	Delta Data Set Ready (DDSR) indicator '1' – The nDSR line has changed its state.
0	R	0x0	Delta Clear To Send (DCTS) indicator '1' – The nCTS line has changed its state.

UART_DLL/UART_DLH

Setting the 7th bit of LCR to '1' can access the divisor latches. You should restore this bit to '0' after setting the divisor latches in order to restore access to the other registers that occupy the same addresses. The 2 bytes form one 16-bit register, which is internally accessed as a single number. You should therefore set all 2 bytes of the register to ensure normal operation. The register is set to the default value of 0 during reset, which disables all serial I/O operations in order to ensure explicit setup of the register in the software. The value set should be equal to (system clock speed) / (16 x desired baud rate).

The internal counter starts to work when the LSB of DL is written, so when setting the divisor, write the MSB first and the LSB last.

UART_DLL

Address: Operational Base + offset (0x0000)

Divisor Latch LSB

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	The LSB of the baud-rate divisor latch.

UART_DLH

Address: Operational Base + offset (0x0004)

Divisor Latch MSB

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	The MSB of the baud-rate divisor latch.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Clock Signals

The frequency selected for UCLK must accommodate the desired range of baud rates:
 $16 \times 65535 \times \text{baud rate} \geq F(\text{UCLK}) \geq 16 \times \text{baud rate}$

Since the baud divisor latch is a 16-bit register, the divisor value is ranging from 1 to 65535. The minimum value must be set as 1. Setting 0 to the divisor is not allowed. The frequency of UCLK must also be within the required error limits for all baud rates to be used. The UART reference clock is from external UCLK for most accurate result.

Operation

Control data is written to the UART line control register, UART_LCR.

UART_LCR defines:

- Transmission parameters
- Word length
- Number of transmission stop bits
- Parity mode
- Break generation

UART_DLL and UART_DLH define the baud rate divisor.

An internal clock enable signal is generated, and is a stream of one UCLK wide pulses with an average frequency of 16 times the desired baud rate. This signal is then divided by 16 to give the transmit clock.

Data Transmission or Reception

Data received or transmitted is stored in two 16-byte FIFOs. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in UART_LCR. Data continues to be transmitted until there is no data left in the transmit FIFO. There are 16 times samples of a UART bit. For each sample of bit, three readings are taken and the majority value is kept. In the following, the middle sampling point is defined, and one sample is took

either side of it.

When the receiver is idle (RXD continuously 1) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by Baud16, begins running and data is sampled on the eighth cycle of that counter in normal UART mode, or the fourth cycle of counter in SIR mode to allow for the shorter logic 0 pulses (half way through a bit period).

The start bit is valid if RXD is still LOW on the eighth cycle of sample clock, otherwise a false start bit is detected and it is ignored.

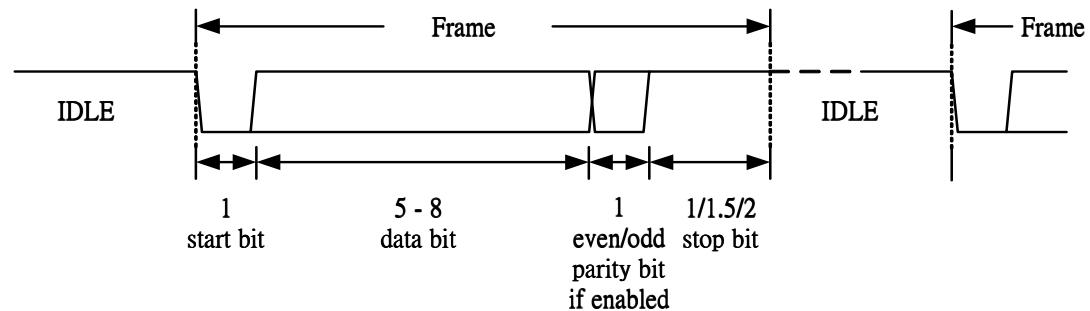
If the start bit was valid, successive data bits are sampled on every 16th cycle of sample clock (one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.

Finally, a valid stop bit is confirmed if RXD is HIGH, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO.

Overrun Condition

The overrun error is set when the FIFO is full, and the next character is completely received in the shift register. The data in the shift register is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO, and another character is received, the received character is copied into the receive FIFO. The overrun state is then cleared.

UART Packet Frame



Chapter 16 General Purpose IO (GPIO)

Overview

The General Purpose IO (GPIO) is an APB slave module that connects to the APB bus. The GPIO has 16 bits programmable input/output organized as four ports, port A, and Port B. Pins of both ports can be configured as either inputs or outputs.

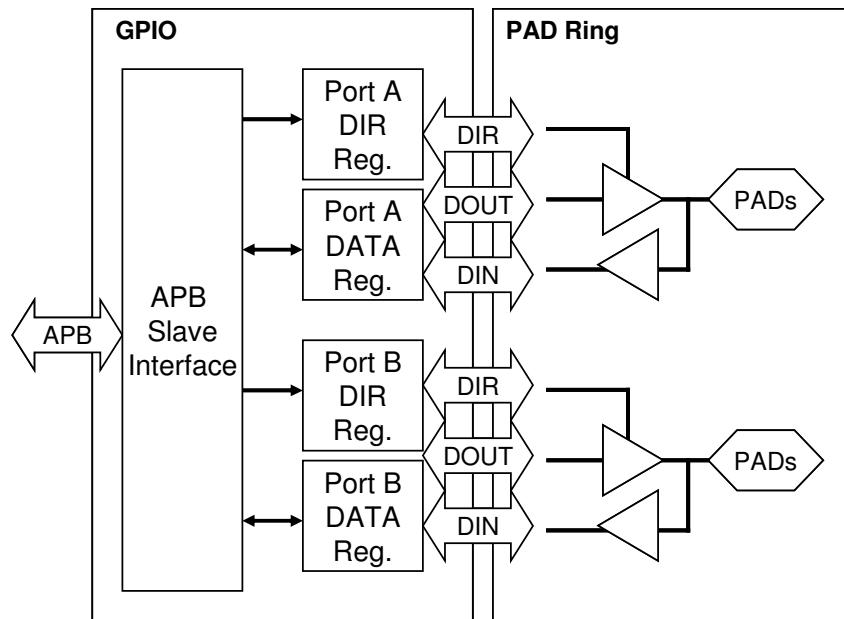
Some of the ports are shared with the function pins. On system reset, all ports are inputs as default. The GPIO interfaces with input/output PAD cells using a data input, data output and output enable line per pin.

Key Features

- Compliance to the AMBA APB interface
- 16 individually programmable input/output pins
- Control word read-back capability
- All ports are inputs on system reset
- The direction registers are programmable

Architecture

Block Diagram



Block Descriptions

The GPIO has up to 16 bits programmable input/output organized as four ports, port A, and port B. The CPU reads and writes data and control/status information to and from

GPIO via APB interface.

Each port has an associated:

- Data register
 - The data register is 8 bits wide and is used to
 - Read the input value on GPIO lines that are configured as inputs
 - Program the output value on GPIO lines that are configured as outputs
- Data direction register
 - The data direction register is 8 bits wide and is programmed to select whether individual input/output pin is configured as an input or an output.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
GPIO_PADR	0x0000	W	0x00000000	Port A data register.
GPIO_PACON	0x0004	W	0x00000000	Port A direction register.
GPIO_PBDR	0x0008	W	0x00000000	Port B data register.
GPIO_PBCON	0x000C	W	0x00000000	Port B direction register.
GPIO_TEST	0x0020	W	0x00000000	GPIO function test register.
GPIO_JEA	0x0024	W	0x00000000	Port A interrupt mask register.
GPIO_JEB	0x0028	W	0x00000000	Port B interrupt mask register.
GPIO_ISA	0x0034	W	0x00000000	Port A interrupt sense register.
GPIO_ISB	0x0038	W	0x00000000	Port B interrupt sense register.
GPIO_IBEA	0x0044	W	0x00000000	Port A interrupt both-edges register.
GPIO_IBEB	0x0048	W	0x00000000	Port B interrupt both-edges register.
GPIO_IEVA	0x0054	W	0x00000000	Port A interrupt event register.
GPIO_IEVB	0x0058	W	0x00000000	Port B interrupt event register.
GPIO_ICA	0x0064	W	0x00000000	Port A interrupt clear register.
GPIO_ICB	0x0068	W	0x00000000	Port B interrupt clear register.
GPIO_ISR	0x0074	W	0x00000000	GPIO interrupt status register.

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

GPIO_PxDR(x=A, B)

Address: Operational Base + offset (0x0000/0x0008)

The GPIO_PxDR is the Port x data register.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x data register.

GPIO_PxCON(x=A, B)

Address: Operational Base + offset (0x0004/0x000C)

The GPIO_PxCON is the Port x data direction register. When asserted HIGH, the corresponding pin is configured as output. On the other hand, it is configured as input. At reset, all bits are configured as input.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x direction register. 0: Input 1: Output

GPIO_TEST

Address: Operational Base + offset (0x0020)

GPIO function test register.

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:1	RW	0x0	Test mode. 0x0 – PORTB -> PORTA 0x1 – PORTA -> PORTB 0x2 – Reserved. 0x3 – Reserved.
0	RW	0x0	Test mode enable indicator. ‘1’ – The GPIO loop-back test mode enable. ‘0’ – Normal mode.

GPIO_IEx (x=A, B)

Address: Operational Base + offset (0x0024/0x0028)

The GPIO_IEx register is the Port x interrupt mask register. Bits set to HIGH in GPIO_IEx allow the corresponding pins to trigger their individual interrupts and the combined **gpio_int** line. On the other hand, bits set to LOW represent disable interrupt trigger on the pins. The default value is set to LOW at reset.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x interrupt mask register. 0: Masked 1: Not masked

GPIO_ISx (x=A, B)

Address: Operational Base + offset (0x0034/0x0038)

The GPIO_ISx register is the Port x interrupt sense register. The default set is edge sensitive at reset.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x interrupt sense register. 0: Edge on corresponding pin is detected. 1: Level on corresponding pin is detected.

GPIO_IBEx (x=A, B)

Address: Operational Base + offset (0x0044/0x0048)

The GPIO_IBEx register is the Port x interrupt both-edges register. When the bits in GPIO_ISx are set to edge sensitive, bits set to HIGH in GPIO_IBEx configure the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the GPIO_IEVx. The default value is set to LOW at reset.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x interrupt both-edges register. 0: Single edge, interrupt generation event is controlled by GPIO_IEVx. 1: Both edges on corresponding pin trigger an interrupt.

GPIO_IEVx (x=A, B)

Address: Operational Base + offset (0x0054/0x0058)

The GPIO_IEVx register is the Port x interrupt event register. When the corresponding bit is asserted to HIGH in GPIO_IEVx, it is configured to detect rising edges or high levels, depending on the corresponding bit value in GPIO_ISx. Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in GPIO_ISx. The default value is set to LOW at reset.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x interrupt event register. 0: Falling edges or low levels on corresponding pin trigger interrupts. 1: Rising edges or high levels on corresponding pin trigger interrupts.

GPIO_ICx (x=A, B)

Address: Operational Base + offset (0x0064/0x0068)

The GPIO_ICx register is the Port x interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt edge detection logic register. Writing a 0 has no effect. The register is write-only and all bits are cleared by a reset.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	W	0x0	Port x interrupt clear register. 0: No effect. 1: Clear the corresponding interrupt detection logic register.

GPIO_ISR

Address: Operational Base + offset (0x0074)

The GPIO_ISR register is the interrupt status registers. Bits read HIGH in this register reflect the status of interrupts trigger conditions detected. Bits read LOW indicate that corresponding input pins have not initiated an interrupt. The register is read-only and all bits are cleared by a reset.

Bit	Attr	Reset Value	Description

31:16	-	-	Reserved.
15:8	R	0x0	Port B interrupt status register. The status of interrupt triggers condition detection on pins. 0: GPIO interrupt not active. 1: GPIO asserting interrupt.
7:0	R	0x0	Port A interrupt status register. The status of interrupt triggers condition detection on pins. 0: GPIO interrupt not active. 1: GPIO asserting interrupt.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

For each port, there is a data register; a mode register and five interrupt control registers. On reads, the data register holds the current status of the corresponding port pins, whether they are configured as inputs or outputs. Writing to a data register only affects the pins that are configured as outputs. Those function pins shared with GPIO ports are not stored in the registers. They will be directly wired to their source modules.

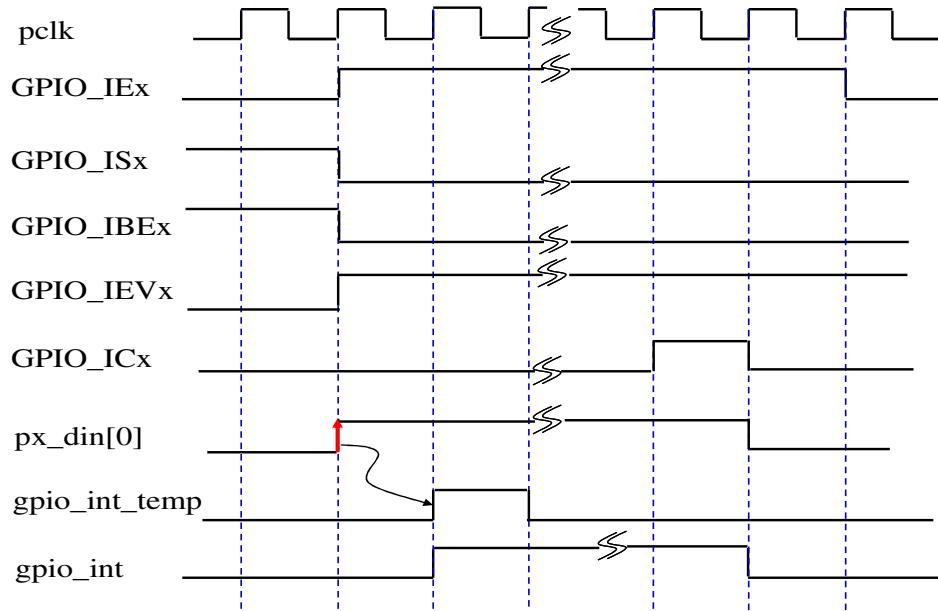
The interrupt section of the GPIO is controlled by a set of twenty registers. When one or more GPIO lines causes an interrupt, a single interrupt output **gpio_int** and/or the individual interrupts can be sent to the interrupt controller.

Programming Sequence

Rising edge detected

1. Bits set to “0” in GPIO_ISx, edge on corresponding pin is detected.
2. Bits set to “0” in GPIO_IBEx configure the corresponding pin to detect single edge.
3. Bits set to “1” in GPIO_IEx allow the corresponding pins to trigger their individual interrupts.
4. Bits set to “1” in GPIO_IEx allow the corresponding pins to trigger their individual interrupts.
5. When one or more input data pins cause a rising edge-triggered interrupt, interrupt output **gpio_int** can be set to “1”.

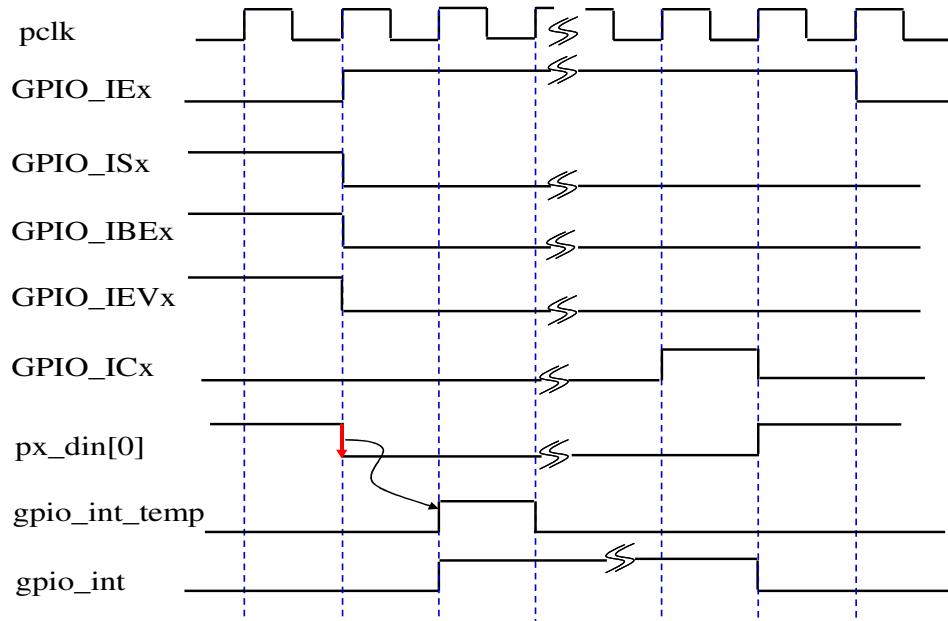
Rising edge detected timing



Falling edge detected

1. Bits set to “0” in `GPIO_ISx`, edge on corresponding pin is detected.
2. Bits set to “0” in `GPIO_IBEx` configure the corresponding pin to detect single edge.
3. Bits set to “0” in `GPIO_IEVx` configure the corresponding pin to detect rising edge.
4. Bits set to “1” in `GPIO_IEx` allow the corresponding pins to trigger their individual interrupts.
5. When one or more input data pins cause a falling edge-triggered interrupt, interrupt output `gpio_int` can be set to “1”.

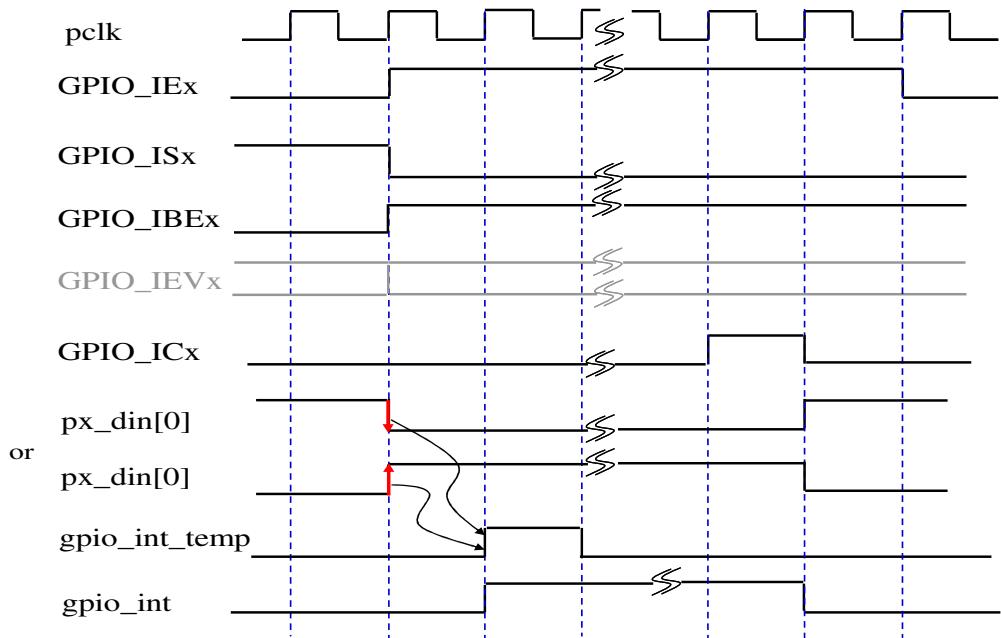
Falling edge detected timing



Both-edges detected

1. Bits set to “0” in GPIO_ISx, edge on corresponding pin is detected.
2. Bits set to “1” in GPIO_IBEx configure the corresponding pin to detect both edges.
3. Bits set to “0” or “1” in GPIO_IEVx are okay.
4. Bits set to “1” in GPIO_IEEx allow the corresponding pins to trigger their individual interrupts.
5. When one or more input data pins cause a falling edge-triggered or rising edge-triggered interrupt, interrupt output gpio_int can be set to “1”.

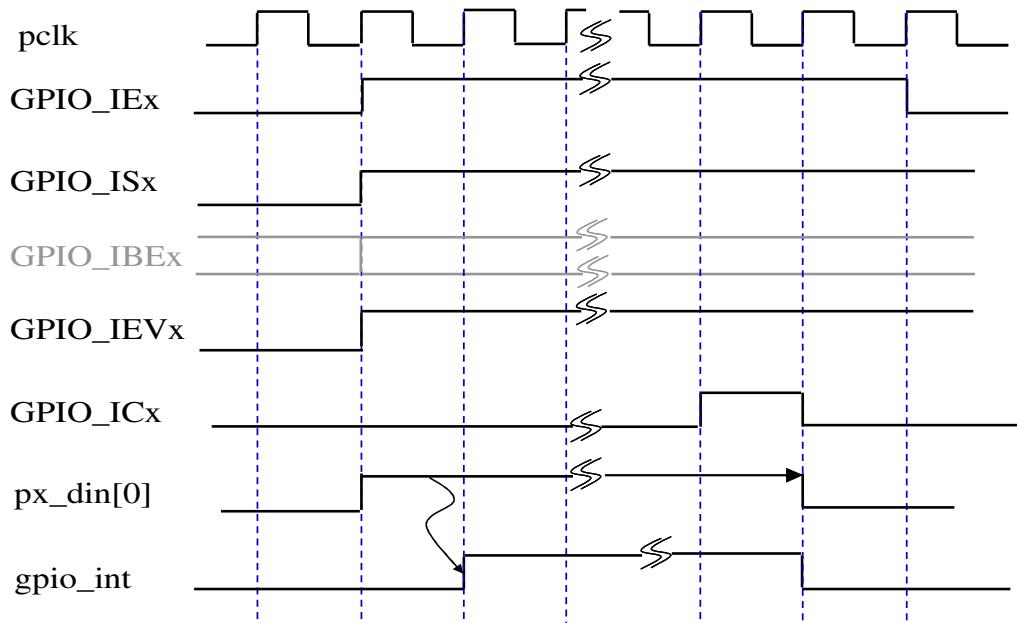
Both-edges detected timing



High level detected

1. Bits set to “1” in GPIO_ISx, level on corresponding pin is detected.
2. Bits set to “0” or “1” in GPIO_IBEx are okay.
3. Bits set to “1” in GPIO_IEVx configure the corresponding pin to detect high level.
4. Bits set to “1” in GPIO_IEx allow the corresponding pins to trigger their individual interrupts.
5. When one or more input data pins causes a high level interrupt, interrupt output gpio_int can be set to “1”.

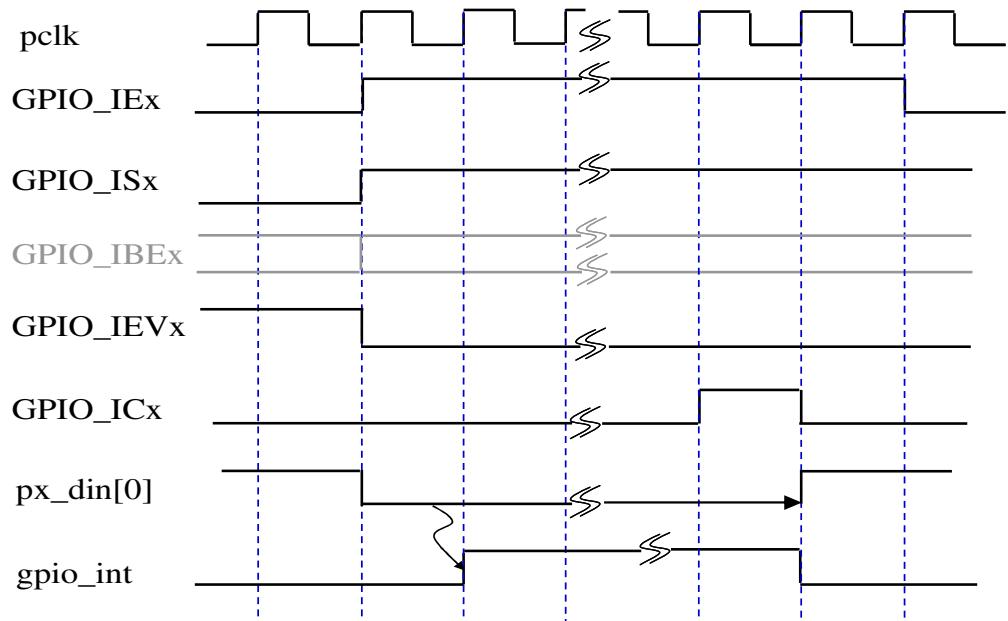
High level detected timing



Low level detected

1. Bits set to “1” in **GPIO_ISx**, level on corresponding pin is detected.
2. Bits set to “0” or “1” in **GPIO_IBEx** are okay.
3. Bits set to “0” in **GPIO_IEVx** configure the corresponding pin to detect low level.
4. Bits set to “1” in **GPIO_IEx** allow the corresponding pins to trigger their individual interrupts.
5. When one or more input data pins causes a low level interrupt, interrupt output **gpio_int** can be set to “1”.

Low level detected timing



Chapter 17 Internal Timers

Overview

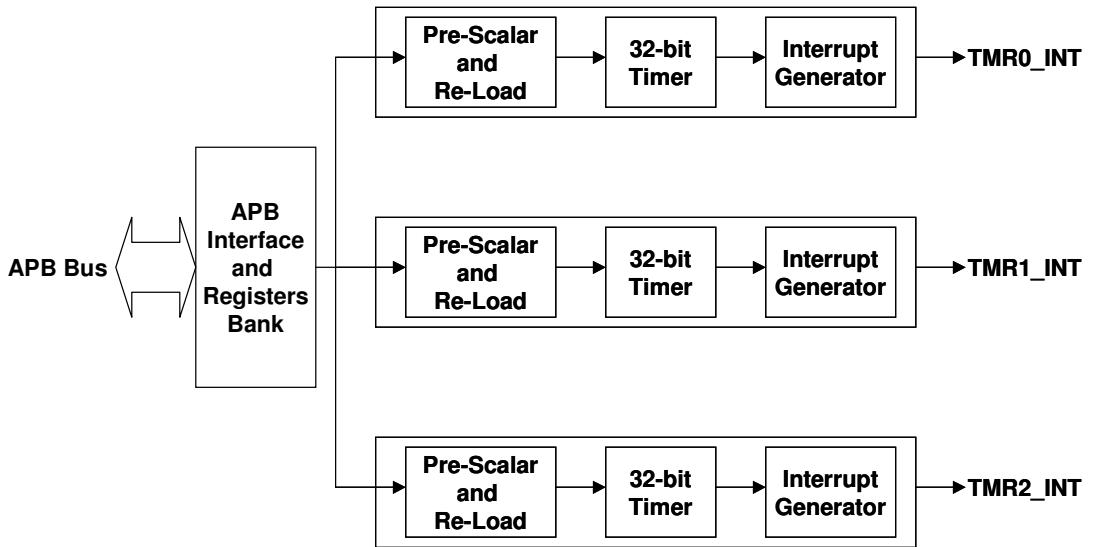
The timer module is an APB slave. This module is composed of 3 timers, TMR0, TMR1 and TMR2. All the TMRs can be used as interval timers to generate periodical interrupts. Timer can operate under free-running mode or periodical mode. When running in periodical mode, the counting value will be loaded into counter via APB interface. The timer will count from the loaded value. When zero is reached, the timer will generate an interrupt then the loaded value will be reloaded into counter. When running in free-running mode, timer will count from its maximum value (0xFFFF_FFFF) down to zero then wraps around. When zero reached, an interrupt will be generated. The interrupts of TMR0, TMR1 and TMR2 are all mask-able.

Key Features

- Built-in three 32 bits timer modules
- Provide independent free-running or periodical mode
- Mask-able interrupts
- Programmable counter

Architecture

Block Diagram



Block Descriptions

The Timer consists of a APB slave interface for timer register setting and three separate 32 bit timer, each timer can be set for different interval time and generate it's own periodical interrupt.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
TMR0LR	0x0000	W	0x00000000	Load register.
TMR0CVR	0x0004	W	0x00000000	Current value register.
TMR0CON	0x0008	W	0x00000002	Control register.
TMR1LR	0x0010	W	0x00000000	Load register.
TMR1CVR	0x0014	W	0x00000000	Current value register.
TMR1CON	0x0018	W	0x00000002	Control register.
TMR2LR	0x0020	W	0x00000000	Load register.
TMR2CVR	0x0024	W	0x00000000	Current value register.
TMR2CON	0x0028	W	0x00000002	Control register.

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

TMR_xLR(x=0,1,2)

Address: Operational Base + offset (0x0000, 0x0010, 0x0020)

Load register for timer counter

Bit	Attr	Reset Value	Description
31:0	W	0x0	Load register for timer counter. Counting value ranges from 0 ~ $(2^{32} - 1)$.

TMRxCVR(x=0,1,2)

Address: Operational Base + offset (0x0004, 0x0014, 0x0024)

Current value register

Bit	Attr	Reset Value	Description
31:0	R	0x0	The current counter value.

TMRxCON(x=0,1,2)

Address: Operational Base + offset (0x0008, 0x0018, 0x0028)

Control register

Bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10	RW	0x0	Counter enable/disable 0: Enable 1: Disable
9	-	-	Reserved.
8	RW	0x0	Timer enable/disable. 0: Disable 1: Enable
7	RW	0x0	Timer counting mode selection. 0: Free-Running 1: Periodical
6:4	RW	0x0	Prescale factor. 0x0: 1 0x1: 1/4 0x2: 1/8 0x3: 1/16 0x4: 1/32 0x5: 1/64 0x6: 1/128 0x7: 1/256
3	RW	0x0	Interrupt mask. 0: Disable 1: Enable
2	RW	0x0	Interrupt clear. This bit is set when an interrupt is pending. Writing a '0' to this bit will clear the interrupt
1	RW	0x1	Interrupt trigger type. 0: Edge trigger 1: Level trigger
0	-	-	Reserved.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

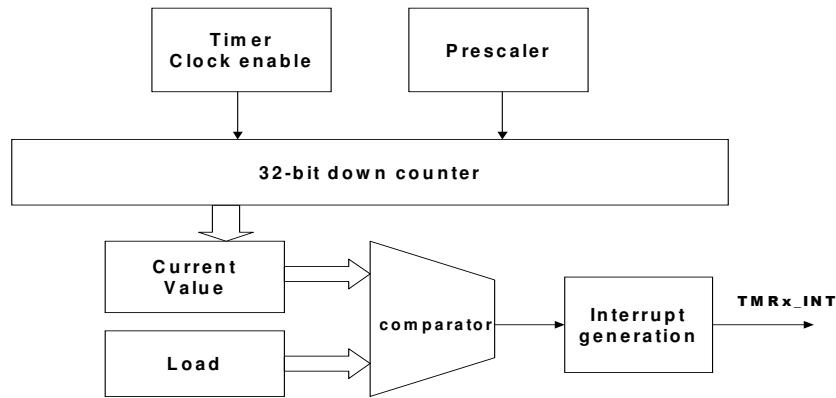
Operation

The timer is loaded by writing to the TMRxLR and, if enabled, counting down to zero. When zero is reached, an interrupt is generated. Writing to the interrupt control bit of the TMRxCON can clear the interrupt.

After reaching a zero count, if the timer is operating in free-running mode it continues to decrement from its maximum value ($2^{32} - 1$). If periodical mode is selected, the timer reloads the count value from the TMRxLR and continues to decrement. Under this mode, the counter effectively generates a periodical interrupt. The mode is selected by setting the bit 6 of the TMRxCON.

At any point, the current counter value may be read from the TMRxCVR via APB interface.

The entire scenario can be explained by the figure as below:



A pre-scale unit generates the timer clock. The enable is then used by the counter to create a clock with a timing of one of the following:

- PCLK
- PCLK/4, PCLK/8, PCLK/16, PCLK/32, PCLK/64, PCLK/128, PCLK/256

The pre-scale value can be determined by programming the bit6 ~ bit4 in TMRxCON register.

The counter clock frequency can be evaluated by the formula as given below:

$$F_{\text{timer}} = F_{\text{PCLK}} / (\text{Pre-scale} * \text{Counter Value})$$

Chapter 18 PWM Timer

Overview

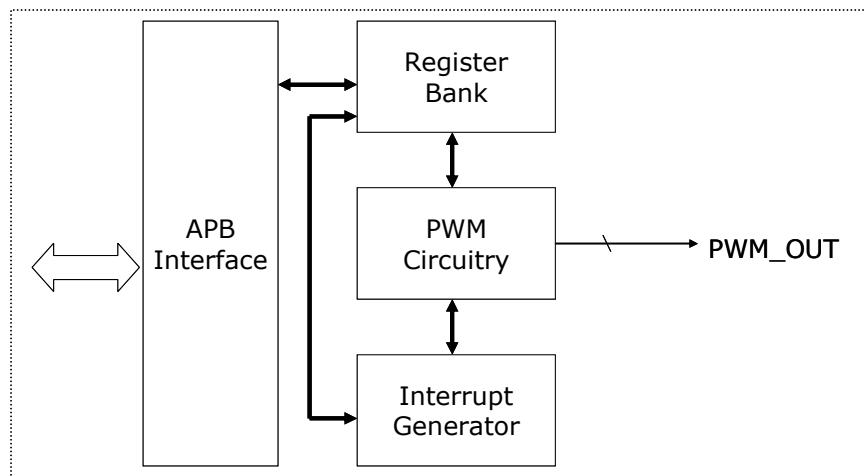
There are four PWM blocks in PWM Timer (PWM0, PWM1, PWM2, and PWM3). Each PWM block built-in 5-bit pre-scalar from PCLK. The PWM Timer supports both reference mode, which can output various duty-cycle waveforms, and capture, which can measure the duty-cycle of input waveform.

Key Features

- Programmable 5-bit pre-scalar
- 32-bit timer/counter facility
- Single-run or continues-run PWM mode
- Support maskable interrupt

Architecture

Block Diagram



Block Descriptions

PWM Register Block

This block controls the setting of PWM mode.

PWM Circuitry

This block includes clock pre-scalar and reference comparator for PWM timer.

Interrupt Generator

This block handles the interrupt generation, masking, and clearing.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
PWMT0_CNTR	0x0000	W	0x00000000	Main counter register
PWMT0_HRC	0x0004	W	0x00000000	PWM HIGH Reference/Capture register
PWMT0_LRC	0x0008	W	0x00000000	PWM LOW Reference/Capture register
PWMT0_CTRL	0x000C	W	0x00000000	PWM control register
PWMT1_CNTR	0x0010	W	0x00000000	Main counter register
PWMT1_HRC	0x0014	W	0x00000000	PWM HIGH Reference/Capture register
PWMT1_LRC	0x0018	W	0x00000000	PWM LOW Reference/Capture register
PWMT1_CTRL	0x001C	W	0x00000000	PWM control register
PWMT2_CNTR	0x0020	W	0x00000000	Main counter register
PWMT2_HRC	0x0024	W	0x00000000	PWM HIGH Reference/Capture register
PWMT2_LRC	0x0028	W	0x00000000	PWM LOW Reference/Capture register
PWMT2_CTRL	0x002C	W	0x00000000	PWM control register
PWMT3_CNTR	0x0030	W	0x00000000	Main counter register
PWMT3_HRC	0x0034	W	0x00000000	PWM HIGH Reference/Capture register
PWMT3_LRC	0x0038	W	0x00000000	PWM LOW Reference/Capture register
PWMT3_CTRL	0x003C	W	0x00000000	PWM control register

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

PWMTx_CNTR (x=0~3)

Address: Operational Base + offset (0x0000/0x0010/0x0020/0x0030)

PWM0 timer counter.

Bit	Attr	Reset Value	Description
31:0	RW	0	Main PWM timer counter. Counting value ranges from 0 ~ $(2^{32} - 1)$.

PWMTx_HRC (x=0~3)

Address: Operational Base + offset (0x0004/0x0014/0x0024/0x0034)

PWM0 HIGH reference or capture register

Bit	Attr	Reset Value	Description
31:0	RW	0	PWM HIGH reference/capture registers

PWMTx_LRC (x=0~3)

Address: Operational Base + offset (0x0008/0x0018/0x0028/0x0038)

PWM0 LOW reference or capture register

Bit	Attr	Reset Value	Description
31:0	RW	0	PWM LOW reference/capture registers

PWMTx_CTRL (x=0~3)

Address: Operational Base + offset (0x000C/0x001C/0x002C/0x003C)

This control register of PWM0 Timer.

Bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13:9	R/W	0	Prescale factor. The counter clock is divided by the prescale factor. Counter clock = PCLK/2 ^{PWMx_CTRL[13:9]+1}
8	R/W	0	Capture mode enable/disable 0: Disable 1: Enable
7	R/W	0	PWM reset. 0: Normal operation 1: Reset PWM
6	R/W	0	Interrupt status and clear bit. Write "1" to clear interrupt status.
5	R/W	0	PWM timer interrupt enable/disable. PWM timer will assert an interrupt when PWMTx_CNTR value is equal to the value of PWMTx_LRC or PWMTx_HRC. 0: Disable 1: Enable
4	R/W	0	Single counter mode. 0: PWMTx_CNTR is restarted after it reaches value equal to the PWMTx_LRC value. 1: PWMTx_CNTR is not increased anymore after it reaches value equal to the PWMT0_LRC value.
3	R/W	0	PWM output enable/disable. 0: Disable 1: Enable
2:1	-	-	Reserved.
0	R/W	0	PWM timer enable/disable. 0: Disable 1: Enable

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Function Description

Operation

The PWM timer is the Pulse-width modulation that can be programmed by controlling the PWMx_CTRL registers. Operating the PWM Timer, users should set the

PWMx_HRC and the PWMx_LRC with the value of low and high periods of the PWM Timer output data. PWMx_HRC will be the number of clock cycles after reset of the PWMx_CNTR when PWM output should go high. Also, PWMx_LRC will be the number of clock cycles after reset of the PWMx_CNTR when PWM output should go low. PWMx_CNTR can be reset with the hardware reset.

Pre-Scale function

The PWM Time has the timing pre-scale function to divide the PCLK clock source before operating the PWM Timer. A pre-scale unit generates the timer clock. The enable is then used by the counter to create a clock with a timing of one of the following:

$$\text{Counter Clock} = \text{PCLK}/2^{\text{PWMx_CTRL[13:9]+1}}$$

The pre-scale value can be determined by programming the bit13 ~ bit9 in PWMTx_CTRL registers.

For PWM stabilization, do not change pre-scale value during the system operation. And reset the system before changing pre-scale value.

Interrupt Feature

Whenever the value of the PWMx_HRC or PWMx_LRC equals to the value of the PWMx_CNTR, an interrupt request can be asserted. The bit 6 in PWMTx_CTRL registers can be set to clear interrupt status.

Chapter 19 Watchdog Timer (WDT)

Overview

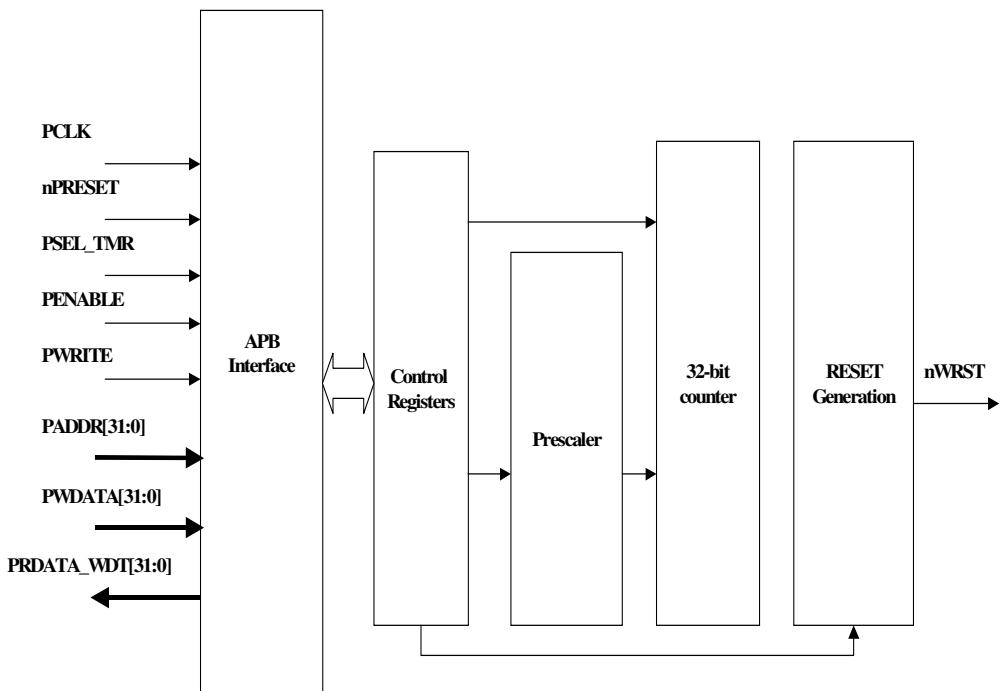
The watchdog timer (WDT) module is an APB slave, providing access to the programmable 32-bit counters. It provides a watchdog function to avoid system malfunction or software failure. Once the watchdog timer is programmed and the system abnormal situations occurred, the watchdog timer would generate a reset signal.

Key Features

- Watchdog function
- Built-in 32 bits programmable reset counter

Architecture

Block Diagram



Block Descriptions

The WDT consist of an APB slave interface for WDT register setting and the Prescaler and a 32 bit timer will generate a reset signal(nWRST) according to the register setting.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
WDTLR	0x0000	W	0x00000000	Load register.
WDTCVR	0x0004	W	0x0000FFFF	Current value register.
WDTCON	0x0008	W	0x00000000	Control register.

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

WDTLR

Address: Operational Base + offset (0x0000)

Load register for timer counter

Bit	Attr	Reset Value	Description
31:0	W	0x0	Load register for timer counter. Counting value ranges from $2^{32} - 1$.

WDTCVR

Address: Operational Base + offset (0x0004)

Current value register.

Bit	Attr	Reset Value	Description
31:0	R	0xFFFF	The current counter value.

WDTCON

Address: Operational Base + offset (0x0008)

Control register

Bit	Attr	Reset Value	Description
31:5	RW	-	Reserved.
4	RW	0x0	Reset enable/disable. Enable or disable bit of watchdog timer output for reset signal. 0: Disable the reset function of the watchdog timer. 1: Enable reset signal of the μplatform core at watchdog timeout.
3	RW	0x0	Watchdog Timer enable/disable. 0: Disable 1: Enable
2:0	RW	0x0	Prescale factor. 0x0: 1 0x1: 1/4 0x2: 1/8 0x3: 1/16 0x4: 1/32 0x5: 1/64 0x6: 1/128 0x7: 1/256

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

When watchdog timer is enabled and the counting value is loaded into counter, it starts to decrement the loaded value until zero is reached. System software must refresh the contents of the counter before zero is reached. If software fails to refresh the counter, the watchdog will generate a reset to resume the entire system. For debugging or performance purpose, programming the register bit in WDTCON can disable this reset. The system refresh requirement and the APB clock can determine the counter value and prescaler value.

A pre-scale unit generates the timer clock. The enable is then used by the counter to

create a clock with a timing of one of the following:

- PCLK
- PCLK/4, PCLK/8, PCLK/16, PCLK/32, PCLK/64, PCLK/128, PCLK/256

The pre-scale value can be determined by programming the bit6 ~ bit4 in WDTCON register.

Chapter 20 Real Time Clock (RTC)

Overview

The Real Time Clock (RTC) is an APB slave device. It can be used to provide a basic alarm function or long time base counter. This is achieved by generating an interrupt signal after counting a programmed number of cycles of a real time clock input. Counting in one-second intervals is achieved by use of a 1 Hz clock that is coming from the clock divider. The RTC also provides a system power on/off sequence triggered by CPU and can further control the system power through output control pin. The RTC core power is independent from system power so the RTC counter can running continuously during system power off.

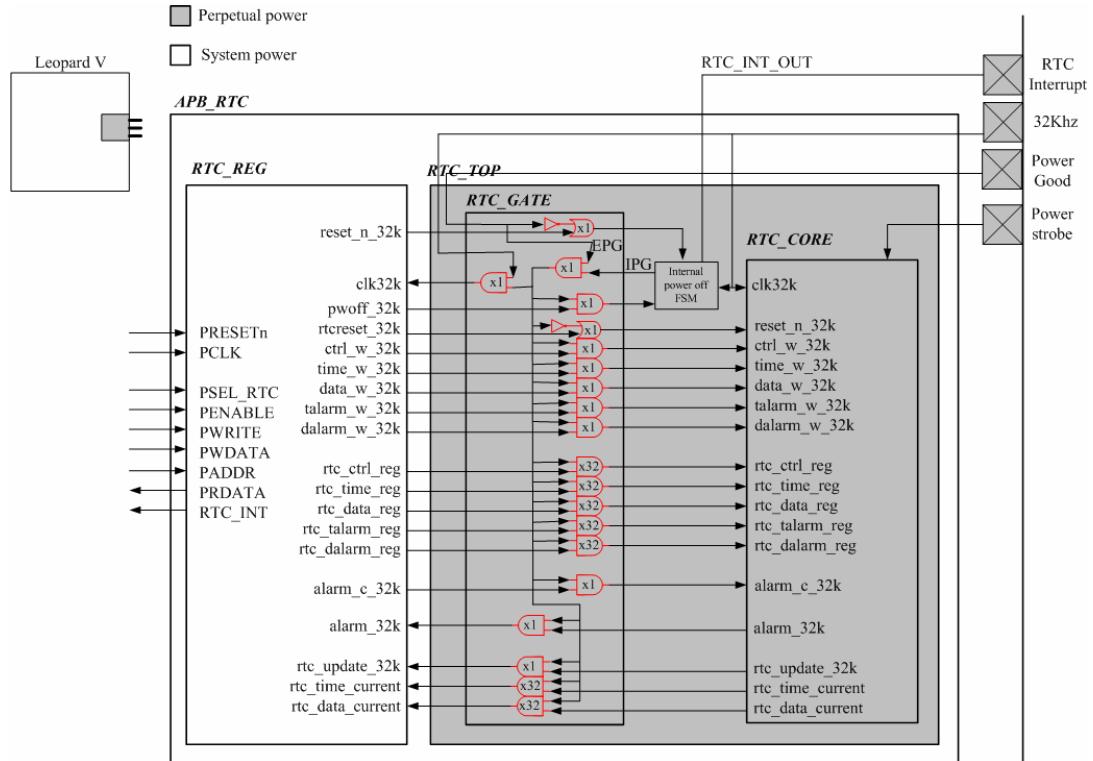
Key Features

- 24 hour time mode with highest precision of one sixteenth of a second
- Calendar function with correction for leap year
- Programmable alarm with interrupt generation
- Mask-able interrupt
- System power off sequence with output control pin
- Programmable alarm wake up system power with output control pin
- Independent RTC reset signal prevent RTC reset during power on/off

Architecture

This section provides a description about the functions and behavior under various conditions.

Block Diagram



Block Descriptions

RTC_REG

RTC registers

The RTC has several software accessible registers. Some of these registers reflect values of BCD time and date counters. Control registers control the operation of the RTC counters and the divider.

Alarm registers

The RTC has several software accessible registers. Some of these registers contain alarm values that set a bit and optionally trigger an alarm interrupt if RTC time or date match the alarm values. Triggering of an alarm interrupt as well as which alarm values trigger an alarm are programmable in the control register.

RTC_CORE

Clock divider

Clock divider divides the external 32KHz clock source to 1Hz clock period.

Clock counter

The RTC clock counters count up to tenth century units in BCD format and with one sixteenth of a second precision.

RTC_GATE

Use AND/OR gates to separate system power domain signals and perpetual power domain signals in RTC core boundary to prevent signal floating when system power off.

Internal power off FSM

A power off/on sequence state machine triggered by software when system into power off mode. The sequence can perform RTC core boundary signal gating and generate an output RTC_INT_OUT signal which controls external system power source.

Registers

This section describes the control/status registers of the design.

Registers Summary

Name	Offset	Size	Reset Value	Description
RTC_TIME	0x0000	W	0x01000000	RTC time register.
RTC_DATE	0x0004	W	0x00000041	RTC date register.
RTC_TALRM	0x0008	W	0x00000000	RTC time alarm register.
RTC_DALRM	0x000C	W	0x00000000	RTC date alarm register.
RTC_CTRL	0x0010	W	0x00000000	RTC control register.
RTC_RESET	0x0014	W	0x00000000	RTC software reset register.
RTC_PWOFF	0x0018	W	0x00000000	System power off register.
RTC_RTCPWFAIL	0x001C	W	0x00000000	RTC power failed register.

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

RTC_TIME

Address: Operational Base + offset (0x0000)

RTC time register

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26:24	RW	0x1	DOW Day of week, with valid values from 1 (Sunday) to 7 (Saturday).
23:22	RW	0x0	TH Ten hours, with valid values from 0 to 2 when operating in 24-hour mode and 0 to 1 when operating in 12-hour mode.
21:18	RW	0x0	H

			Hours, with valid values from 0 to 9 except when ten hours is 2.
17:15	RW	0x0	TM Ten minutes, with valid values from 0 to 5.
14:11	RW	0x0	M Minutes, with valid values from 0 to 9.
10:8	RW	0x0	TS Ten seconds, with valid values from 0 to 5.
7:4	RW	0x0	S Seconds, with valid values from 0 to 9.
3:0	RW	0x0	SOS One sixteenth of a second, with valid values from 0 to 15.

RTC_DATE

Address: Operational Base + offset (0x0004)

RTC date register

Bit	Attr	Reset Value	Description
31:24	-	-	Reserved.
26:23	RW	0x0	TC Ten centuries, with valid values from 0 to 9.
22:19	RW	0x0	C Centuries, with valid values from 0 to 9.
18:15	RW	0x0	TY Ten years, with valid values from 0 to 9.
14:11	RW	0x0	Y Years, with valid values from 0 to 9.
10	RW	0x0	TM Ten months, with valid values from 0 to 1.
9:6	RW	0x1	M Months, with valid values from 0 to 9 except when ten months is 1.
5:4	RW	0x0	TD Ten days, with valid values from 0 to 3 except for February.
3:0	RW	0x1	D Days, with valid values from 0 to 9 except when ten days is 2 or 3.

RTC_TALRM

Address: Operational Base + offset (0x0008)

RTC time alarm register

Bit	Attr	Reset Value	Description
31	RW	0x0	CDOW When set, day of week is compared to day of week alarm value.

30	RW	0x0	CH When set, hour is compared to hour alarm value.
29	RW	0x0	CM When set, minute is compared to minute alarm value.
28	RW	0x0	CS When set, second is compared to second alarm value.
27	RW	0x0	CSOS When set, sixteen of second is compared to sixteen of second alarm value.
26:24	RW	0x0	DOW Day of week, with valid values from 1 (Sunday) to 7 (Saturday).
23:22	RW	0x0	TH Ten hours, with valid values from 0 to 2.
21:18	RW	0x0	H Hours, with valid values from 0 to 9 except when ten hours is 2.
17:15	RW	0x0	TM Ten minutes, with valid values from 0 to 5.
14:11	RW	0x0	M Minutes, with valid values from 0 to 9.
10:8	RW	0x0	TS Ten seconds, with valid values from 0 to 5.
7:4	RW	0x0	S Seconds, with valid values from 0 to 9.
3:0	RW	0x0	SOS One sixteenth of a second precision, with valid values from 0 to 15.

RTC_DALRM

Address: Operational Base + offset (0x000C)

RTC date alarm register

Bit	Attr	Reset Value	Description
31	-	-	Reserved.
30	RW	0x0	CC When set, century is compared to century alarm value.
29	RW	0x0	CY When set, year is compared to year alarm value.
28	RW	0x0	CM When set, month is compared to month alarm value.
27	RW	0x0	CD When set, day is compared to day alarm value.
26:23	RW	0x0	TC Ten centuries, with valid values from 0 to 9.
22:19	RW	0x0	C

			Centuries, with valid values from 0 to 9.
18:15	RW	0x0	TY Ten years, with valid values from 0 to 9.
14:11	RW	0x0	Y Years, with valid values from 0 to 9.
10	RW	0x0	TM Ten months, with valid values from 0 to 1.
9:6	RW	0x0	M Months, with valid values from 0 to 9 except when ten months is 1.
5:4	RW	0x0	TD Ten days, with valid values from 0 to 3 except for February.
3:0	RW	0x0	D Days, with valid values from 0 to 9 except when ten days field is 2 or 3.

RTC_CTRL

Address: Operational Base + offset (0x0010)

RTC control register

Bit	Attr	Reset Value	Description
31	RW	0x0	EN When set, RTC is operating normally. When cleared, RTC operation is halted.
30	RW	0x0	ALRM This bit represents alarm status. When it is set, an alarm is pending. The software must write zero to this bit in order to clear it.
29	RW	0x0	ALRM_PWON Alarm power on enable. When set, RTC power off/on sequence will wait the alarm to power on the system when system is in power off mode.
28	RW	0x0	RTC_INT_OUT_EN RTC_INT_OUT pin enable.
27	RW	0x0	SOS bypass.
26:0	RW	0x0	DIV Divide factor for dividing real-time clock source down to sixteen of a second. For example, If the frequency of RTC clock is 32.768KHz, this value should be set to 0x7FF (0x800-1).

RTC_RESET

Address: Operational Base + offset (0x0014)

RTC control reset register

Bit	Attr	Reset Value	Description

1	W	0x0	RTCRESET Software reset of RTC counter. Write this bit to high will reset RTC counter.
0	W	0x0	CRESET Software reset of RTC control circuit. Write this bit to high will reset RTC control circuit.

RTC_PWOFF

Address: Operational Base + offset (0x0018)

System power off register

Bit	Attr	Reset Value	Description
0	W	0x0	PWOFF Write this bit to high to trigger the RTC power off sequence.
0	R	0x0	Good to write RTC control register indicator. '1' : No gating between perpetual power signals and system power signals. '0' : Gated between perpetual power signals and system power signals.

RTC_RTCPWFAIL

Address: Operational Base + offset (0x001C)

RTC counter reset register

Bit	Attr	Reset Value	Description
0	R	0x0	RTCPWFAIL RTC power fail indicator, the RTC power fail indicator will be loaded into bit0 every time system power on. Bit0 goes high indicate RTC power has been lost before.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

Initialization/Changing time and date

User has one sixteenth of a second to initialize/change all the time/date counters after the first write to any of the time/date counters while the RTC is enabled. This guarantees that the sixteenth of a second counter roll over does not occur and thus potentially roll over other counters.

Time and Date counters operation

Time and date counters count time and date in BCD format. Time counters use 24-hour format and date counters support leap year calculation.

For normal operation time and date counters must be enabled by setting bit RTC_CTRL[EN]. They operate with a precision of sixteenth of a second.

Time and date counters run continuously during system power off and those counters can only be reset by software reset SRESET.

Alarm

An Alarm can be set with any combination of century, year, month, day, day of week, hour, minute, second or tenth of a second. To set an alarm, write the desired values into appropriate alarm registers and enable comparison of desired groups of bits in alarm registers.

When the time and date counters match all enabled alarm groups of bits, RTC_CTRL[ALRM] bit is asserted. The software must later clear this bit in order to allow detection of future alarms.

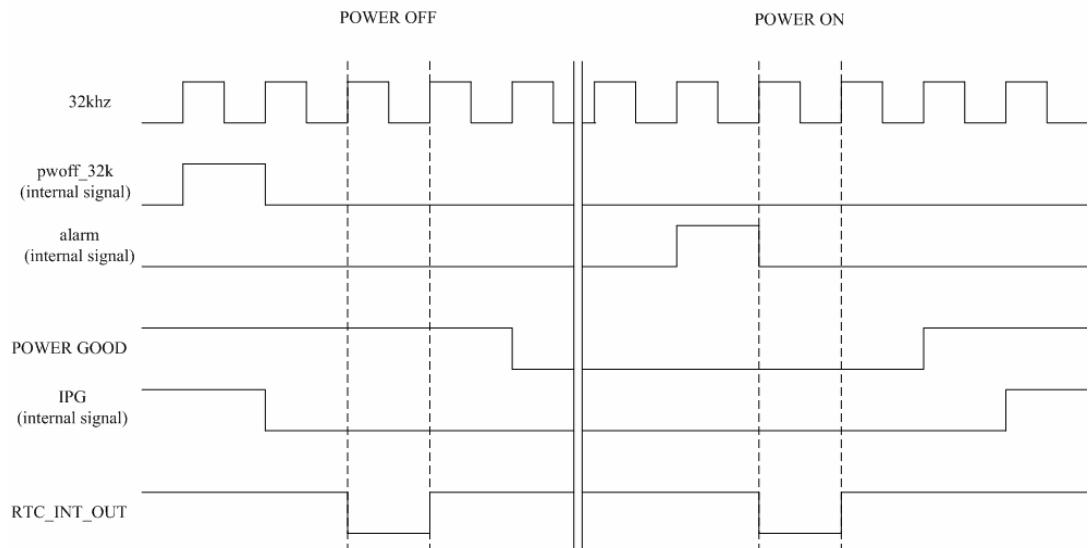
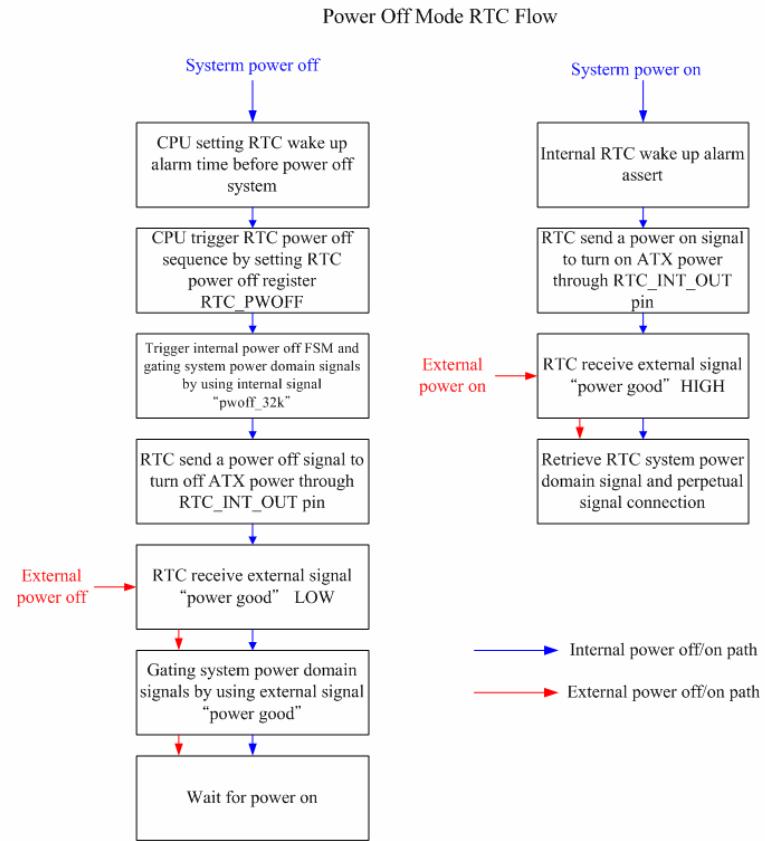
The alarm can also be used to wake up the system when system is in power off mode, by setting RTC_CTRL[ALRM_PWON] and RTC_CTRL[RTC_INT_OUT_EN] to high in advance, the RTC power on sequence will wait for the alarm be asserted and sent an signal to power on external system power source through RTC_INT_OUT pin.

Alarm interrupt

If RTC_CTRL[ALRM] bit is asserted, an alarm interrupt is generated once alarm is triggered. Software should clear the RTC_CTRL[ALRM] bit in order to clear the pending alarm interrupt.

System power off/on

If RTC_PWOFF[PWOFF] bit is set, the internal power off sequence will be triggered. The boundary signals between system power domain and RTC perpetual power domain will be gated by internal signal IPG first to prevent power leakage and the RTC will then generate a negative pulse through RTC_INT_OUT to turn off the external system power source. When system power is turned off, the battery power will supply the RTC counter. Also, the RTC can turn on the system power by sending a negative pulse through RTC_INT_OUT to turn on the external system power source when RTC alarm assert. If the system power is off or on manually, the RTC core will gate the different power domain signal by external power good signal to prevent power leakage. The Flow and waveform of power off/on is show below.



Programming sequence

Software reset sequence

1. Write CRESET bit0 to '1' to reset RTC control and RTCRESET bit0 to '1' to reset RTC counter the first time initial RTC.
2. Read register RTC_PWOFF bit0 to check the status of the RTC signal gating, if this bit is high, then access the RTC register directly. If this bit is low, then program the software reset CRESET to reset the RTC control since the power off sequence is not process normally.
3. Configure the clock divider and enable the RTC counter by setting RTC_CTRL.
4. Adjust the time/date or setting alarm by programming RTC_TIME, RTC_DATE, RTC_TALRM and RTC_DALRM.

Power off/on system by using RTC

1. Set wake up alarm time by programming RTC_TALRM, RTC_DALRM and set RTC_CTRL[ALRM_PWON] to '1' if using internal alarm wake up mechanism.
2. Enable RTC_INT_OUT pin by setting RTC_CTRL[RTC_INT_OUT_EN] to '1'.
3. Set RTC_PWOFF bit0 to '1' to trigger RTC power off sequence.
4. Power on sequence will be trigger by RTC alarm if enable alarm wake up mechanism or it will be power on manually.

Chapter 21 SPI Controller

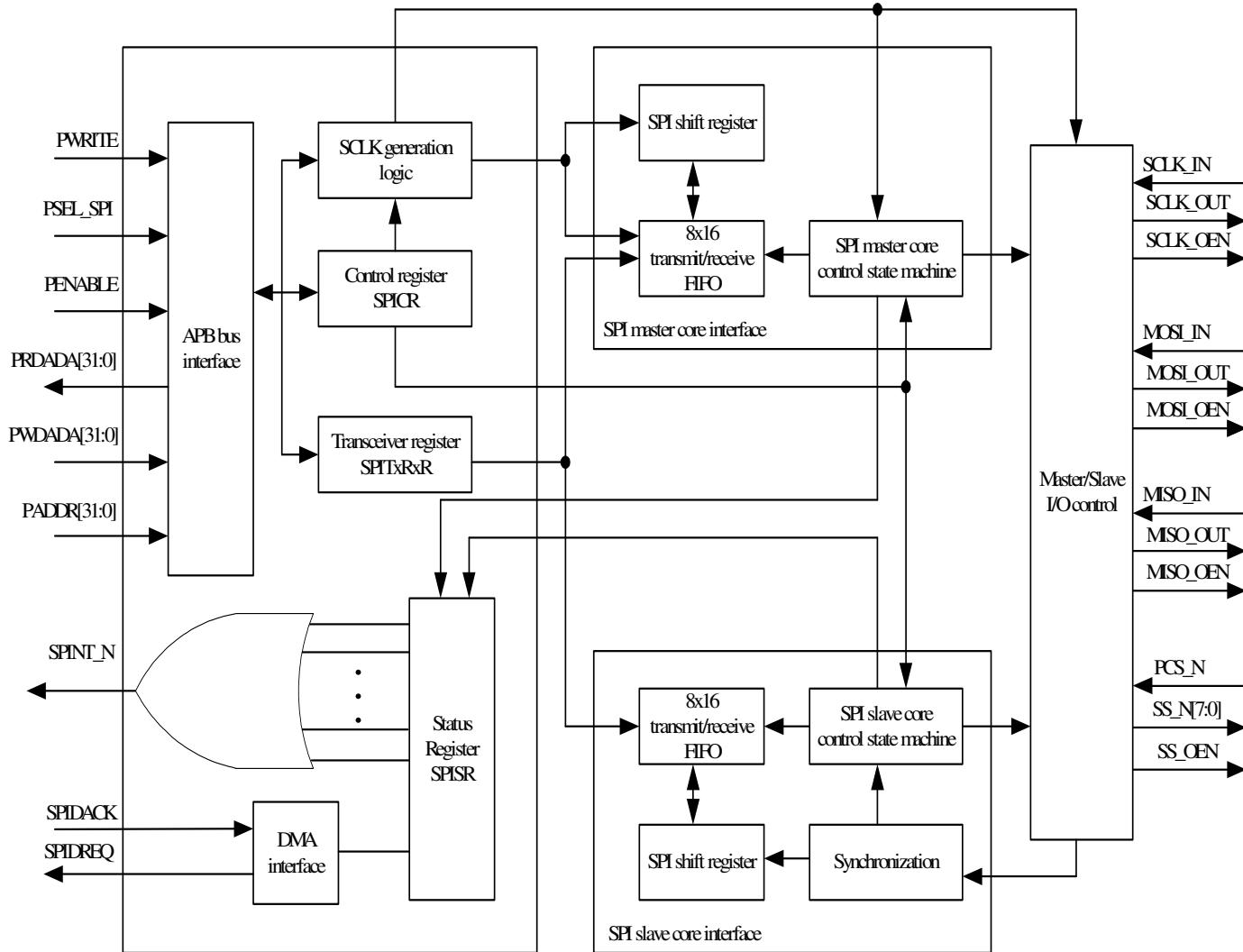
Overview

The Serial Peripheral Interface (SPI) master/slave controller is a full-duplex, synchronous, serial data link that is standard across many microprocessors, microcontrollers, and peripherals. It enables communication between microprocessors and peripherals and/or inter-processor communication. The SPI master/slave controller system is flexible enough to interface directly with numerous commercially available peripherals. The SPI master/slave controller is compatible with above-mentioned protocols as SPI bus master or slave. At the host side, the core acts like an APB compliant slave device.

Key Features

- AMBA APB slave interface
- Support master or slave mode
- DMA Interface
- Four transfer protocols available with selectable clock polarity and clock phase
- Different bit rates available for SCLK
- Full duplex synchronous serial data transfer
- Bi-direction mode
- Support 2 slave devices
- MSB or LSB first data transfer

Architecture Block Diagram



Note : SPINT_N is ORed by RXEIF, TXEIF, SPIORIF, SPICIF, SSNIF, RXAVIF, TXFEIF, RXRECOIF, RXREGUIF, RXREG2IF, RXREG3IF and CHARLNIF

Block Descriptions

The SPI master/slave controller consist of an APB Slave interface for control and

transceiver register setting, a SPI master controller interface that generate SPI signals for peripherals and a SPI slave controller interface that include synchronizer to sync sclk clock. The transmit/receive FIFO used to buffer the data between APB slave interface and SPI master/slave controller.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
SPI_TxR	0x0000	W	0x00000000	SPI master/slave controller transmit FIFO input
SPI_RxR	0x0000	W	0x00000000	SPI master/slave controller receiver FIFO output
SPI_IER	0x0004	W	0x00000000	Enable/Mask interrupts generated by the SPI master controller
SPI_FCR	0x0008	W	0x00000000	SPI master/slave controller FIFO control register
SPI_FWCR	0x000C	W	0x00000100	SPI master/slave controller transaction flow control register
SPI_DLYCR	0x0010	W	0x00000000	SPI master controller delay control register (Master only)
SPI_TxCR	0x0014	W	0x00000000	Transmit counter (Master only)
SPI_RxCR	0x0018	W	0x00000000	Receive counter (Master only)
SPI_SSCR	0x001C	W	0x00000000	SPI master/slave controller slave select and characteristic register
SPI_ISR	0x0020	W	0x00000000	SPI master/slave controller interrupt status register
SPI_FIFO_STAT	0x0024	W	0x00000011	SPI Transmit FIFO and Receive FIFO status
SPI_TX_REG0	0x0028	W	0x00000000	SPI transmit register 0 for normal usage
SPI_TX_REG1	0x002C	W	0x00000000	SPI transmit register 1 for normal usage
SPI_TX_REG2	0x0030	W	0x00000000	SPI transmit register 2 for normal usage
SPI_TX_REG3	0x0034	W	0x00000000	SPI transmit register 3 for normal usage
SPI_RX_REG0	0x0038	W	0x00000000	SPI receive register 0 for normal usage
SPI_RX_REG1	0x003C	W	0x00000000	SPI receive register 1 for normal usage
SPI_RX_REG2	0x0040	W	0x00000000	SPI receive register 2 for normal usage
SPI_RX_REG3	0x0044	W	0x00000000	SPI receive register 3 for normal usage

			usage
--	--	--	-------

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

SPI_TxR

Address: Operational Base + offset (0x0000)

This register contains data to be transmitted on the SPI master/slave controller bus on the MOSI pin.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	W	0x0	SPI master controller transmit data FIFO

SPI_RxR

Address: Operational Base + offset (0x0000)

This register contains the data received from the SPI master/slave controller bus on the MISO pin.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	R	0x0	SPI master controller receive data FIFO

SPI_IER

Address: Operational Base + offset (0x0004)

This register contains the bits to control the interrupt generation of this SPI master/slave controller.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	RW	0x0	Character length mismatch interrupt enable bit. (CHARLNIEN) (Slave only) The bit enables character length mismatch interrupt. When finishing transfer, the received data which character length mismatch the setting of SPI_SSCR[14:11]. “1” enable. “0” disable.
10	RW	0x0	Rx register 3 data available interrupt enable bit. (RXREG3IEN) (Slave only) The bit enables Rx register 3 data available interrupt when one data available in Rx register 3. “1” enable. “0” disable.
9	RW	0x0	Rx register 2 data available interrupt enable bit. (RXREG2IEN) (Slave only) The bit enables Rx register 2 data available interrupt when one data available in Rx register 2.

			“1” enable. “0” disable.
8	RW	0x0	Rx register 1 data available interrupt enable bit. (RXREG1IEN) (Slave only) The bit enables Rx register 1 data available interrupt when one data available in Rx register 1. “1” enable. “0” disable.
7	RW	0x0	Rx register 0 data available interrupt enable bit. (RXREG0IEN) (Slave only) The bit enables Rx register 0 data available interrupt when one data available in Rx register 0. “1” enable. “0” disable.
6	RW	0x0	Transmit FIFO empty interrupt enable bit. (TXFEIEN) The bit enables transmit FIFO interrupt when transmit FIFO is empty. “1” enable. “0” disable.
5	RW	0x0	Slave select signal rising detection enable bit. (SSNRIEN) (Slave only) The bit enables slave select signal rising detection interrupt when the master set slave select bit from “0” to “1”. “1” enable. “0” disable.
4	RW	0x0	Receive data available enable bit. (RXAVIEN) The bit enables receive FIFO interrupt when at least one data available in receive FIFO. “1” enable. “0” disable.
3	RW	0x0	Transmit FIFO interrupt enable bit (TxFIEN). This bit enables transmit FIFO interrupt when transmit FIFO trigger level is reached. “1” enable. “0” disable.
2	RW	0x0	Receive FIFO interrupt enable bit (RxFIEN). This bit enables receive FIFO interrupt when receive FIFO trigger level is reached. “1” enable. “0” disable.
1	RW	0x0	Receive FIFO overrun interrupt enable bit (RxFOIEN). This bit enables receive FIFO overrun interrupt when receive FIFO overrun condition is occurred. “1” enable. “0” disable.

0	RW	0x0	Receive transfer complete interrupt enable bit (RxCIEN). (Master only) This bit enables receive transfer complete interrupt each time a receive transaction is ended. “1” enable. “0” disable.
---	----	-----	--

SPI_FCR

Address: Operational Base + offset (0x0008)

The FCR allows selection of the FIFO trigger level (the number of entries in receive FIFO required to enable the receive FIFO interrupt and the number of empty entries in the transmit FIFO required to enable the transmit FIFO interrupt). In addition, the FIFOs can be cleared using this register.

Bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13:11	RW	0x0	Define the receive FIFO interrupt trigger level. The receive FIFO interrupt trigger level meaning is described below. For example trigger level 4 entries is for indication that at least there has 4 entries data available in receive FIFO. 0x0 – 2 entries 0x1 – 4 entries 0x2 – 6 entries 0x3~0x7 – Reserved.
10:8	RW	0x0	Define the transmit FIFO interrupt trigger level. The transmit FIFO interrupt trigger level meaning is described below. For example trigger level 4 entries is for indication that at least there has 4 entries empty location for pushing data into transmit FIFO. 0x0 – 2 entries 0x1 – 4 entries 0x2 – 6 entries 0x3~0x7 – Reserved.
7:4	-	-	Reserved.
3	W	0x0	SPI master/slave controller receive FIFO reset bit (CLRRXF_N). Writing a ‘1’ to this bit will reset receive FIFO.
2	W	0x0	SPI master/slave controller transmit FIFO reset bit (CLRTXF_N). Writing a ‘1’ to this bit will reset transmit FIFO.
1	R	0x0	Transmit FIFO full flag (TxFF). This bit is set whenever transmit FIFO is full.
0	R	0x0	Receive data available flag (RxDAF). This bit is set whenever at least has one data entry available in receive FIFO.

Notes:

The transmit and receive FIFO reset signal would sustain **3 pclk cycle**, during the reset

period any access to FIFO would be ignored.

SPI_FWCR

Address: Operational Base + offset (0x000C)

This register is used to control SPI master/slave controller transaction flow.

Bit	Attr	Reset Value	Description
31:15	-	-	Reserved.
14	RW	0x0	Slave select signal control enable bit (SSC). (Master only) 0: slave select signal is controlled by HW 1: slave select signal is controlled by SW
13	RW	0x1	Slave select signal active bit (SSA). (Master R/W, and Slave read only) Writing a '1' to this bit will let slave select signal to be not active. The value of slave select signal will turn from '0' into '1'. Writing a '0' to this bit will let slave select signal to be active. The value of slave select signal will turn from '1' into '0'.
12	RW	0x0	SPI master/slave mode (SPIMD). This bit selects SPI master/slave mode. "0" SPI is in Slave mode "1" SPI is in Master mode
11	W	0x0	SPI master/slave controller soft reset bit (SRST_N). Writing a '1' to this bit will reset the SPI master/slave controller logic.
10	RW	0x0	SPI master/slave enable bit (SPIEN). This bit enables the SPI master controller. "1" enables the SPI master/slave "0" disables the SPI master/slave
9	RW	0x0	SPI run bit (SPIRUN). (Master only) When the CPU set this bit from "0" to "1", the SPI master/slave controller begins to transfer the data stored in the transmit FIFO and/or receive the data into receive FIFO on the SPI master/slave controller bus. And it will be cleared to "0" after the transaction is ended automatically.
8	RW	0x1	SPI master controller clock idle enable bit (CKIDLEN). (Master only) This bit determines whether the SPI master controller clock could be asserted in an idle state or not during a transaction process if the master core meet the receive FIFO full or transmit FIFO empty condition. "1" SPI master controller clock could be asserted in an idle state. "0" SPI master controller clock could not be asserted in an idle state.
7	-	-	Reserved.

6	RW	0x0	Hardware DMA request enable
5	RW	0x0	Transmit and receive simultaneously transfer enable (TxRxsten). (Master only) This bit is used to indicate whether the transmit and receive transfer would concurrently happen during a transaction. “0” Tx and Rx would not concurrently happen. “1” Tx and Rx would concurrently happen.
4	RW	0x0	SPI master/slave controller clock polarity bit (CPOL). This bit determines the polarity of SCLK. “0” SCLK is low when idle. “1” SCLK is high when idle.
3	RW	0x0	Clock Phase Bit (CPHA). This bit determines the clock phase of SCLK in relationship to the serial data. “0” data is valid on first SCLK edge (rising or falling) after slave select has asserted. “1” data is valid on second SCLK edge (rising or falling) after slave select has asserted.
2	RW	0x0	LSB-First Enable (LSBEN). “1” LSB first transfer. “0” MSB first transfer.
1	-	-	Reserved.
0	RW	0x0	Loop back mode (LBKMD). (Master only) This bit is to indicate the operation mode of the SPI master controller is in a normal operation mode or in a loop back mode. ‘0’ Normal operation mode. ‘1’ Loop back mode.

Notes:

The soft reset signal would sustain **3 pclk cycles**, during the reset period any access to FIFO would be ignored.

SPI_DLYCR

Address: Operational Base + offset (0x0010)

This register is used to set the necessary clock delay during a transaction according to the slave device specification requirement.

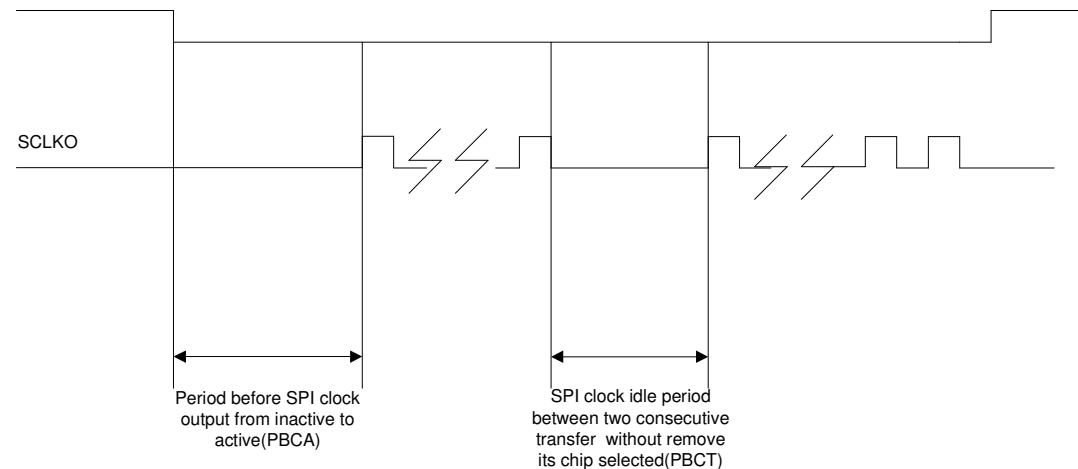
Bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	RW	0x0	Period between Tx and Rx transfer (PBTxRx). This field defines the delay between transmit transfer complete and receive transfer start. 0x0: Non SPI master controller clock delay. 0x1: 4 SPI master controller clock delay. 0x2: 8 SPI master controller clock delay. 0x3: 16 SPI master controller clock delay. 0x4: 32 SPI master controller clock delay.

			0x5: 64SPI master controller clock delay. 0x6: 128 SPI master controller clock delay. 0x7: 256 SPI master controller clock delay.
7:6	-	-	Reserved.
5:3	RW	0x0	Period between two consecutive transfers (PBCT). This field defines the delay between consecutive transfers to the device without removing its chip select. 0x0: Non SPI master controller clock delay. 0x1: 4 SPI master controller clock delay. 0x2: 8 SPI master controller clock delay. 0x3: 16 SPI master controller clock delay. 0x4: 32 SPI master controller clock delay. 0x5: 64 SPI master controller clock delay. 0x6: 128 SPI master controller clock delay. 0x7: 256 SPI master controller clock delay.
2:0	RW	0x0	Period before SPI master controller clock active (PBCA). This field defines the delay before SPI master controller clock is from idle to active after the chip select is asserted. 0x0: 1/2 SPI master controller clock delay. 0x1: 4 SPI master controller clock delay. 0x2: 8 SPI master controller clock delay. 0x3: 16 SPI master controller clock delay. 0x4: 32 SPI master controller clock delay. 0x5: 64SPI master controller clock delay. 0x6: 128 SPI master controller clock delay. 0x7: 256 SPI master controller clock delay.

Notes:

The delay meaning will be illustrated by timing diagram below

Chip select



SPI_TxCR

Address: Operational Base + offset (0x0014)

This register controls the total transfer data size in each transmit transaction

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	0: Stop the transmit transaction. 1-65535: Start to a transmit transaction.

SPI_RxCR

Address: Operational Base + offset (0x0018)

This register controls the total transfer data size in each receive transaction.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	0: Stop the receive transaction. 1-65535: Start to a receive transaction.

SPI_SSCR

Address: Operational Base + offset (0x001C)

SPI master/slave controller slaves select and characteristic control register.

Bit	Attr	Reset Value	Description
31:15	-	-	Reserved.
14:11	RW	0x0	Character length determines bits (SPICHRL). This field specifies how many bits would be transferred in each transfer. 0x0: Reserved 0x1: Reserved 0x2: Reserved 0x3: 4 bits (Master only) 0x4: 5 bits (Master only) 0x5: 6 bits (Master only) 0x6: 7 bits (Master only) 0x7: 8bits 0x8: 9 bits 0x9: 10 bits 0xA: 11 bits 0xB: 12 bits 0xC: 13 bits 0xD: 14 bits 0xE: 15 bits 0xF: 16bits
10:8	RW	0x0	SPI master controller slave select register (SPISSR). (Master only) 0x0: Slave0 device (SPIx_SS[0]) is set to an active state. 0x1: Slave0 device (SPIx_SS[1]) is set to an active state.
7:6	-	-	Reserved.
5:0	RW	0x0	SPI master controller clock divisor bits (SPIDIVR). (Master only) The value of SPIDIVR is used to generate the transmit and

			receive bit rate of this SPI master controller. And the bit rate equation will be described more detail at below section.
--	--	--	---

Notes:

CPHA, CPOL, should be set to match the protocol expected by the SPI slave device.

SPI_ISR

Address: Operational Base + offset (0x0020)

SPI master/slave controller interrupt status register.

Bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	RW	0x0	Character length mismatch interrupt flag. (CHARLNIF) (Slave only) While finishing transfer, the received data which character length mismatch the setting of SPI_SSCR[14:11], the bit is set. An interrupt will be asserted to the CPU when both this bit is set and CHARLNEN bit is enabled. The bit is cleared while writing “1” into the register.
10	RW	0x0	Rx register 3 data available interrupt flag. (RXREG3IF) (Slave only) This bit is set when data available in Rx register 3. An interrupt will be asserted to the CPU when both this bit is set and RXREG3IEN bit is enabled. The bit is cleared while writing “1” into the register.
9	RW	0x0	Rx register 2 data available interrupt flag. (RXREG2IF) (Slave only) This bit is set when data available in Rx register 2. An interrupt will be asserted to the CPU when both this bit is set and RXREG2IEN bit is enabled. The bit is cleared while writing “1” into the register.
8	RW	0x0	Rx register 1 data available interrupt flag. (RXREG1IF) (Slave only) This bit is set when data available in Rx register 1. An interrupt will be asserted to the CPU when both this bit is set and RXREG1IEN bit is enabled. The bit is cleared while writing “1” into the register.
7	RW	0x0	Rx register 0 data available interrupt flag. (RXREG0IF) (Slave only) This bit is set when data available in Rx register 0. An interrupt will be asserted to the CPU when both this bit is set and RXREG0IEN bit is enabled. The bit is cleared while writing “1” into the register.
6	R	0x0	Transmit FIFO empty interrupt flag. (TXFEIF) This bit is set when transmit FIFO is empty. An interrupt will be asserted to the CPU when both this bit is set and

			TXFEIEN bit is enabled. This bit will be cleared when transmit FIFO is not empty.
5	RW	0x0	Slave select signal rising interrupt flag. (SSNRIF) (Slave only) This bit is set when slave select signal rising. An interrupt will be asserted to the CPU when both this bit is set and SSNRIEN bit is enabled. The bit is cleared while writing “1” into the register.
4	R	0x0	Receive data available interrupt flag. (RXAVIF) This bit is set when at least one data available in receive FIFO. An interrupt will be asserted to the CPU when both this bit is set and RXAVIEN bit is enabled. This bit will be cleared when receive FIFO is empty.
3	R	0x0	Transmit FIFO interrupt flag (TxFIF). This bit is set when the transmit FIFO trigger level is reached, and CPU wishes to keep transmit data to the device. An interrupt will be asserted to the CPU when both this bit is set and TxFIEN bit is enabled. This bit will be cleared when transmit FIFO pointer drops below trigger level
2	R	0x0	Receive FIFO interrupt flag (RxFIF). This bit is set whenever the receive FIFO trigger level is reached. An interrupt will be asserted to the CPU when both this bit is set and RxFIEN bit is enable. This bit will be cleared when receive FIFO pointer drops below trigger level
1	R	0x0	SPI master/slave controller overrun interrupt flag (SPIORIF). ‘1’ – If the receive FIFO is full and another character has been received in the receiver shift register. If another character is starting to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. The bit is cleared upon the receive FIFO is cleared by software simultaneously. ‘0’ – No overrun state an another SPI master controller transaction
0	R	0x0	Receive complete interrupt flag (RxCIF). (Master only) This bit is set whenever the receive transaction is over. An interrupt will be asserted to the CPU when both this bit is set and RxCIEN bit is enabled. The bit is cleared upon reading from the register.

SPI_FIFO_STAT

Address: Operational Base + offset (0x0024)

This register contains SPI Transmit FIFO and Receive FIFO status.

Bit	Attr	Reset Value	Description
-----	------	-------------	-------------

31:7	-	-	Reserved.
6	R	0x0	Receive FIFO full bit. (RxFF_FULL) This bit shows Receive FIFO is full or not. “1”: Receive FIFO is full. “0”: Receive FIFO is not full.
5	R	0x0	Receive FIFO half full bit. (RxFF_HALFFULL) This bit shows Receive FIFO is half full or not. “1”: There are four data in Receive FIFO. “0”: There aren’t four data in Receive FIFO.
4	R	0x1	Receive FIFO empty bit. (RxFF_EMPTY) This bit shows Receive FIFO is empty or not. “1”: Receive FIFO is empty. “0”: Receive FIFO is not empty.
3	-	-	Reserved.
2	R	0x0	Transmit FIFO full bit. (TxFF_FULL) This bit shows transmit FIFO is full or not. “1”: Transmit FIFO is full. “0”: Transmit FIFO is not full.
1	R	0x0	Transmit FIFO half full bit. (TxFF_HALFFULL) This bit shows transmit FIFO is half full or not. “1”: There are four data in Transmit FIFO. “0”: There aren’t four data in Transmit FIFO.
0	R	0x1	Transmit FIFO empty bit. (TxFF_EMPTY) This bit shows transmit FIFO is empty or not. “1”: Transmit FIFO is empty. “0”: Transmit FIFO is not empty.

SPI_TX_REGx (x = 0, 1, 2, 3)

Address: Operational Base + offset (0x0028/0x002C/0x0030/0x0034)

This register contains data to be transmitted on the SPI controller bus on the MISO pin.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	RW	0x0	SPI master/slave controller transmit data register. (Slave only)

SPI_RX_REGx (x = 0, 1, 2, 3)

Address: Operational Base + offset (0x0038/0x003C/0x0040/0x0044)

This register contains the data received from the SPI controller bus on the MOSI pin.

Bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:0	R	0x0	SPI master/slave controller receive data register. (Slave only)

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

SPI Slave receives HW command from Master before transferring data.

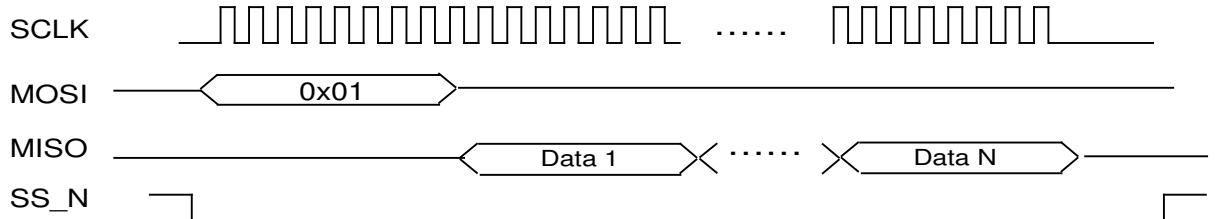
The list of instructions and their operation codes are contained in the following table. All instructions are start with a high-to-low pcs_n transition.

Command	Description
0x01	MISO data from Slave Tx FIFO
0x02	MISO data from Slave Tx/Rx FIFO status
0x03	MISO data from Slave Tx register 0
0x04	MISO data from Slave Tx register 1
0x05	MISO data from Slave Tx register 2
0x06	MISO data from Slave Tx register 3
0x08	MOSI data to Slave Rx FIFO
0x09	MOSI data to Slave Rx register 0
0x0A	MOSI data to Slave Rx register 1
0x0B	MOSI data to Slave Rx register 2
0x0C	MOSI data to Slave Rx register 3

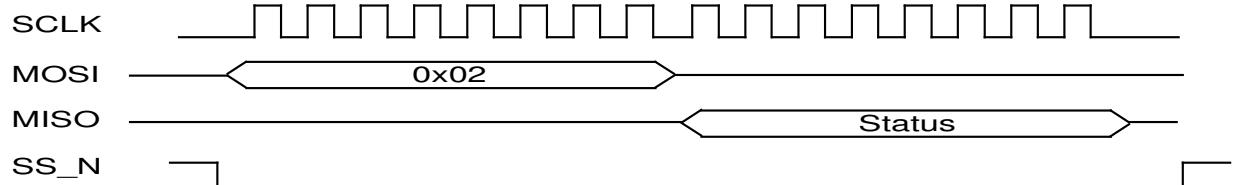
Note: Command length is depend on SPI_SSCR[14:11]. The minimum size is 8 bits. For example, if bit length is 16 bits, the master should transmit 0x0008 to the slave in order to write data to slave Rx FIFO.

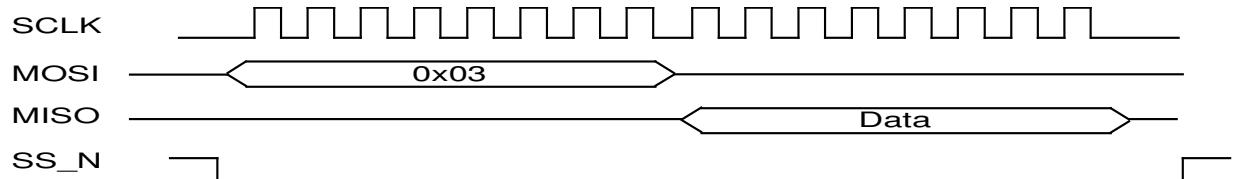
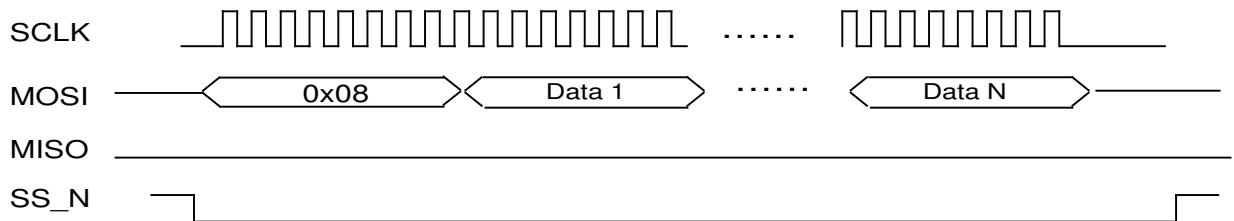
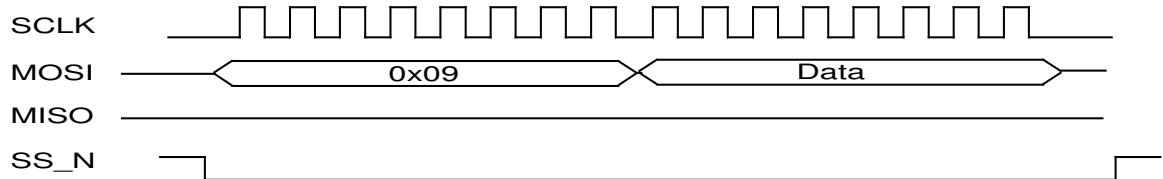
HW command timing

Command 0x01

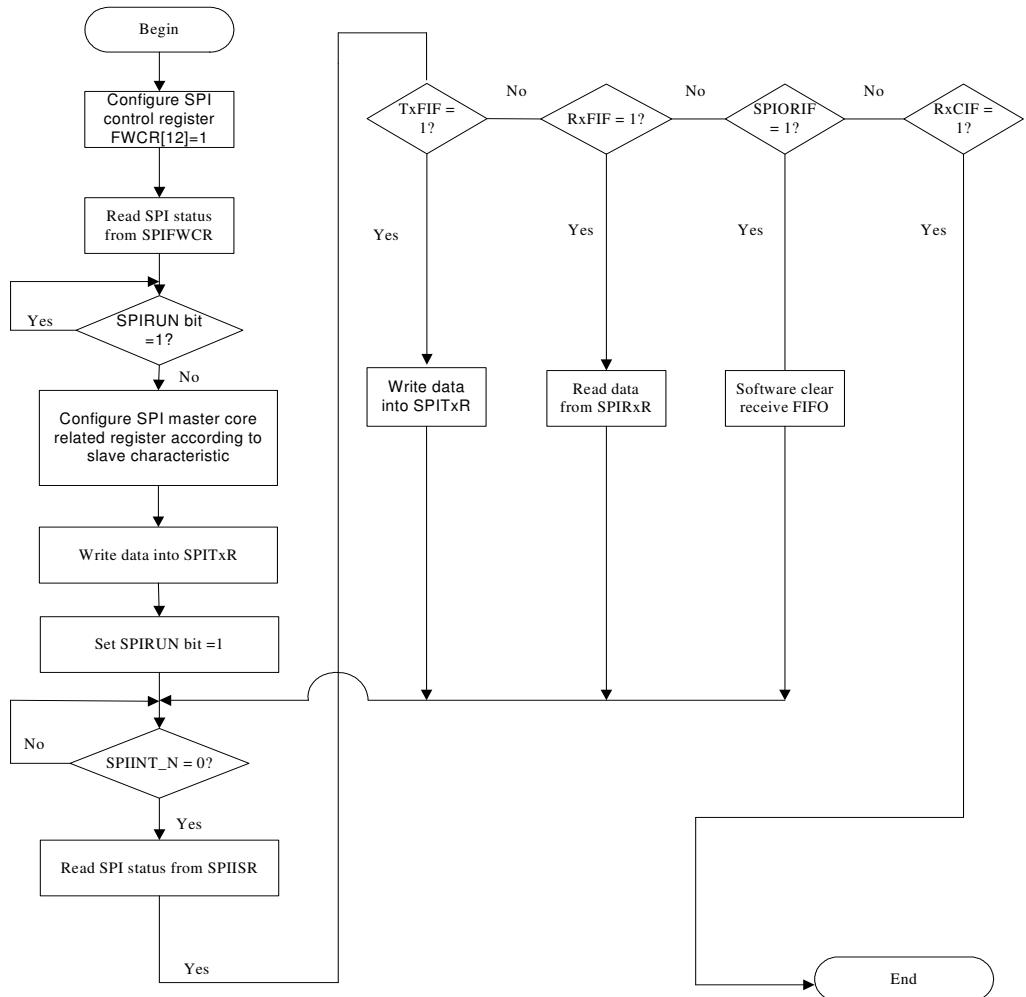


Command 0x02

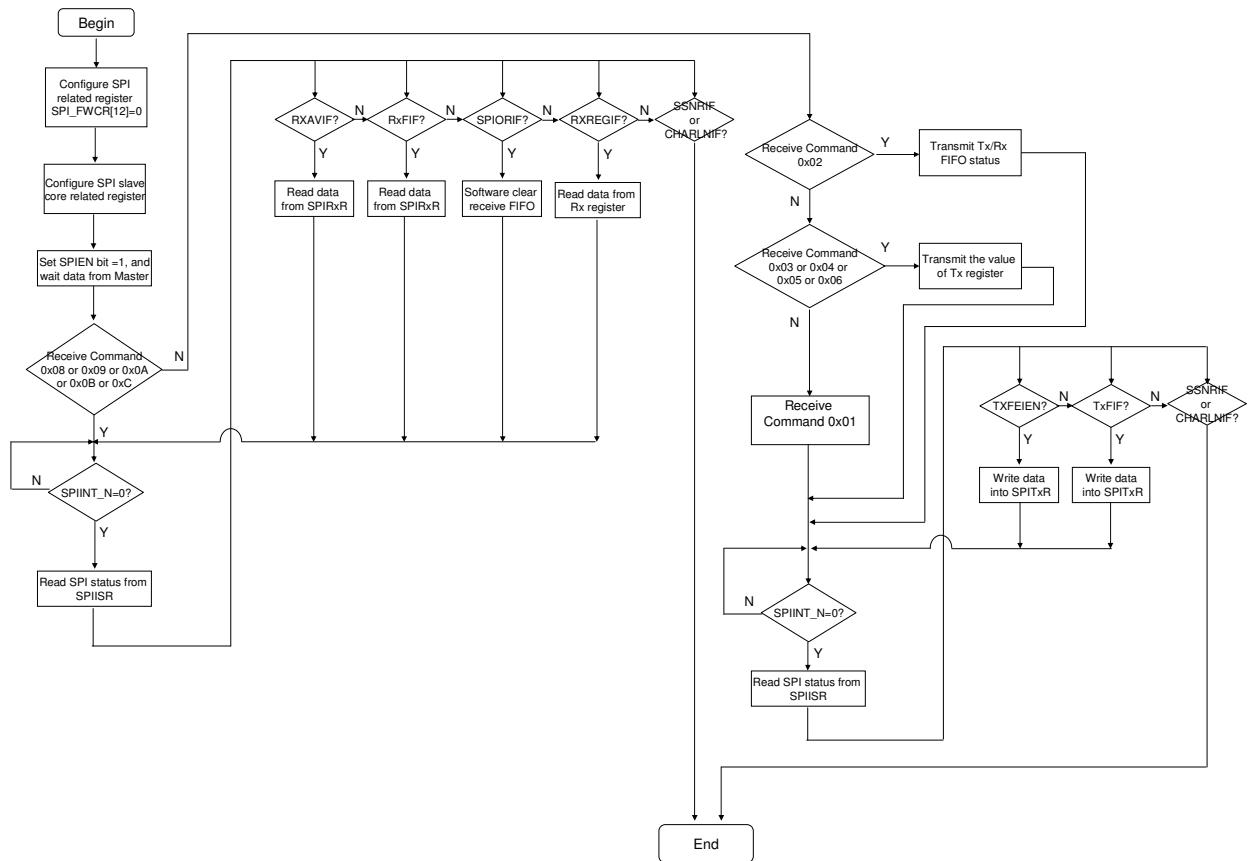


Command 0x03 ~ Command 0x06Command 0x08Command 0x09 ~ Command 0x0CSPI master/slave controller operation flow chart

The flow chart below is to describe how the software to configure and perform a SPI master controller transaction through this SPI master/slave controller.

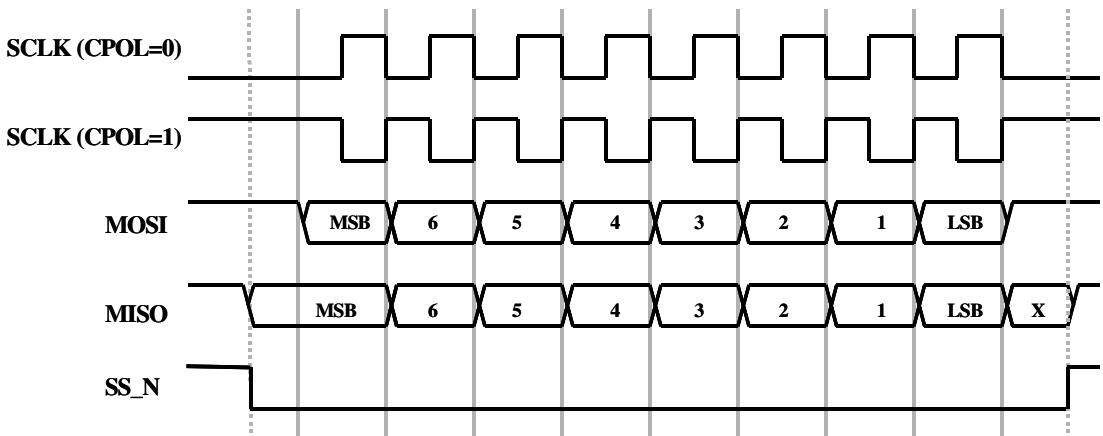


The flow chart below is to describe how the software to configure and perform a SPI slave controller transaction through this SPI master/slave controller.

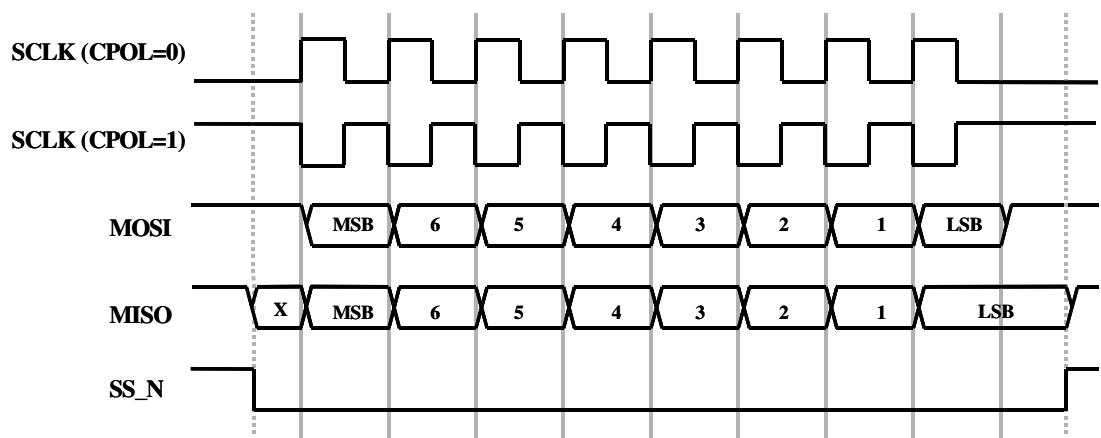


SPI master/slave controller data transfer waveform

CPHA=0 transfer waveform



CPHA=1 transfer waveform



Chapter 22 I2C Controller

Overview

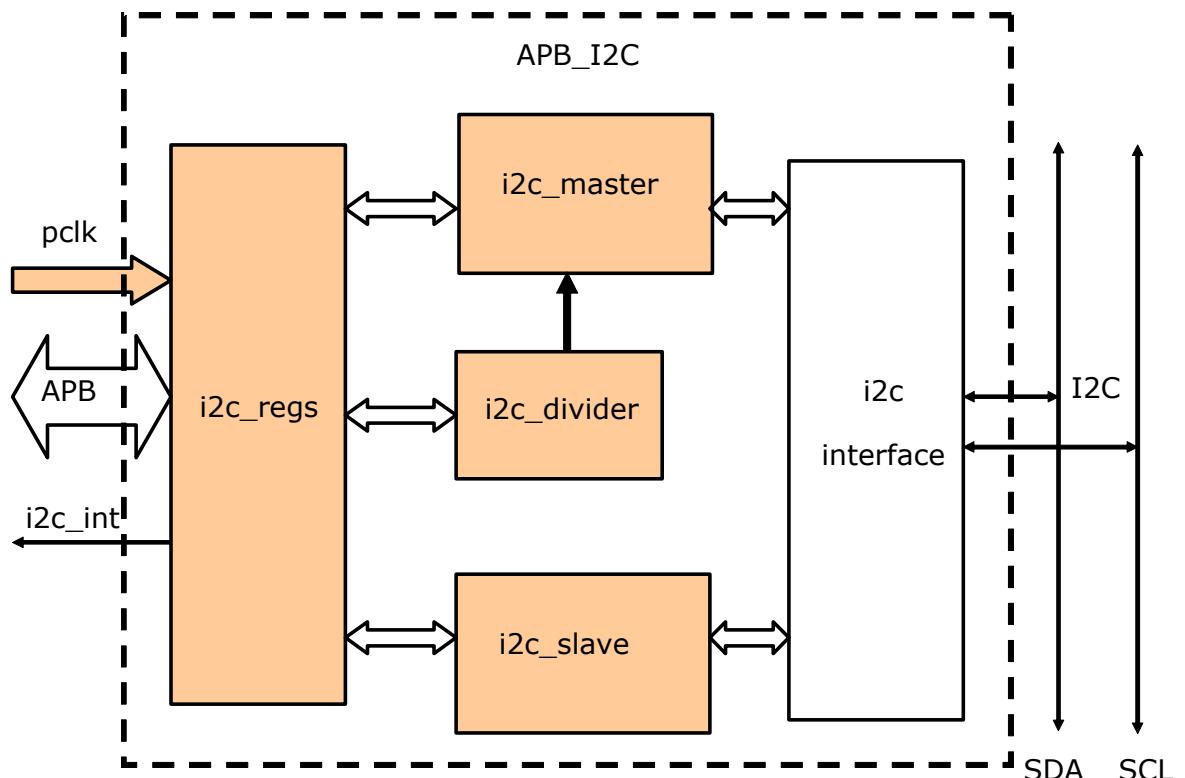
The Inter-Integrated Circuit (I2C) is a two wired (SCL and SDA), bi-directional serial bus that provides an efficient and simple method of information exchange between devices. The I2C bus is a multi-master bus protocol using arbitration to avoid bus collision if two or more masters attempt to control the bus at the same time. This I2C bus controller supports both master and slave modes acting as a bridge between AMBA protocol and generic I2C bus system.

Key Features

- Item Compatible with I2C-bus
- AMBA APB slave interface
- Supports master and slave modes of I2C bus
- Multi masters operation
- Software programmable clock frequency and transfer rate up to 400Kbit/sec
- Supports 7 bits and 10 bits addressing modes
- Interrupt or polling driven byte-by-byte data transfer
- Clock stretching and wait state generation

Architecture

Block Diagram



Block Descriptions

i2c_regs – Control and Status Registers

The i2c_regs component is used to control the I2C controller operation by the host with APB interface. It implements the register set and the interrupt functionality. The CSR component operates synchronously with the pclk clock.

i2c_master – I2C Master Control and State Machine

The I2C master controller implements the I2C master operation for transmit data to and receive data from other I2C devices. The I2C master controller operates synchronously with the pclk.

i2c_slave – I2C Slave Control and State Machine

The I2C slave controller implements the I2C master operation for transmit data to and receive data from other I2C devices. The I2C slave controller operates synchronously with the pclk.

i2c_divider – Clock Divider

The clock divider module generates I2C clock SCL output signals from pclk at frequency according the given equation.

i2c_interface – I2C interface (These are logics under top module. There is actually no such a module)

SDA output enable from I2C master controller and slave controller are ANDed together as the output ports. Similarly, SCL output enable from I2C master controller and slave controller are ANDed together. SDA output and SCL output are actually ties to the ground since I2C is an open drain architecture. So once enabled, SDA/ SCL on I2C will be pulled low.

Registers

This chapter describes the control/status registers of the design.
Software should read and write these registers using 32-bits accesses.

Registers Summary

Name	Offset	Size	Reset Value	Description
I2C_MTXR	0x0000	W	0x00000000	Master transmit register input
I2C_MRXR	0x0004	W	0x00000000	Master receive register output
I2C_STXR	0x0008	W	0x00000000	Slave transmit register input
I2C_SRXR	0x000C	W	0x00000000	Slave receive register output
I2C_SADDR	0x0010	W	0x000003FF	I2C controller slave address
I2C_IER	0x0014	W	0x00000000	Enable/Mask interrupts generated by the I2C controller
I2C_ISR	0x0018	W	0x00000000	Interrupt status register
I2C_LCMR	0x001C	W	0x00000000	I2C line command register
I2C_LSR	0x0020	W	0x00000000	I2C core status
I2C_CONR	0x0024	W	0x00000000	I2C operation register 1
I2C_OPR	0x0028	W	0x00000000	I2C operation register 2

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access

Detail Register Description

I2C_MTXR

Address: Operational Base + offset (0x0000)

This register contains data to be transmitted on the I2C for master purpose.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	I2C master transmit data register.

I2C_MRXR

Address: Operational Base + offset (0x0004)

This register contains data to be received on the I2C for master purpose.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	I2C master receive data register.

I2C_STXR

Address: Operational Base + offset (0x0008)

This register contains data to be transmitted on the I2C for slave purpose.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	I2C slave transmit data register.

I2C_SRXR

Address: Operational Base + offset (0x000C)

This register contains data to be received on the I2C for slave purpose.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	I2C slave receive data register.

I2C_SADDR

Address: Operational Base + offset (0x0010)

This register contains address to be matched on the I2C for slave purpose.

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9:0	RW	0x3FF	Slave address.

I2C_IER

Address: Operational Base + offset (0x0014)

This register contains the bits to control the interrupt generation of I2C controller.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7	RW	0x0	Arbitration lose interrupt enable bit. “1” - enable. “0” - disable.
6	RW	0x0	Abnormal stop interrupt enable bit. “1” - enable. “0” - disable.
5	RW	0x0	Broadcast address matches (address zero) interrupt enable bit. “1” - enable. “0” - disable.
4	RW	0x0	Slave address matches interrupt enable bit. “1” - enable. “0” – disable.
3	RW	0x0	Slave ACK period interrupt enable bit (SRX mode). “1” - enable. “0” - disable.
2	RW	0x0	Slave receives ACK interrupt enable bit (STX mode). “1” - enable. “0” - disable.
1	RW	0x0	Master ACK period interrupt enable bit (MRX mode). “1” - enable. “0” - disable.
0	RW	0x0	Master receives ACK interrupt enable bit (MTX mode). “1” - enable. “0” - disable.

I2C_ISR

Address: Operational Base + offset (0x0018)

I2C interrupt status register.

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7	RW	0x0	Arbitration lose status bit. “1” – Arbitration lose occurs

			“0” – No Arbitration lose occurs Write this bit “0” to clear. Write “1” will not change this bit.
6	RW	0x0	Abnormal stop status bit. “1” – Abnormal stop occurs “0” – No abnormal stop occurs Write this bit “0” to clear. Write “1” will not change this bit.
5	RW	0x0	Broadcast address (address zero) matches status bit. “1” – Broadcast address matches. “0” – No broadcast address matches. Write this bit “0” to clear. Write “1” will not change this bit.
4	RW	0x0	Slave address matches status bit. “1” – Slave address matches. “0” – No slave address matches (When read). Clear slave address matches interrupt (When write). Write this bit “0” to clear. Write “1” will not change this bit.
3	RW	0x0	Slave ACK period interrupt status bit (SRX mode). “1” – interrupt generation “0” – no interrupt generation Write this bit “0” to clear. Write “1” will not change this bit.
2	RW	0x0	Slave receives ACK interrupt status bit (STX mode). “1” – interrupt generation “0” – no interrupt generation Write this bit “0” to clear. Write “1” will not change this bit.
1	RW	0x0	Master ACK period interrupt status bit (MRX mode). “1” – interrupt generation “0” – no interrupt generation Write this bit “0” to clear. Write “1” will not change this bit.
0	RW	0x0	Master receives ACK interrupt status bit (MTX mode). “1” – interrupt generation “0” – no interrupt generation Write this bit “0” to clear. Write “1” will not change this bit.

I2C_LCMR

Address: Operational Base + offset (0x001C)

This register contains the bits to generate start and stop commands of I2C controller.

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	“RESUME” condition generation bit.

			“1” - enable. “0” - disable. Write “1” to start “RESUME” action. It cannot be cancelled by write “0”. This bit is self-cleared after “RESUME” action. Write “0” is undefined.
1	RW	0x0	“STOP” condition generation bit. “1” - enable. “0” - disable. Write “1” to start “STOP” action. It cannot be cancelled by write “0”. This bit is self-cleared after “STOP” action. Write “0” is undefined.
0	RW	0x0	“START” condition generation bit. “1” - enable. “0” - disable. Write “1” to start “START” action. It cannot be cancelled by write “0”. This bit is self-cleared after “START” action. Write “0” is undefined.

I2C_LSR

Address: Operational Base + offset (0x0020)

This register is used to read I2C core status.

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1	R	0x0	I2C receives ACK status bit (MTX and STX modes). “0” – I2C bus receives ACK “1” – I2C bus receives NAK.
0	R	0x0	I2C core busy status bit. ‘1’ – After START condition detect. ‘0’ – After STOP condition detect.

I2C_CONR

Address: Operational Base + offset (0x0024)

This register is used to set the operation modes and ACK enable bit of I2C controller.

Bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4	RW	0x0	I2C bus acknowledge enable register “0” – enable (ACK). “1” – disable (NAK). When enable, the SDA is free (TX mode), and is LOW (RX mode) in acknowledge time.
3	RW	0x0	Master receive/transmit mode select bit “0”: receive. “1”: transmit.
2	RW	0x0	Master port enable bit “0”: disable. “1”: enable.
1	RW	0x0	Slave receive/transmit mode select bit “0”: receive. “1”: transmit.
0	RW	0x0	Slave port enable bit “0”: disable. “1”: enable.

I2C_OPR

Address: Operational Base + offset (0x0028)

This register is used to set I2C core enable bit, frequency divider factor, internal state machine reset and slave address length modes.

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	I2C slave address mode bit. “0” – 7 bits address mode. “1” – 10 bits address mode.
7	RW	0x0	I2C state machine (both master/slave) reset bit. “0” – don’t reset state machine “1” – reset state machine
6	RW	0x0	I2C core enable bit “0” – disable I2C controller. “1” – enable I2C controller.
5:0	RW	0x0	I2C clock divisor bits (I2CCDVR). The value of I2CCDVR is used to generate the transmit and receive bit rate of the I2C master part. And the bit rate equation will be described more detail in section below.

Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

The I2C controller supports both the Master and Slave functions. It also supports the 7-bits/10-bits addressing mode and support general call address. The maximum clock frequency and transfer rate can be up to 400Kbit/sec.

The operations of I2C controller is divided to 3 parts and described separately: initialization, master mode programming, and slave mode programming.

More details are listed in the Program Sequence section.

Initialization

The I2C controller is based on AMBA APB bus architecture and usually is part of a SOC. So before I2C operates, some system setting & configuration must be conformed, which includes:

- I2C Register memory mapping
- I2C interrupt connection type: CPU interrupt scheme should be considered. If the I2C interrupt is connected to extra Interrupt Controller module, we need decide the INTC vector.
- I2C Clock Rate: The I2C controller uses the APB clock/5 as the system clock so the APB clock will determine the I2C bus clock. The correct register setting is subject to the system requirement.

Master Mode Programming

- SCL Clock: When the I2C controller is programmed in Master mode, the SCL frequency is determined by I2C_OPR register. The SCL frequency is calculated by the following formula

$$\text{SCL Divisor} = (\text{I2CCDVR}[5:3] + 1) \times 2^{(\text{I2CCDVR}[2:0] + 1)}$$

$$\text{SCL} = \text{PCLK} / 5 * \text{SCLK Divisor}$$

The I2CCDVR[2:0] is coarse factor and I2CCDVR[5:3] is fine tune factor for the SCL clock.

- Data Receiver Register Access: The Master data receive register (I2C_MRXR) can only be correctly accessed at Master Receiver Mode. When accessing the I2C_MRXR register, make sure the I2C_CONR[3:2] is set to receiver mode.
- Start Command and 1'st Byte Address: The I2C controller combines the start command and 1'st byte address data output together. SW cannot issue start command only. So before issuing start command, SW must prepare the correct address data (include R/W bit) to the I2C_MTXR register. Since the I2C protocol allows the repeated start command, the resume command needs to be issued with repeated start.
- Interrupt and Resume: I2C controller interrupt status is generated by HW and cleared by SW. The clear scheme is to write 0 to the correspond bit. Writing 1 to interrupt status bits will not get any affect. The interrupt clear affect only the interrupt status bits. For reasons of flexibility and interrupt latency reduction, the interrupt clear will not resume the I2C function. The I2C function resumes when I2C_LCMR register resume bit is set to 1. This separates the I2C service routine from the ISR (Interrupt Service Routine). SW must carefully design the I2C service routine because the I2C controller may be locked in some special state. When operating at Transmit mode, SW should prepare the next transmit data on the I2C_MTXR register before issue the resume command.
- Read/Write Command: The I2C Read/Write command depends on the last bit of address. SW should take the responsibility of Read/Write control. The Read/Write control and the Master Receive/Transmit mode setting must be correctly set before resume the I2C function.
- Multi-Master Arbitration: When I2C controller works on Multi-Master I2C bus, HW will detect the bus busy condition and arbitration loss. When it happens, HW will stop the transaction and notify SW. SW should take the responsibility of re-transmit and time-out handling.
- Master Interrupt Condition: There are 3 interrupt bits in I2C_ISR register related to master mode.

Master ACK (Bit 0): The bit is asserted when Master receives ACK. In other words, the interrupt happens only at Master Transmit Mode.

Master ACK Period (Bit 1): The bit is asserted when Master needs to send out ACK. In other words, the interrupt happens only at Master Receive Mode.

Arbitration Loss (Bit 7): The bit is asserted when Master starts a transaction but lose the bus arbitration.

- Stop Command: Master can issue Stop command when receive Master ACK or Master ACK Period interrupt. Because the Stop command is attached at the end of a transaction, the resume command needs to be issued with stop command. According to the I2C spec, the NAK must be sent out at Receive Mode in Master ACK Period before Stop.

Slave Mode Programming

- Data Receiver Register Access: The Slave data receive register (I2C_SRXR) can only be correctly accessed at Slave Receiver Mode. When accessing the I2C_SRXR register, make sure the I2C_CONR[1:0] is set to slave mode.
- 7 Bits and 10 Bits Address: I2C Slave transaction starts when Slave address is matched. The I2C controller supports both the standard 7 bits address mode and 10 bits address mode. The I2C controller filters out the transaction of which address is not matched with the I2C_SADDR register. However, I2C controller only filters the 1st address mode, SW should take care the 2nd address for 10 bits address mode. The 7 or 10 bits address mode is set with I2C_OPR[8].
- 7 Bits Address Setting: Slave function begins upon detecting a received address matched with I2C_SADDR register. The I2C_SADDR does not include the Read/Write bit. The I2C_SADDR[9:7] is ignored at 7 bits address mode.
- 10 Bits Address Setting: The I2C_SADDR register must be correctly initialized before slave function start to work. The I2C_SADDR does not include the Read/Write bit. The I2C controller detects the received 1st address by the I2C_SADDR[9:8] combined with the 10 bits mode address prefix(0b11110xx).

Address Matching: The 1st transaction received by I2C controller in slave mode is the address. When the address matched with the slave address of the I2C controller, it will notify SW with an interrupt. I2C_ISR[4] high represents the incoming slave address matched with the specific slave address of the I2C controller. I2C_ISR[5] high represents the broadcast, the general call (0x00), is detected.

When address matched, SW should read the I2C_SRXR register to figure out the transaction is a read or write and set the slave mode accordingly before resume ACK. If the next transaction is a read, the read data needs to be prepared to the I2C_STXR before resume.

- 10 Bits Address 2nd Phase: Because the I2C controller takes care only the 1st address matching at 10 bits address mode, SW should take care the 2nd address comparison. When the 2nd byte address comparison fails, SW should issue a reset, I2C_OPR[7], and issue a resume. After reset and resume, HW will ignore the rest coming transactions until next start detected.
- Interrupt and Resume: the interrupt is generator by I2C controller and cleared by SW. The clear scheme is to write 0 to the corresponding bit. Writing 1 to interrupt status bits will not get any affect. The interrupt clear affect only the interrupt status bits. For reasons

of flexibility and interrupt latency reduction, the interrupt clear will not resume the I2C function. The I2C function resumes when I2C_LCMR register resume bit is set to 1. This separates the I2C service routine from the ISR (Interrupt Service Routine). SW must carefully design the I2C service routine because the I2C controller may be locked in some special state. When operating at Transmit mode, SW should prepare the next transmit data on the I2C_STXR register before issue the resume command.

- Read/Write Command: The I2C Read/Write command depends on the last bit of address. SW should take the responsibility of Read/Write control. The Read/Write control and the Master Receive/Transmit mode setting must be correctly set before resume the I2C function.

- Slave Interrupt Condition: There are 5 interrupt bits in I2C_ISR register related to slave mode.

Slave ACK (Bit 2): The bit is asserted when Slave receives ACK. In other words, this interrupt happens only at Slave Transmit Mode.

Slave ACK Period (Bit 3): The bit is asserted when Slave needs to send out ACK. In other words, this interrupt happens only at Slave Receive Mode.

Slave address match (bit 4): The bit is asserted when the coming address is matched with the slave address of the I2C controller. Slave ACK interrupt is not asserted when Slave address match interrupt is asserted.

Broadcast address match (bit 5): The bit is asserted when the coming address matched with general call (0x00) address. Slave ACK interrupt is not asserted when general call address match interrupt is asserted.

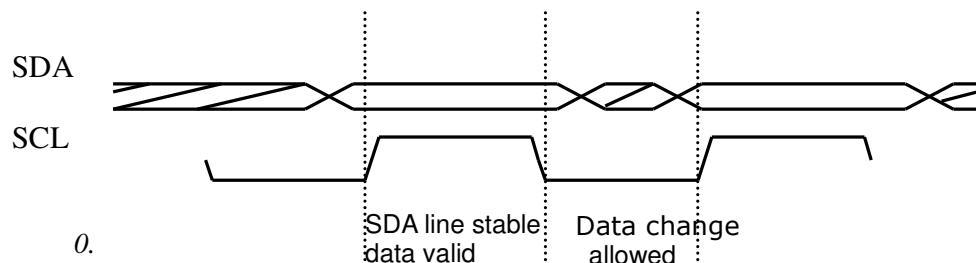
Abnormal stop occurs (bit 6): The bit is asserted when Slave receive abnormal stop.

I2C controller data transfer waveform

- Bit transferring

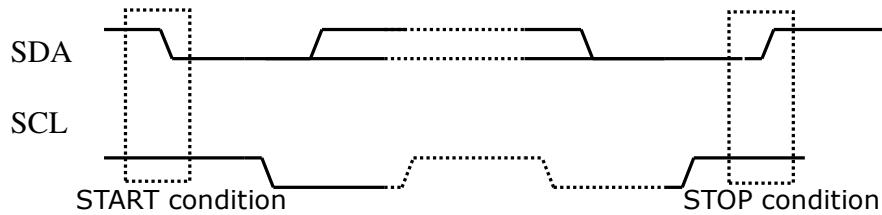
(a) Data Validity

The SDA line must be stable during the high period of SCL, and the data on SDA line can only be changed when SCL is in low state.



(b) START and STOP conditions

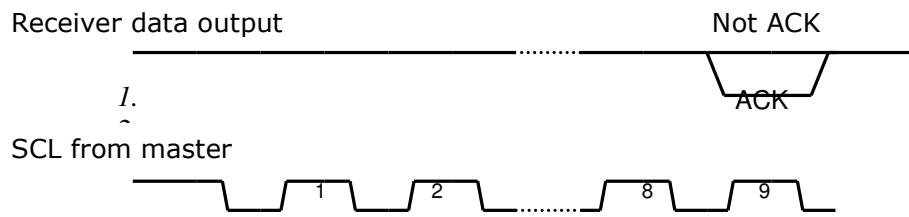
START condition occurs when SDA goes low while SCL is in high period. STOP condition is generated when SDA line goes high while SCL is in high state.



Data transfer

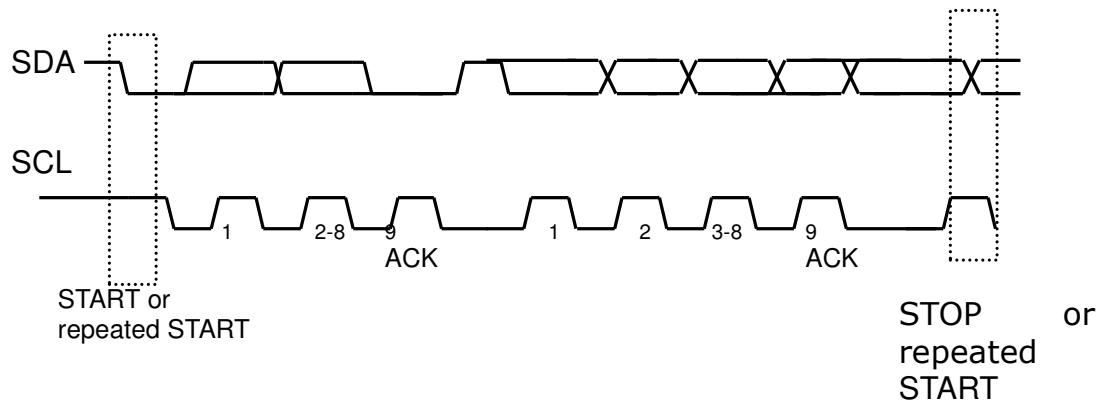
(a) Acknowledge

After a byte of data transferring (clocks labeled as 1~8), in 9th clock the receiver must assert an ACK signal on SDA line, if the receiver pulls SDA line to low, it means “ACK”, on the contrary, it’s “NOT ACK”.



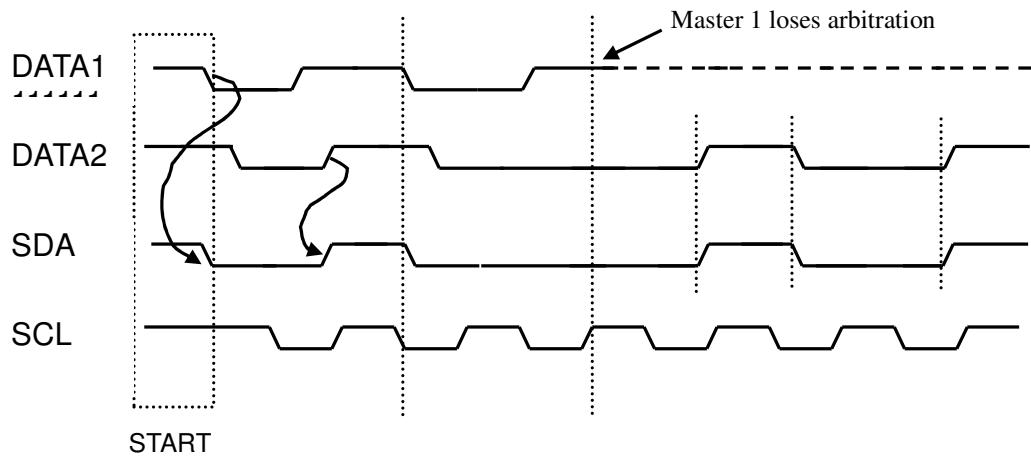
(b) Byte transfer

The master own I2C bus might initiate multi byte to transfers to a slave, the transfers starts from a “START” command and ends in a “STOP” command. After every byte transfer, the receiver must reply an ACK to transmitter.



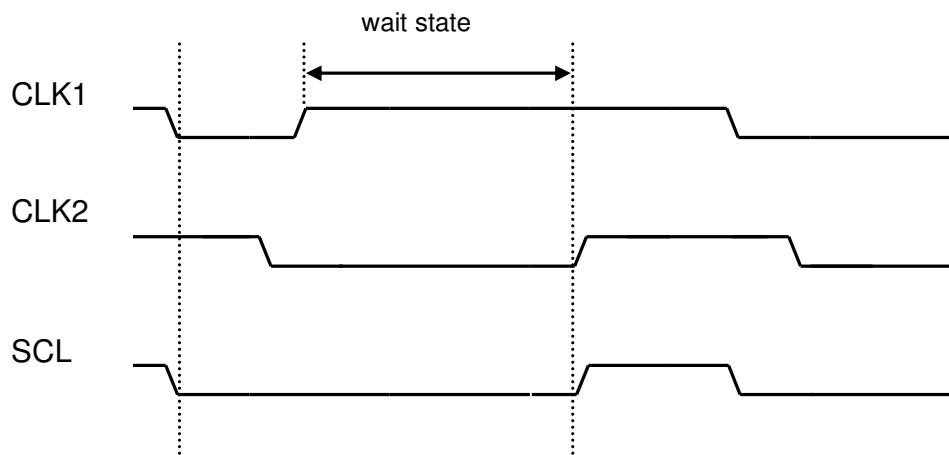
(c) Arbitration

Arbitration takes place at SDA line, while the SCL line is at high level. The master transmits a high level, while another master transmits a low level will lose arbitration.



(d) Synchronization

Clock synchronization is performed using the wired-and connection of I2C interface to the SCL line.

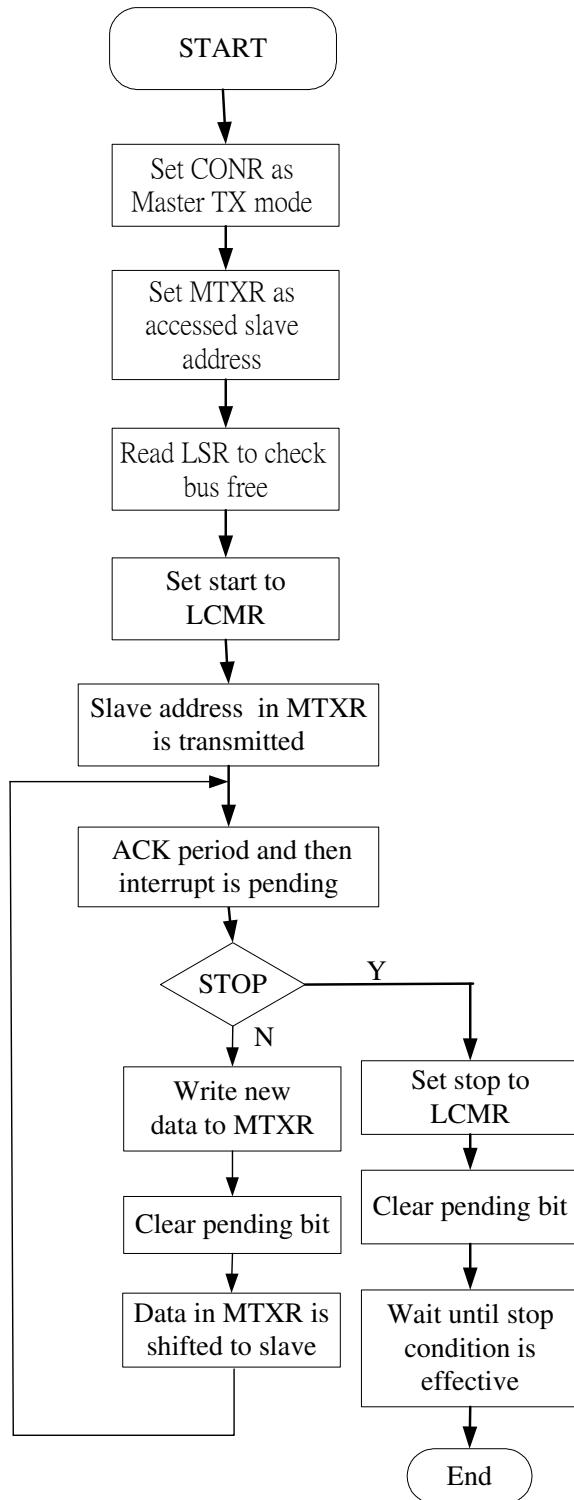


Programming sequence

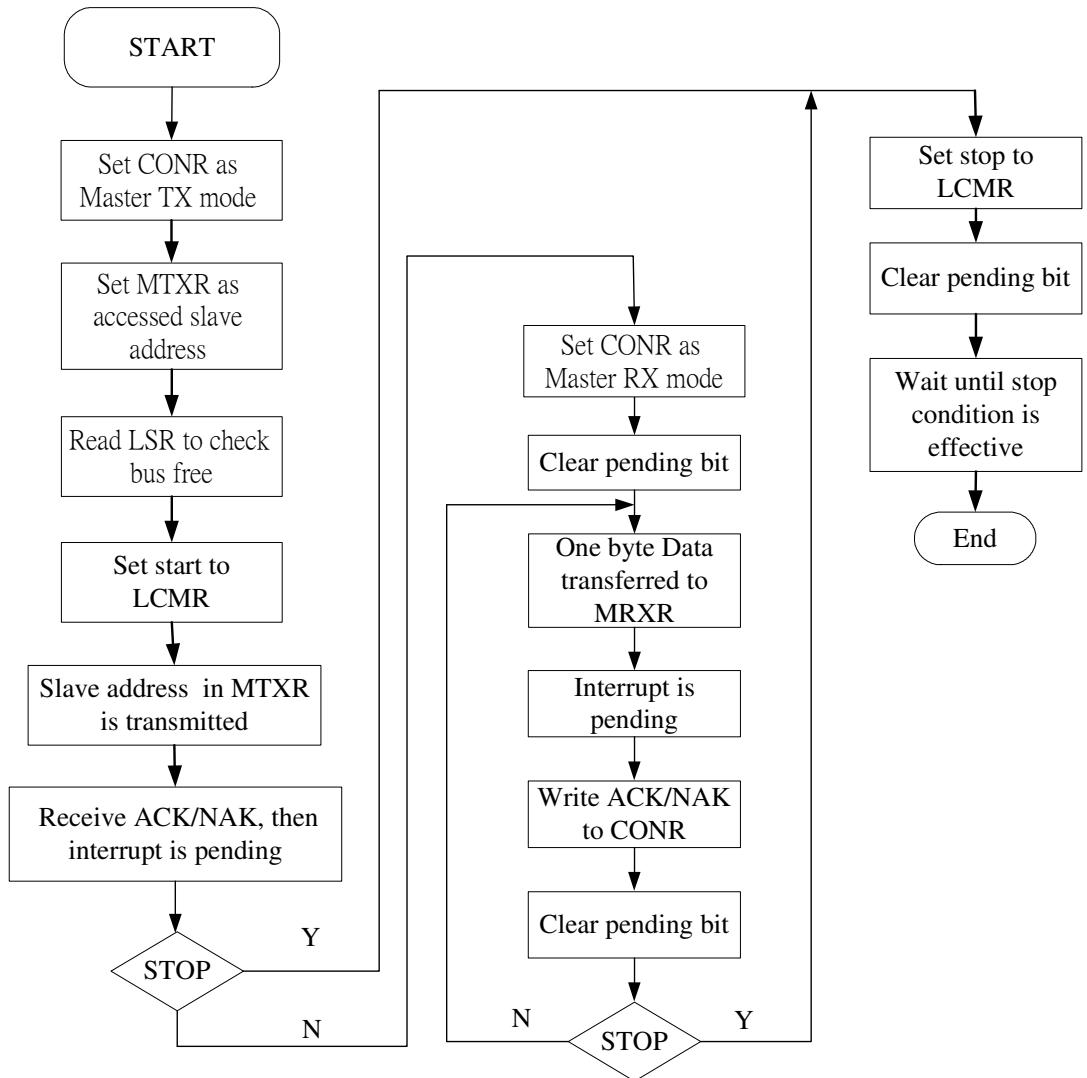
Control/Status Register programming sequence

The I2C controller core operation flow chart below is to describe how the software configures and performs an I2C transaction through this I2C controller core. Descriptions are divided into 4 sections, master transmit mode, master receive mode, slave transmit mode and slave receive mode. Users are strongly advised to following.

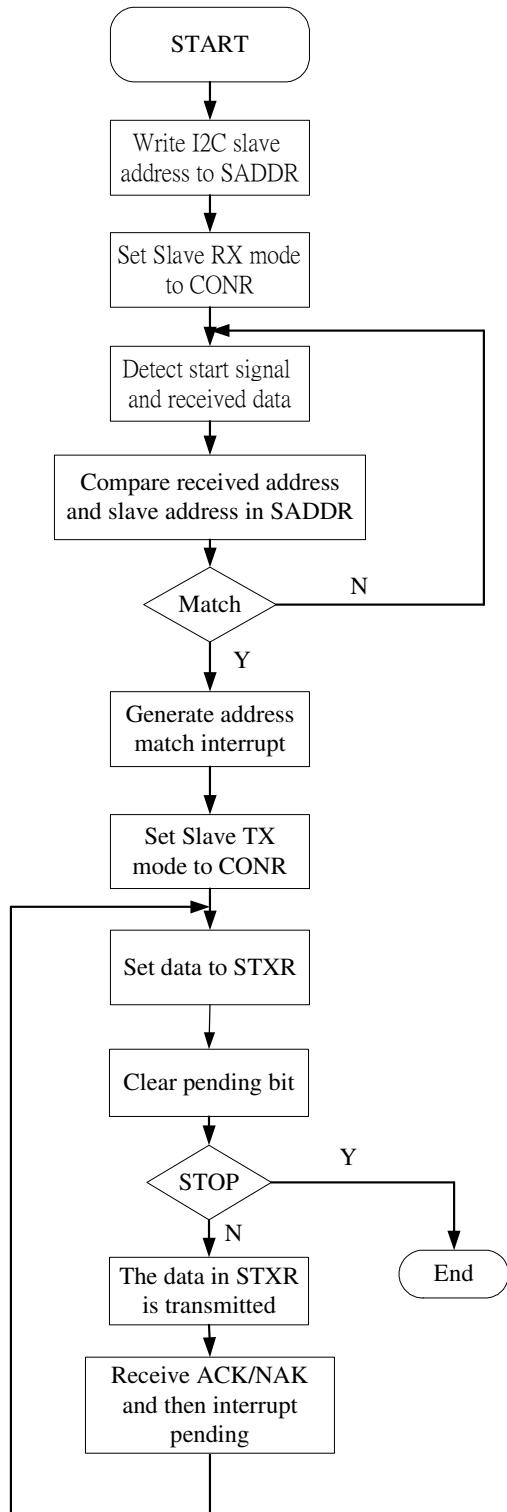
Operations for Master/Transmitter Mode



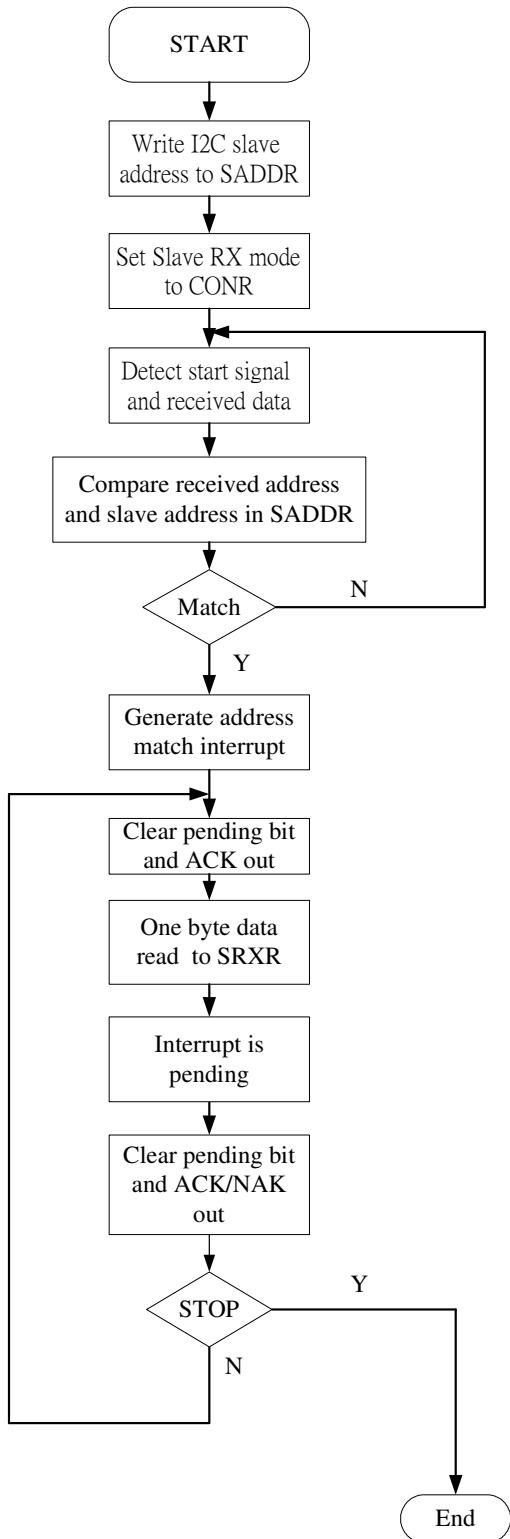
Operations for Master/Receiver Mode



Operations for Slave/Transmitter Mode



Operations for Slave/Receiver Mode



Chapter 23 I2S Controller

Overview

The I2S Controller is designed for interfacing between the APB bus and the I2S bus. The I2S bus (Inter-IC sound bus) is a serial link for digital audio data transfer between devices in the system and was invented by Philips Semiconductor. Now it is widely used by many semiconductor manufacturers.

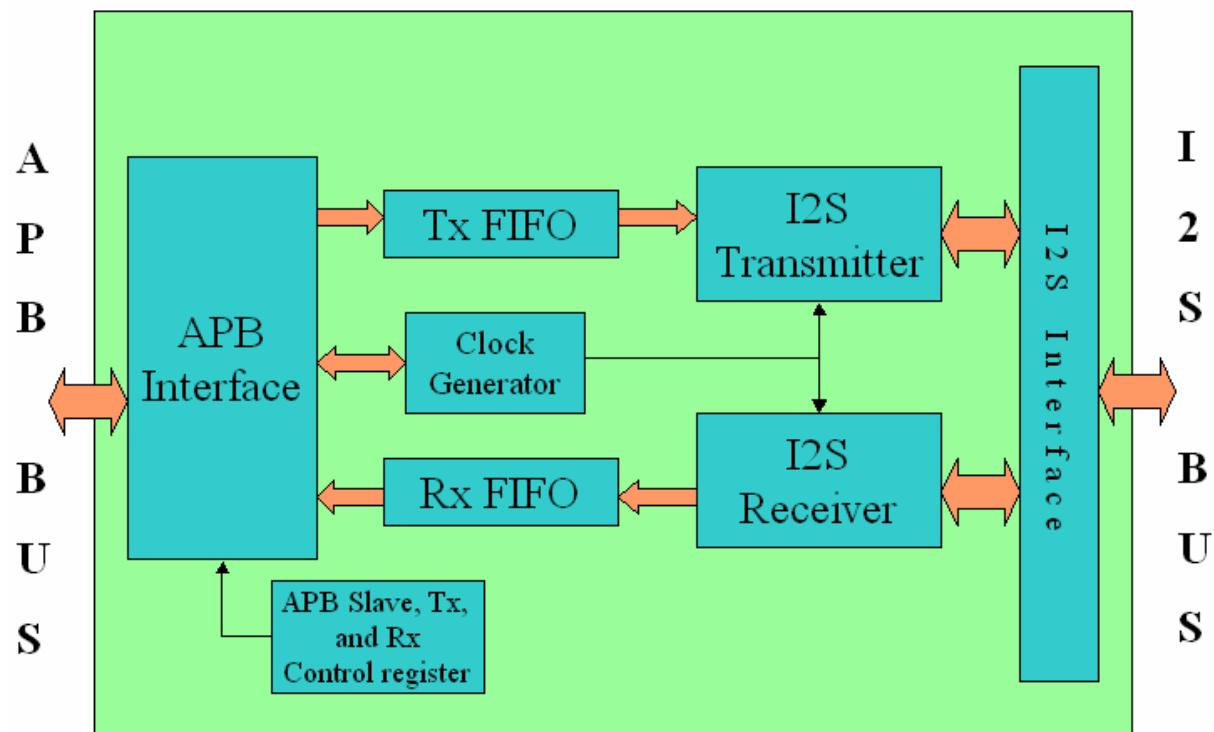
Devices often use the I2S bus are ADC, DAC, DSP, CPU, etc. With the I2S interface, we can connect audio devices and the embedded SoC platform together and provide an audio interface solution for the system.

Key Features

- Have both transmitter and receiver
- Support mono/stereo audio file
- Support audio resolution: 8, 16 bits
- Support audio sample rate from 32 to 96 KHz
- Support I2S, Left-Justified, Right-Justified digital serial audio data interface
- Have 2 FIFOs with hardware configurable size for Tx and Rx transfer

Architecture

Block Diagram



Block Descriptions

APB Interface/Control Register

The APB Interface implements the APB slave operation. It contains control registers of APB slave, transmitter and receiver inside. Through the APB slave, system can control this I2S design.

Clock Generator

The Clock Generator implements clock generation function. The input source clock to the module is MCLK_I2S, and by the divider of the module, the clock generator generates SCLK and LRCK to transmitter and receiver.

I2S Transmitter

The I2S Transmitter implements transmission operation. The transmitter can act as either master or slave, with I2S, Left-Justified, and right-Justified serial audio interface. The digital data input is through TX FIFO, and output serial data to I2S Interface.

I2S Receiver

The I2S Receiver implements Receive operation. The receiver can act as either master or slave, with I2S, Left-Justified, and right-Justified serial audio interface. The serial data input is from I2S interface, and input digital data to RX FIFO.

TX FIFO/RX FIFO

The TX FIFO/RX FIFO is the buffer to store audio data. Both FIFOs have their FIFO control circuit.

I2S Interface

The I2S Interface is used to connect I2S bus and both transmitter and receiver of the design. For transmitter, it has four stereo output channels to connect devices, like audio DAC. It is only one of four stereo channels active at one time. For receiver, it has one stereo input channel. The I2S Interface also implements loop-back mode. At loop-back mode, the TX channel 0 is connected directly to RX channel.

Registers

Registers Summary

Name	Offset	Size	Reset Value	Description
I2S_OPR	0x0000	W	0x00000018	I2S version control and operation start register.
I2S_TXR	0x0004	W	0x00000000	I2S transmitter FIFO input.
I2S_RXR	0x0008	W	0x00000000	I2S receiver FIFO output.
I2S_TXCTL	0x000C	W	0x00010811	I2S transmitter control register.
I2S_RXCTL	0x0010	W	0x00010811	I2S receiver control register.
I2S_FIFOSTS	0x0014	W	0x00010055	I2S transmit and receive FIFO control register.
I2S_IER	0x0018	W	0x00000000	I2S interrupt Enable/Mask register.
I2S_ISR	0x001C	W	0x00000000	I2S interrupt status register.

Notes:

Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** – WORD (32 bits) access

Detail Register Description

I2S_OPR

Address: Operational Base + offset (0x0000)

This register contains I2S Controller's version and transmit/receive operation bit.

Bit	Attr	Reset Value	Description
23:18	-	-	Reserved
17	W	0x0	Reset Tx logic. Writing to this bit will reset Tx logic and its FSM.
16	W	0x0	Reset Rx logic. Writing to this bit will reset Rx logic and its FSM.
15:7	-	-	Reserved
6	RW	0x0	HDMA REQ1 Disable 0 : Enable HDMA REQ1 1 : Disable
5	RW	0x0	HDMA REQ2 Disable 0 : Enable HDMA REQ2 1 : Disable
4	RW	0x1	HDMA_REQ1_CH This bit is to indicate the Hardware DMA IF1 is used for which FIFO 0 : TX 1 : RX
3	RW	0x1	HDMA_REQ2_CH This bit is to indicate the Hardware DMA IF2 is used for which FIFO

			0 : TX 1 : RX
2	RW	0x0	I2S loop-back mode. This bit is to indicate the operation mode of the I2S Controller is in a normal operation mode or in a loop-back mode. 0 : Normal operation mode. 1 : Loop-back mode.
1	RW	0x0	I2S transmit-operation start. The transmitter starts to send SCLK and LRCK signals and transmit data stored in the Tx FIFO to receiver after this bit is set to 1. (Transmitter acts as a master)
0	RW	0x0	I2S receive-operation start. The receiver starts to send SCLK and LRCK signals and receive data from transmitter after this bit is set to 1. (Receiver acts as a master)

I2S_TXR

Address: Operational Base + offset (0x0004)

I2S transmit FIFO input.

Bit	Attr	Reset Value	Description
31:0	W	0x0	Written data in this register will be transmitted on the I2S bus through the Transmit FIFO.

I2S_RXR

Address: Operational Base + offset (0x0008)

I2S receive FIFO output.

Bit	Attr	Reset Value	Description
31:0	R	0x0	Received data from I2S bus through the Receive FIFO will be read in this register.

I2S TXCTL

Address: Operational Base + offset (0x000C)

This register controls the setting of the transmitter.

5:4	RW	0x1	Sample data resolution. Number of bits that are transmitted from each audio word. 0x0 : 8 bits 0x1 : 16 bits 0x2 ~ 0x3: Reserved.
3	RW	0x0	Mono/Stereo mode. When the bit is set to 1, transmitter is at Mono mode, and data output from left channel. Default is stereo mode.
2:1	RW	0x0	Bus Interface mode Choose the type of the bus interface. 0x0 : I2S 0x1 : Left - Justified 0x2 : Right - Justified 0x3 : reserved
0	RW	0x1	Master/Slave mode select. This bit decides that transmitter acts as a master or slave. 1 : Master mode 0 : Slave mode

I2S_RXCTL

Address: Operational Base + offset (0x0010)

This register controls the setting of the receiver.

			0x0 : I2S 0x2 : Right - Justified 0x3 : reserved	0x1 : Left - Justified
0	RW	0x1	Master/Slave mode select. This bit decides that transmitter acts as a master or slave. 1 : Master mode 0 : Slave mode	

I2S_FIFOSTS

Address: Operational Base + offset (0x0014)

This register shows FIFO status and interrupts trigger level.

Bit	Attr	Reset Value	Description	
31:20	-	-	Reserved	
19:18	RW	0x0	Tx interrupt trigger level. 0x0 : almost empty 0x2 : almost full	0x1 : half full 0x3 : reserved
17:16	RW	0x1	Rx interrupt trigger level. 0x0 : almost empty 0x2 : almost full	0x1 : half full 0x3 : reserved
15:10	-	-	Reserved	
9	R	0x0	Tx FIFO half full flag. This bit is set whenever Tx FIFO is half full.	
8	R	0x0	Rx FIFO half full flag. This bit is set whenever Rx FIFO is half full.	
7	R	0x0	Tx FIFO almost full flag. This bit is set whenever Tx FIFO is almost full.	
6	R	0x1	Tx FIFO almost empty flag. This bit is set whenever Tx FIFO is almost empty.	
5	R	0x0	Rx FIFO almost full flag. This bit is set whenever Rx FIFO is almost full.	
4	R	0x1	Rx FIFO almost empty flag. This bit is set whenever Rx FIFO is almost empty.	
3	R	0x0	Tx FIFO full flag. This bit is set whenever Tx FIFO is full.	
2	R	0x1	Tx FIFO empty flag. This bit is set whenever Tx FIFO is empty.	
1	R	0x0	Rx FIFO full flag. This bit is set whenever Rx FIFO is full.	
0	R	0x1	Rx FIFO empty flag. This bit is set whenever Rx FIFO is empty.	

I2S_IER

Address: Operational Base + offset (0x0018)

This register contains the bits to control the interrupt generation of this I2S Controller.

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Tx FIFO data trigger interrupt enable bit.

			This bit enables the interrupt when Tx FIFO's trigger level is reached. 0x0 : Disable. 0x1 : Enable.
1	RW	0x0	Rx FIFO data trigger interrupt enable bit. This bit enables the interrupt when Rx FIFO's trigger level is reached. 0x0 : Disable. 0x1 : Enable.
0	RW	0x0	Rx FIFO overrun interrupt enable bit. This bit enables the interrupt when Rx FIFO overrun condition is occurred. 0x0 : Disable. 0x1 : Enable.

I2S_ISR

Address: Operational Base + offset (0x001C)

I2S interrupt status register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	Tx FIFO almost empty interrupt. This bit is set when Tx FIFO's trigger level is reached, and CPU wishes to keep transmitting data to the device. The bit is cleared when data in Tx FIFO is above trigger level.
1	R	0x0	Rx FIFO data trigger interrupt. This bit is set when Rx FIFO's trigger level is reached. The bit is cleared when data in Rx FIFO is below trigger level.
0	R	0x0	Rx FIFO overrun interrupt. This bit is set when Rx FIFO is full and another character has been received in the receiver shift register. If another character is starting to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. The bit is cleared after Rx FIFO is cleared by software simultaneously.

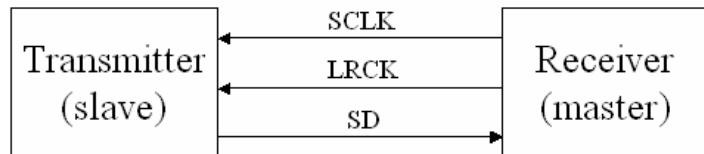
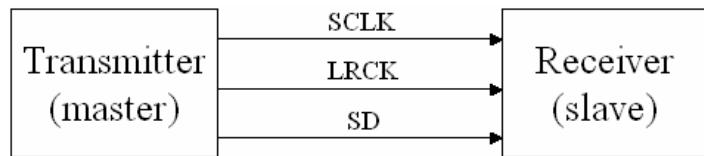
Notes: Attr: **RW** – Read/writable, **R** – Read only, **W** – Write only

Functional Description

Operation

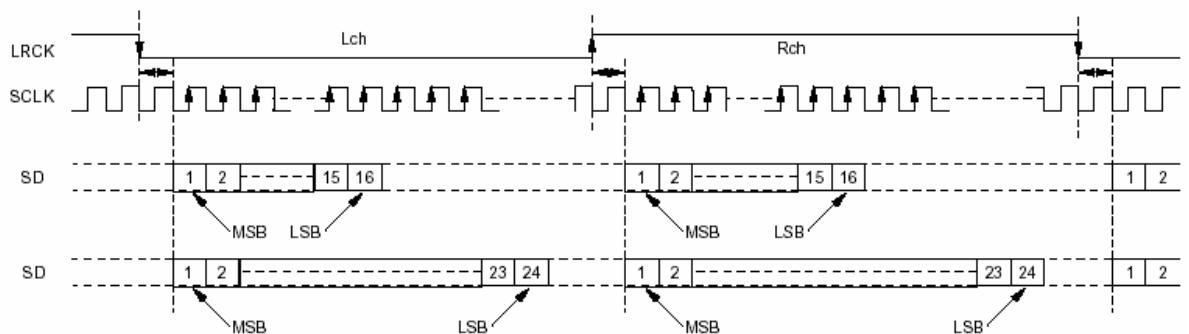
Digital audio serial data interface format

The I2S interface supports three digital serial data interface formats for audio data transfer: I2S, Left-Justified, Right-justified. All these formats have SCLK, LRCK, and SD signals. The signal's direction is as below:



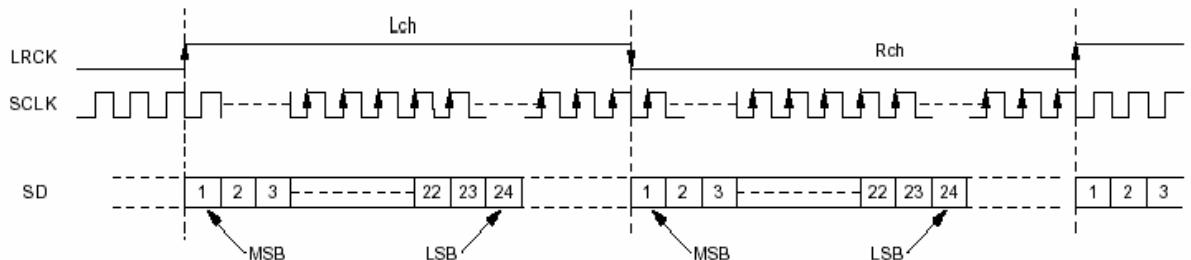
I2S Interface format

This is the waveform of I2S interface. For LRCK signal, it goes “low” to indicate left channel and “high” to right channel. For SD signal, it transfers MSB-first and sends the MSB bit one SCLK clock cycle after LRCK goes low.



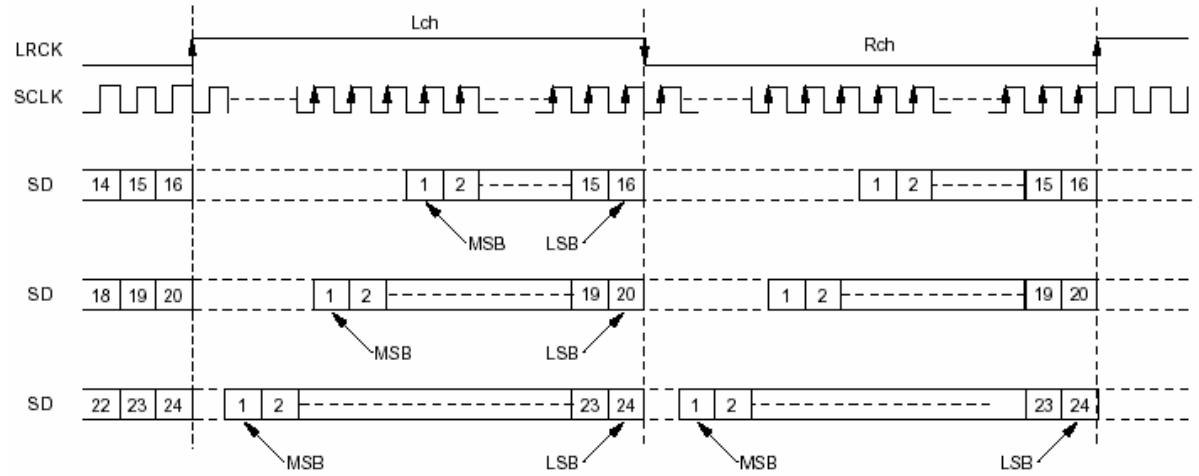
Left-Justified Interface format

This is the waveform of Left-Justified interface. For LRCK signal, it goes “high” to indicate left channel and “low” to right channel. For SD signal, it transfers MSB-first and sends the MSB bit at the same time when LRCK goes high.



Right-justified Interface format

This is the waveform of Right-Justified interface. For LRCK signal, it goes “high” to indicate left channel and “low” to right channel. For SD signal, it transfers MSB-first; but different from I2S or Left-Justified interface, its data is aligned to LSB at falling edge of the LRCK signal.



I2S Normal Operation

There are four conditions: transmitter-master & receiver-master; transmitter-master & receiver-slave; transmitter-slave & receiver-master; transmitter-slave & receiver-slave. When transmitter acts as a master, it sends all signals to receiver (slave), and CPU control when to send clock and data to the receiver. When acting as a slave, SD signal still goes from transmitter to receiver, but SCLK and LRCK signals are from receiver (master) to transmitter. Based on three interface specifications, transmitting data should be ready before transmitter receives SCLK and LRCK signals. CPU should know when the receiver to initialize a transaction and when to send data.

When the receiver acts as a master, it sends SCLK and LRCK signals to the transmitter (slave) and receives serial data. So CPU must tell the transmitter when to start a transaction for it to prepare transmitting data then start a transfer and send clock and channel-select signals. When the receiver acts as a slave, CPU should only do initial setting and wait for all signals and then start reading data.

I2S initial setting

Before transmitting or receiving data, CPU need do initial setting to the I2S register. These includes CPU settings, I2S interface registers settings, and maybe the embedded SoC platform settings. These registers must be set before starting data transfer. Detail settings about the registers of the I2S interface refer to chapter 5 “Function Registers”.

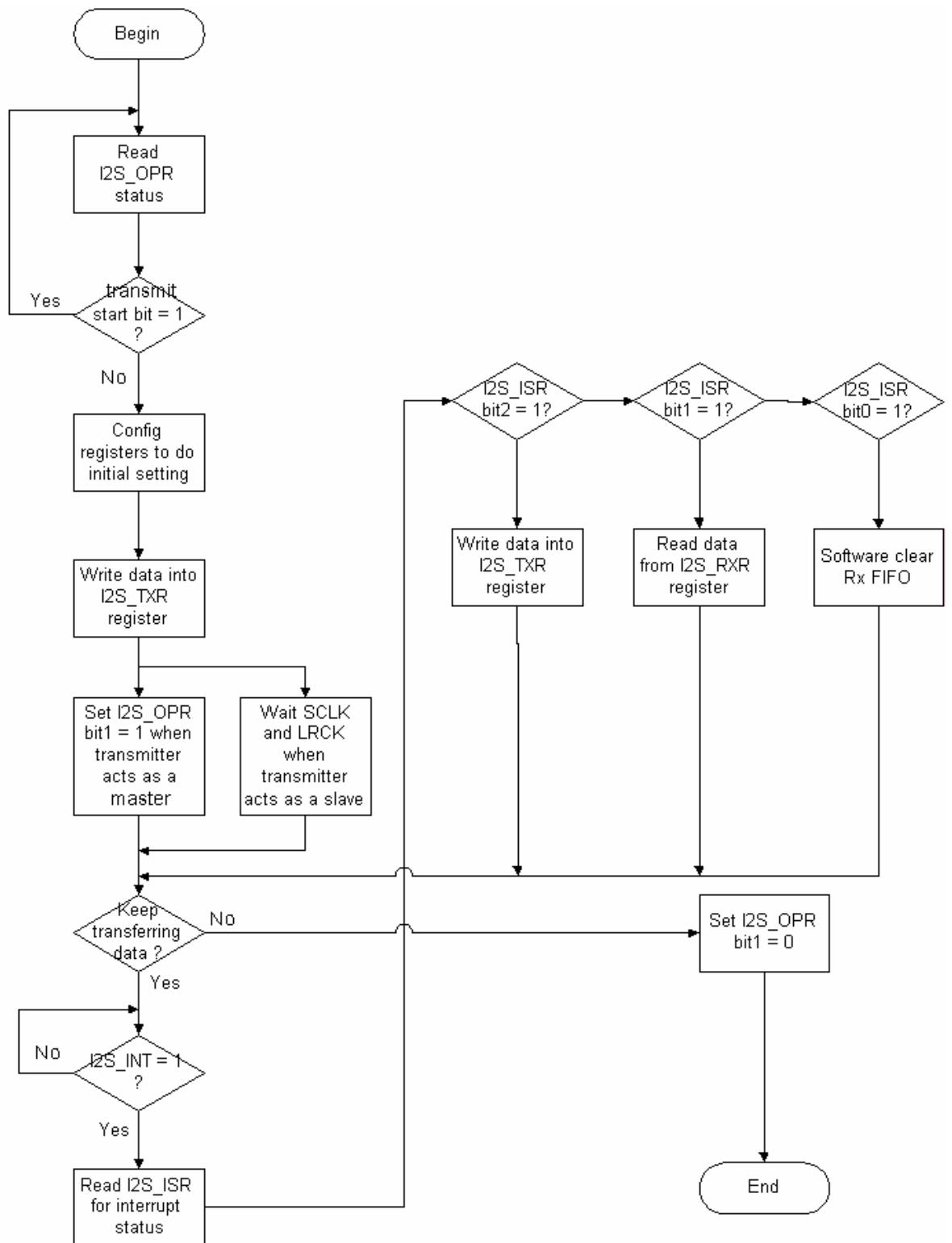
I2S Loop-back Test

The I2S interface has two operation modes: Normal mode and Loop-back mode. In the Loop-back mode operation, transmitter acts as a master and receiver acts as a slave. Transmitter device 0's output is connected to receiver's input. That is, TX_SCLKOUT[0], TX_LRCKOUT[0] and SDO[0] is connected to RX_SCLKIN, RX_LRCKIN and SDI.

Programming sequence

I2S Tx Operation Flow Chart

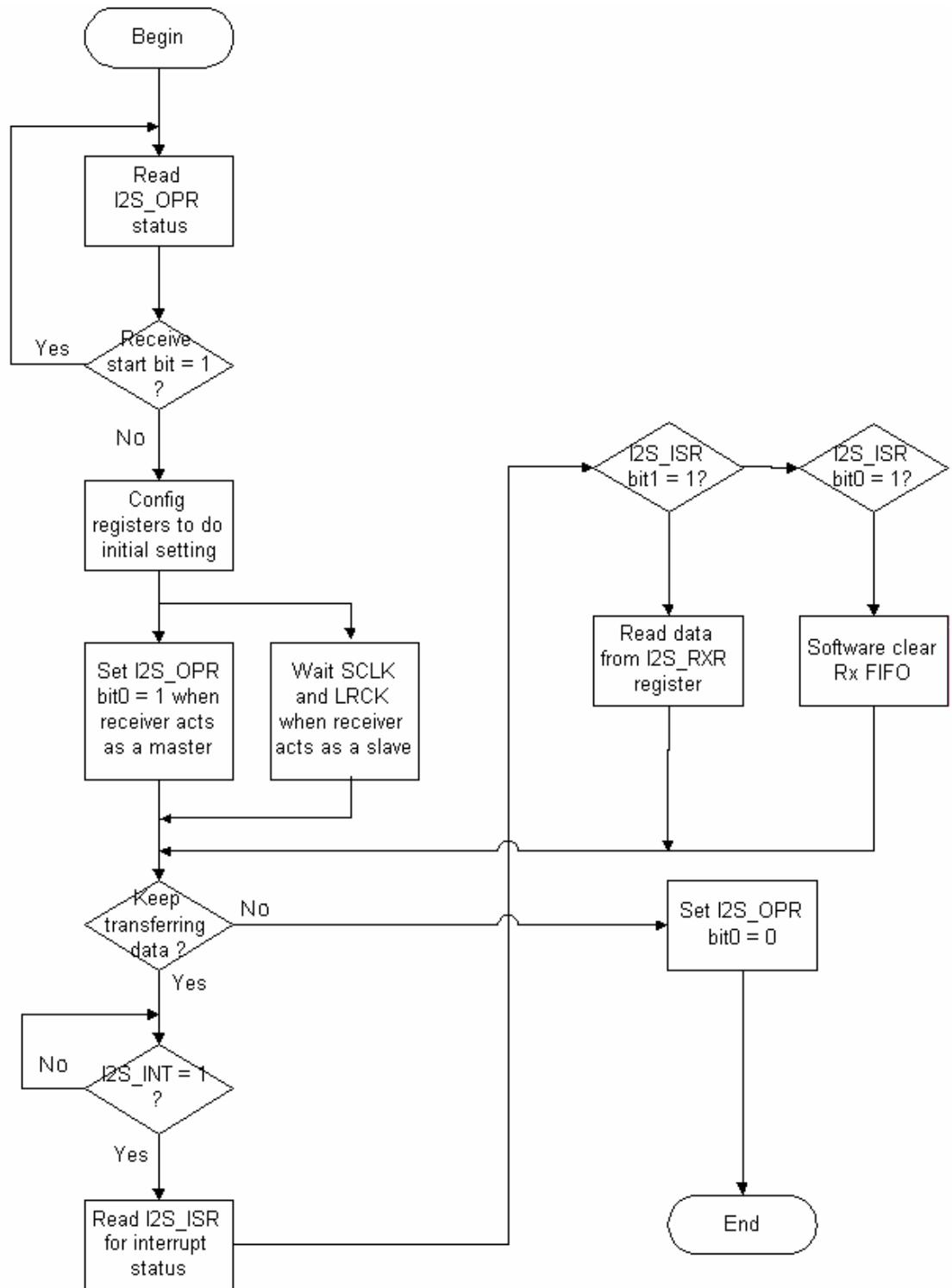
The flow chart below is to describe how the software to configure and perform an I2S transmitting transaction from transmitter's view.



I2S Rx Operation Flow Chart

The flow chart below is to describe how the software to configure and perform an I2S

receiving transaction from receiver's view.



Chapter 24 Mechanical Data

Package Dimensions

