PRODUCT DATA SHEET

RK27XX

Revision A1.1
October, 2007

TABLE OF CONTENT

About This Manual

The information in this databook includes a functional description, signal and parameter descriptions, and a memory map of RK27xx Multimedia Processors. This databook provides details on how to initialize, configure, and program the ICs.

Release Note

Document Version Information

Product Version Information	
Product Name	RK27xx
Version Number	A1.1
Previous Version	
Release Date	October 26,2007

Document History

Version Number	Release Date	Change Notes	
A 0.1	July 3 2007	First release	
A 0.2	July 22 2007	System configuration	modified
		System control unit	modified
		LCD controller	added
A0.3	August 9, 2007	Audio codec	added
A0.4	August 11, 2007	DW DMA	added
		Pin Description	modified
A1.0	Setptember 13,2007	Upated	
A1.1	October 26,2007	RK2706 Pin define	modified
		IO direction	modified
		PWM	modified

Chapter 1 Product Overview

1.1 Overview

The RK27xx is an integrated system-on-chip with Dual Core architecture, which will focus on multimedia product application such as MP3, MP4, PMP etc. The Dual-Core architecture integrated ARM7EJC and DSP micro processor and by co-operating of those two CPUs, RK27xx can get high performance on Low-Power platform . The chapter 1 and chapter 2 will introduce the features, block diagram, and signal descriptions of RK27xx and the chapter 3 through chapter 24 will describe the full function of each module in detail. The ARM7EJC detail specification and instruction set refer to ARM's Technical Manual.

1.2 Features

- **System Operation**
  - Dual Core Architecture (ARM7EJC+DSP)
  - Support system boot sequentially from ARM7EJC to DSP
  - Selectable JTAG debug method
    - ARM7EJC debug only
    - DSP debug only
    - ARM7EJC+DSP dual core debug
  - Selectable booting method
    - Boot from NOR Flash
    - Boot from Embedded ROM
  - Internal memory space
    - DSP IMEM 32Kwords
    - DSP DMEM 32Kwords
    - ARM7EJC Embedded Sync SRAM 4Kbytes
    - ARM7EJC Embedded Boot ROM 8Kbytes

- **Clock & Power Management**
  - Use one 24MHz crystal oscillator
  - Use three PLL (For ARM/DSP/CODEC+HSADC)
  - Support different AHB Bus and ARM7EJC clock ratio
    - 1:1 and 1:2 mode
  - Max frequency of every domain
    - ARM7EJC 200MHz (default 12MHz)
    - AHB BUS 133MHz (default 12MHz)
    - APB BUS 133MHz (default 12MHz)
    - DSP 150MHz (default 24MHz)
  - Power management mode
    - Normal, slow, idle, stop, and power-off mode

- **Processor**
  - ARM7EJC
    - Max frequency 200MHz
    - ARMv5TE architecture with Jazella technology
    - 8x Java performance improvement
    - Five-stage pipeline
    - DSP extensions with 32x16 MAC
    - Small, cost-effective, processor macro cell
    - Unified Cache architecture , Size is 16KB(8words/line)
    - Two way configurable write-through cache
  - DSP
    - Max frequency 150MHz

- RISC-based four-way superscalar architecture
- Eight-stage pipeline
- Support 4GB address space

- Dual-Core communication unit
  - Support dual cores system
  - Mailbox handshaking with interrupt mechanism
    - Support two AHB slave ports
    - Each mailbox element includes one data word register, one command word register, and one flag bit that can represent one interrupt
    - 4 interrupts to Host Interrupt Controller
    - 4 interrupts to Client Interrupt Controller

- **Memory controller**
  - Static/SDRAM Memory controller
    - Dynamic memory interface support
    - Asynchronous static memory device support including SRAM, ROM and Flash with or without asynchronous page mode
    - Support 1 independent AHB slave port for control register, and up to 2 individual AHB slave port for data access
    - Support 1 chip selects for SDRAM and 2 chip selects for static memory
    - support 16-bit wide SDRAM and 8-bit or 16-bit wide static memory
    - Support industrial standard SDRAM from 16MBytes to 128MBytes devices
    - 16Mbytes access space per static memory support
  - NAND Flash controller
    - Support 4 chip selects for NAND flash
    - support 8-bit wide data
    - Flexible CPU interface support
    - Embedded 4x512bytes size buffer to improve performance
    - Support internal DMA transfer from/to flash
    - 512bytes、2Kbytes、4Kbytes page size support
    - Hardware ECC
  - SD controller
    - Compliant with SD spec. Version 1.01 except SPI mode
    - Compliant with Multimedia Card system specification V3.3 except SPI mode, Stream R/W mode and I/O (Interrupt) mode
    - Variable SD/MMC card clock rate 0 – 25 MHz which depends on APB clock frequency
    - Controllable SD/MMC card clock to save power consumption
    - Support MMU ping-pong structure to enhance the SD/MMC data transfer performance
    - Support DMA transfer

- **Sensor controller**
  - Support 24MHz 、48MHz、27MHz clock input
  - Support CCIR656 PAL/NTSC
  - Support YUYV and UYVY format input
  - Support YUV 4:2:2 and YUV 4:2:0 format output
  - Support up to 3.0 Mega Pixels sensor input
- **LCD controller**
  - Support video data input format
    - 24bit RGB
    - 16bit RGB
    - YUV 4:2:2
    - YUV 4:2:0
  - Support serial 8bit RGB LCD panel with dummy data or not
  - Support Parallel 24bit (max) RGB LCD panel
  - Built-in 2 640x32bit buffer
  - Support DMA transfer
- **DMA Controller**

- 3 DMA Controller on-chip
- AHB DMA (HDMA)
  - Integrated in AHB BUS0
  - Two DMA channels support
  - 8 hardware request handshaking support
  - Built-in 32x16 data FIFO
  - Support hardware and software trigger DMA transfer mode
- AHB-to-AHB Bridge (A2A)
  - Integrated between AHB BUS0 and AHB BUS1
  - Provide AHB-to-AHB bus protocol translation
  - Support AHB-to-AHB DMA or Single AHB DMA
  - Two DMA Channels support
  - On-the-fly mode between two level bus support
  - 4 hardware request handshaking support
  - Support hardware and software trigger DMA transfer mode
- DW DMA
  - Integrated in AHB BUS1
  - Four DMA Channels support
  - 8 hardware request handshaking support
  - Support hardware and software trigger DMA transfer mode
  - Build-in 4 data FIFO : 64bytes/64bytes/64bytes/64bytes
  - Scatter/Gather transfer support
  - LLP transfer support
  - Two master for on-the-fly support

- **USB interface**
  - UHC(USB HOST Controller) and UDC(USB DEV Controller) share PHY
  - Only support UHC or UDC work at the same time
  - USB2.0 Host Controller
    - USB 2.0 high speed host controller and specification compliant
    - Intel™ EHCI host controller interface specification compliant
    - OHCI host controller interface specification compliant
    - Supports USB Transceiver Macro cell Interface+ (UTMI+ level 3)
  - USB2.0 Device Controller
    - Complies with the Universal Serial Bus specification Rev. 2.0, Supports USB Full Speed (12Mb/sec) and High Speed (480 Mb/sec), is Backward compatible with USB1.1
    - On-chip USB2.0 PHY and Parallel Bus Interface Engine (PIE)
    - Suspend / Resume operation - Supports USB remote wake-up

- **Low_speed Peripheral interface**
  - Serial Peripheral Interface (SPI) Master Controller
    - Four transfer protocols available with selectable clock polarity and clock phase
    - Different bit rates available for SCLK
    - Bi-direction mode
  - UART (16550)
    - 2 UART support
    - UART0 support modem function
    - Supports up to 3Mbps baud-rate
    - 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).
  - I2C controller
    - Multi masters operation
    - Software programmable clock frequency and transfer rate up to 400Kbit/sec
    - Supports 7 bits and 10 bits addressing modes
  - I2S
    - 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

- PWM
  - Built-in three 32 bit timer modulers
  - Programmable counter
  - Chained timer for long period purpose
  - 4-channel 32-bit timer with Pulse Width Modulation (PWM)
  - Programmable duty-cycle, and frequency output
- General Purpose IO (GPIO)
  - Support 48 individually programmable input/output pins
  - Support GPIO with interrupt capability
- Timers
  - Built-in three 32 bits timer modules
  - Programmable counter
- Watchdog Timer (WDT)
  - Watchdog function
  - Built-in 32 bits programmable reset counter
- Real Time Clock (RTC)
  - Support perpetual RTC core power
  - Programmable alarm with interrupt for system power wake up
  - System power off sequence with output control pin
  - RTC core power loss indication

- Analog IP interface
  - ADC Converter
    - 4-channel single-ended 10-bit 1MSPS Successive Approximation Register (SAR) analog-to-digital converter
    - Supply 2.5V to 3.6V for analog and 0.9V to 1.3V for digital interface
    - Very Low Power : <0.4mW power consumption at 1MSPS
  - CODEC
    - Support built-in codec or external codec through I2S interface
    - Build in Stereo 24-bit Delta-Sigma DAC with on-chip headphone amplifier
    - Build in Stereo 16-bit Sigma-Delta ADC

- **Operation Frequency**
  - ARM7: up to 200MHz
  - DSP : up to 150MHz
  - AMBA AHB bus: up to 133 MHz
  - AMBA APB bus: up to 66 MHz

- **Package**
  - RK2706 LQPG128
  - RK2708/RK2710 LQFP144
  - BGA169

- **Operation Temperature Range**
  - 0°C to +125°C

- **Operation Voltage Range**
  - Core: 1.2 V
  - I/O : 3.3V

1.3 Pin Diagram

1.4 Pin Description

RK2706	RK2708	RK2710			
PIN128	PIN144	PIN144	Names	Direction	PIN Description
1	1	1	NPOR	I Pull up	Power on Reset
2	2	2	SDT_A6	O	SDRAM/SRAM addr[6]
3	3	3	SDT_A5	O	SDRAM/SRAM addr[5]
4	4	4	SDT_A4	O	SDRAM/SRAM addr[4]
5	5	5	SDT_A3	O	SDRAM/SRAM addr[3]
6	6	6	SDT_A2	O	SDRAM/SRAM data[2]
7	7	7	SDT_A1	O	SDRAM/SRAM data[1]
8	8	8	SDT_A0	O	SDRAM/SRAM data[0]
9	9	9	LCD_D0	O	lcd data[0]
10	10	10	LCD_D1	O	lcd data[1]
11	11	11	LCD_D2	O	lcd data[2]
12	12	12	LCD_D3	O	lcd data[3]
13	13	13	VDDIO	P	IO POWER SUPPLY 3.3V
14	14	14	VSS	P	GROUND
15	15	15	XIN24M	I OSC	Crystal 24M input
16	16	16	XOUT24M	O OSC	Crystal 24M output
17	17	17	VDD	P	CORE POWER SUPPLY 1.2V
18	18	18	LCD_D4	O	lcd data[4]
19	19	19	LCD_D5	O	lcd data[5]
20	20	20	LCD_D6	O	lcd data[6]
21	21	21	LCD_D7	O	lcd data[7]
22	22	22	PE0/LCD_D8	IO Pull up/O	GPIO E0/LCD data[8]
23	23	23	PE1/LCD_D9	IO Pull up/O	GPIO E1/LCD data[9]
24	24	24	PE2/LCD_D10	IO Pull up/O	GPIO E2/LCD data[10]
25	25	25	PE3/LCD_D11	IO Pull up/O	GPIO E3/LCD data[11]
26	26	26	PE4/LCD_D12	IO Pull up/O	GPIO E4/LCD data[12]
27	27	27	PE5/LCD_D13	IO Pull up/O	GPIO E5/LCD data[13]
28	28	28	PE6/LCD_D14	IO Pull up/O	GPIO E6/LCD data[14]
29	29	29	PE7/LCD_D15	IO Pull up/O	GPIO E7/LCD data[15]
30	30	30	PA0/LCD_D16/RXD0	IO Pull up/O/O	GPIO A1/LCD data[16]/UART0 rxd
31	31	31	PA1/LCD_D17/TXD0	IO Pull up/O/O	GPIO A1/LCD data[17]/UART0 txd
32	32	32	LCD_CLK/LCD_RS	O	RGB dot clock / MUC pannel RS
X	33	33	VIP_D0/HADC_D0	I pull up	HS adc data[0]
X	34	34	VIP_D1/HADC_D1	I pull up	HS adc data[1]
X	35	35	VIP_D2/HADC_D2	I pull up	HS adc data[2]
X	36	36	VIP_D3/HADC_D3	I pull up	HS adc data[3]
X	37	37	VIP_D4/HADC_D4	I pull up	VIP data[0]
X	38	38	VIP_D5/HADC_D5	I pull up	VIP data[1]
X	39	39	VIP_D6/HADC_D6	I pull up	VIP data[2]
X	40	40	VIP_D7/HADC_D7	I pull up	VIP data[3]
33	41	41	LCD_HSYNC	O	RGB HSYNC/MCU_WR
34	42	42	PA7/LCD_VSYNC	IO Pull up/O	GPIO

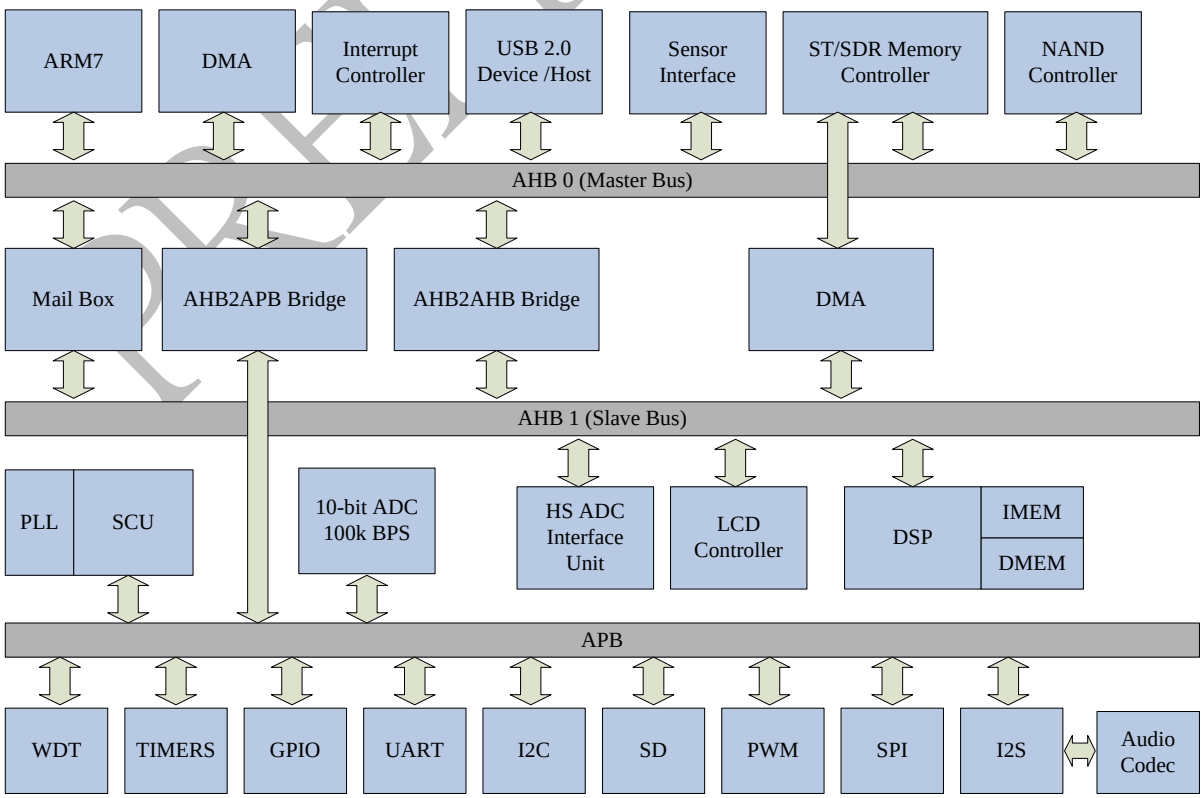
					A7/RGB_VSYNC/MCU_CS
35	43	43	FLASH_RDY	I pull up	NAND FLASH R/B
36	44	44	FLASH_RDN	O	NAND FLASH RD
37	45	45	FLASH_CS0	O	NAND FLASH CS0
38	46	46	FLASH_D7	B	nand flash data[7]
39	47	47	FLASH_D6	B	nand flash data[6]
40	48	48	FLASH_D5	B	nand flash data[5]
41	49	49	FLASH_D4	B	nand flash data[4]
42	50	50	FLASH_D3	B	nand flash data[3]
43	51	51	FLASH_D2	B	nand flash data[2]
44	52	52	FLASH_D1	B	nand flash data[1]
45	53	53	FLASH_D0	B	nand flash data[0]
46	54	54	FLASH_CLE	O	nand flash cle
47	55	55	FLASH_ALE	O	nand flash ale
48	56	56	FLASH_WRN	O	NAND FLASH WR
49	57	57	VDD	P	CORE POWER SUPPLY 1.2V
50	58	58	VSS	P	GROUND
51	59	59	VDDIO	P	IO POWER SUPPLY 3.3V
52	60	60	PB2/SDDATA0/SPI_MISO	IO Pull up/B/B	GPIO B2/SD DATA OUT/SPI_MISO
53	61	61	PB3/SDCMD/SPI_MOSI	IO Pull up/B/B	GPIO B3/SD DATA IN/SPI_MOSI
54	62	62	PB5/SDCLK/SPI_CLK	IO Pull up/O/O	GPIO B3/SD CLK/SPI_CLK
55	63	63	CODEC_AIL1	AI	L-channel analog input 1
56	64	64	CODEC_AIR1	AI	R-channel analog input 1
57	65	65	CODEC_MIC	AI	Mic input
58	66	66	CODEC_VCOM	AO	Internal biasing voltage
59	67	67	CODEC_VSSA	P	Ground for Codec
60	68	68	CODEC_VDDA	P	Power supply for CODEC, 3.3V
61	69	69	CODEC_AOHPL	AO	L-channel headphone output
62	70	70	CODEC_VSSAO	P	Ground for amplifiers
63	71	71	CODEC_VDDAO	P	Power supply for amplifiers 3.3V
64	72	72	CODEC_AOHPR	AO	R-channel headphone output
65	73	73	PA5/FLASH_CS1	IO Pull up/O	GPIO A5/FLASH CS1
66	74	74	SDA/FLASH_CS3/PB7	IO Pull up/O/IO Pull up	SCL/NAND FLASH CS3/GPIO B7
67	75	75	SCL/FLASH_CS2/PB6	IO Pull up/O/IO Pull up	SCL/NAND FLASH CS2/GPIO B6
68	76	76	PC0	IO Pull down	GPIO C0
69	77	77	PC1/LCD_DEN	IO Pull down/O	GPIO C1/RGB DEN
70	78	78	PC2/I2S_SCLK	IO Pull down/B	GPIO C2/I2S SCLK
71	79	79	PC3/I2S_LRCK	IO Pull down/B	GPIO C3/I2S LRCK
72	80	80	PC4/I2S_SDI	IO Pull down/I	GPIO C4/I2S DATA IN
73	81	81	PC5/I2S_SDO	IO Pull down/O	GPIO C5/I2S DATA OUT
74	82	82	PC6/I2SCLK	IO Pull down/O	GPIO C6/I2S CLOCK OUT
75	83	83	PC7/ST_CSN1	IO Pull up/O	GPIO C7/static memory CS1
X	84	84	PF0/VIP_CLKO	IO pull up/O	GPIO F7/VIP clock out

X	85	X	PD0/SDPCA/TXD1	IO Pull up/O/O	GPIO D0/SD Power control/Uart1 txd
X	86	X	PD1/SDCDA/RXD1	IO Pull up/O/O	GPIO D1/SD detect/Uart1 rxd
X	87	85	PD3/SD_CKE	IO Pull up/O	GPIO D3/SDRAM CKE
X	X	86	PD6/PWM2	IO Pull down/O	GPIO D6/PWM2
X	X	87	PD5/PWM1	IO Pull down/O	GPIO D5/PWM1
76	88	88	PD4/PWM0	IO/O	GPIO D4/PWM0
77	89	89	VDDIO	P	IO POWER SUPPLY 3.3V
78	90	90	VSS	P	GROUND
79	91	91	VDD	P	CORE POWER SUPPLY 1.2V
80	92	92	SDT_D15	B	SDRAM/SRAM data[15]
81	93	93	SDT_D14	B	SDRAM/SRAM data[14]
82	94	94	SDT_D13	B	SDRAM/SRAM data[13]
83	95	95	SDT_D12	B	SDRAM/SRAM data[12]
84	96	96	SDT_D11	B	SDRAM/SRAM data[11]
85	97	97	VBUS_DET	I Pull down	USB VBUS detect
86	98	98	LADC_AIN0	A	10bit adc channel0 input
87	99	99	LADC_AIN1	A	10bit adc channel1 input
88	100	100	LADC_AIN2	A	10bit adc channel2 input
89	101	101	LADC_VSSA	P	10bit adc analog ground
90	102	102	PHY_VDDA	P	USB PHY Analog Power 3.0V-3.3V
91	103	103	PHY_DN	A	USB DN
92	104	104	PHY_DP	A	USB DP
93	105	105	PHY_VSSA	P	USB PHY Analog Ground
94	106	106	PHY_REF	A	Resistor for USB，6.04K 1%
95	107	107	PHY_VDDP	P	USB PHY PLL POWER，1.2V
96	108	108	PHY_VSSP	P	USB PHY PLL Ground
97	109	109	SDT_D10	B	SDRAM/SRAM data[10]
98	110	110	SDT_D9	B	SDRAM/SRAM data[8]
99	111	111	SDT_D8	B	SDRAM/SRAM data[8]
100	112	112	SDT_D7	B	SDRAM/SRAM data[7]
101	113	113	SDT_D6	B	SDRAM/SRAM data[6]
102	114	114	SDT_D5	B	SDRAM/SRAM data[5]
103	115	115	SDT_D4	B	SDRAM/SRAM data[4]
104	116	116	SDT_D3	B	SDRAM/SRAM data[3]
105	117	117	SDT_D2	B	SDRAM/SRAM data[2]
106	118	118	SDT_D1	B	SDRAM/SRAM data[1]
107	119	119	SDT_D0	B	SDRAM/SRAM data[0]
108	120	120	SD_DQM1	O	SDRAM dqm[1]
109	121	121	SD_DQM0	O	SDRAM dqm[0]
110	122	122	SD_WEN	O	SDRAM wen
111	123	123	VDDIO	P	IO POWER SUPPLY 3.3V
112	124	124	VSS	P	GROUND
113	125	125	VDD	P	CORE POWER SUPPLY 1.2V
114	126	126	SD_CASN	O	SDRAM casn
115	127	127	SD_RASN	O	SDRAM rasn
116	128	128	SD_CLK	O	SDRAM clkout

117	129	129	SD_CSN	O	SDRAM csn
118	130	130	SD_BA0	O	SDRAM ba[0]
119	131	131	SD_BA1	O	SDRAM ba[1]
120	132	132	SDT_A12/PF2	O/IO Pull up	SDRAM/SRAM addr[12]/GPIO F2
121	133	133	SDT_A11/PF1	O/IO Pull up	SDRAM/SRAM addr[11]/GPIO F1
122	134	134	SDT_A10	O	SDRAM/SRAM addr[10]
123	135	135	SDT_A9	O	SDRAM/SRAM addr[9]
124	136	136	SDT_A8	O	SDRAM/SRAM addr[8]
125	137	137	SDT_A7	O	SDRAM/SRAM addr[7]
X	138	138	TEST	I PullDown	Test mode
X	139	X	VIP_PCLK	I pullup	VIP pclk
X	140	139	VIP_VSYNC/HADC_D9	I pullup	VIP vsync
X	141	140	VIP_HSYNC/HADC_D8	I pullup	VIP hsync
X	X	141	HADC_CLK	O	hsadc clk output
126	142	142	ZPLL_VDDA	P	DSP PLL power 1.2V
127	143	143	PLL_VSSA	P	DSP PLL Ground
128	144	144	APLL_VDDA	P	ARM PLL power 1.2V
NOTE:	**P**		**power supply pad**		
	IO		**IO pad**		
	O		**output pad**		
	I		**input pad**		

1.5 Architercture

1.5.1 Block Diagram



Chapter 2 System Configuration

2.1 Descriptions

2.1.1 AHB Master Priority

AHB Arbiter decides the priority of AHB masters. The RK27xx provides two bus arbiters on each AHB bus (AHB0 and AHB1). The lower master has higher priority when their priority levels are set to a same value. For detail arbiter control register setting, please reference AHB bus arbiter chapter.

The following table describes the priority of IPs on each BUS.

Priority of AHB_0 Masters

Master No.	Priority	Block
5	Highest	VIP
6		UHC
7		UDC
10		HDMA
14		A2A&DMA
15	Lowest	ARM7EJ

Priority of AHB_1 Masters

Master No.	Priority	Block
1	Highest	A2A
14		DWDMA
15	Lowest	DSP / DCM

2.1.2 Interrupts

The RK27xx provides an interrupt controller for ARM7EJC processor, which has 32 interrupt sources for internal blocks or external devices usage , and generates one interrupt request IRQ to ARM7EJC. Each interrupts trigged type can be configured by software. For detail interrupt controller setting, please reference interrupt controller chapter. Another , for DSP processor , 13 external interrupts are reserved for user, which are always edge sensed , and must remain asserted high for at least one clock period.

The following table describes interrupt source number for two processors. The following table describes each interrupt controller's source number.

INTC0 (ARM Interrupts)

Source #	Source Description	Polarity
31 (High)	Software Interrupt	-
30	Software Interrupt	-
29	Software Interrupt	-
28	DSP	High level
27	LCDC	High level
26	NANDC	High level
25	DWDMA	High level
24	VIP	High level
23	GPIO1 (a/b group)	High level
22	ADC	High level
21	PWM3	High level
20	PWM2	High level
19	PWM1	High level
18	PWM0	High level

17	UHC	High level
16	UDC	High level
15	I2S	High level
14	I2C	High level
13	A2A bridge & DMA	High level
12	HDMA	High level
11	SPI master controller	High level
10	SD	High level
9	SCU	High level
8	RTC	High level
7	AHB0 MAILBOX	High level
6	SoftWare Interrupt	-
5	GPIO0 (a/b group)	High level
4	Timer0_2	High level
3	Timer0_1	High level
2	Timer0_0	High level
1	UART1	High level
0 (Low)	UART0	High level

INTC1 (DSP interrupts)

Source #	Source Description	Polarity
nmi	ARM7EJC	High level
13	Software Interrupt	High level
12	Software Interrupt	High level
11	PWM0	High level
10	GPIO0 (c/d group)	High level
9	HS_ADC	High level
8	LCDC	High level
7	VIP	High level
6	Timer1 (internal)	High level
5	Timer0 (internal)	High level
4	AHB1 MAILBOX	High level
3	DWDMA	High level
2	A2A bridge & DMA	High level
1	HDMA	High level
0	UHC or UDC (combinational)	High level

2.1.3 DMA hardware request

The RK27xx provides 3 DMA controllers , total 20 hardware dma request interfaces, which have different request polarity requirements.
The following table describes DMA hardware request connection.

HDMA

Source #	Source Description	Polarity
0	UART0 rxd	LOW level
1	UART0 txd	LOW level
2	UART1 rxd	LOW level
3	UART1 txd	LOW level
4	SPI rxd	LOW level
5	SPI txd	LOW level
6	I2S txd	LOW level
7	I2S rxd	LOW level

A2A bridge &DMA

Source #	Source Description	Polarity
0	SD/MMC	LOW level
1	Reserved	-
2	Reserved	-
3	Reserved	-

DWDMA

Source #	Source Description	Polarity
0	HS_ADC	High level
1	Reserved	High level
2	Reserved	High level
3	Reserved	High level
4	LCDC line, for Y channel	High level
5	LCDC line backup, for UV channel	High level
6	LCDC blank, for Y&UV channel	High level
7	LCDC FIFO, for RGB channel	High level

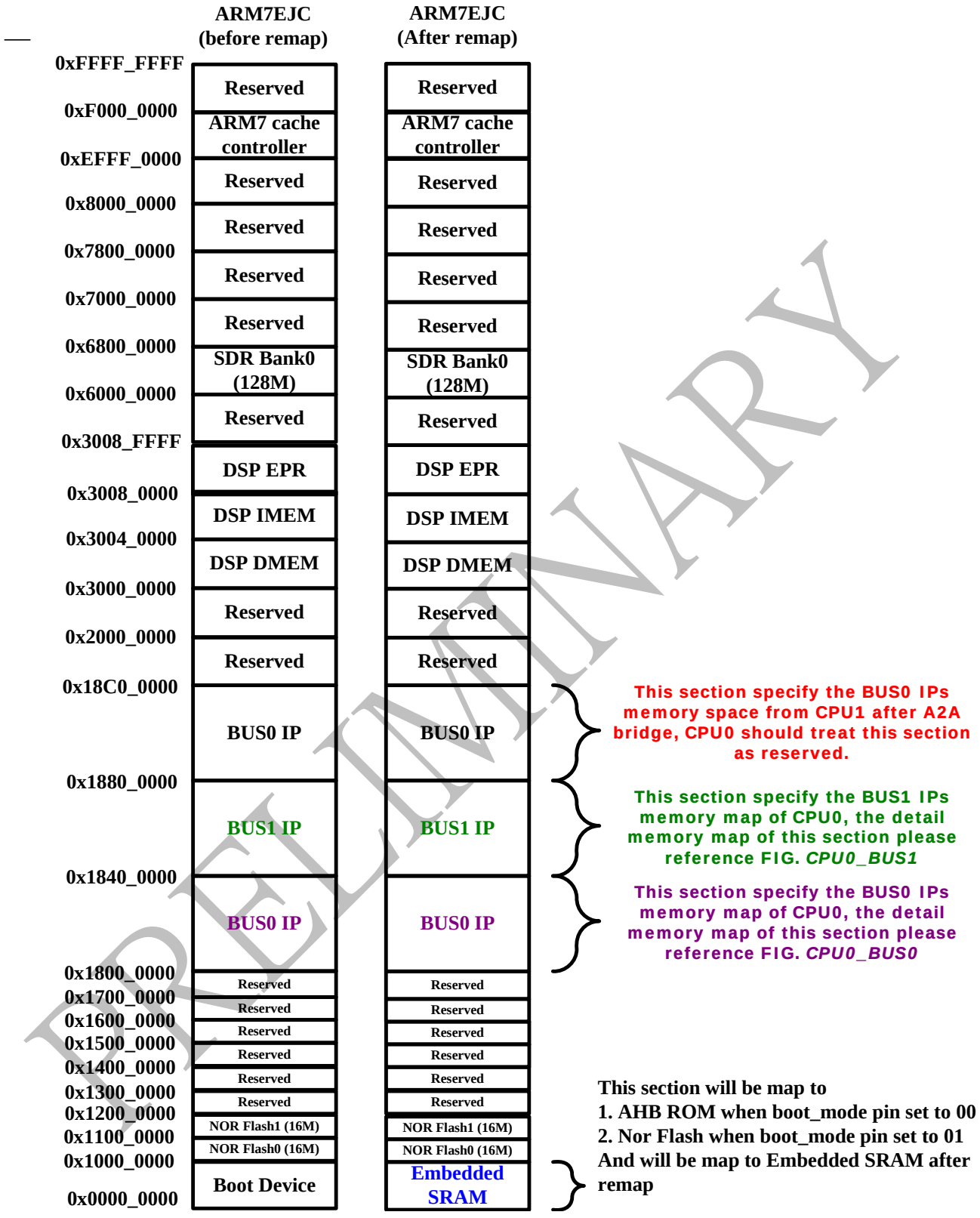
2.2 System Address Map

2.2.1 Default Memory Map

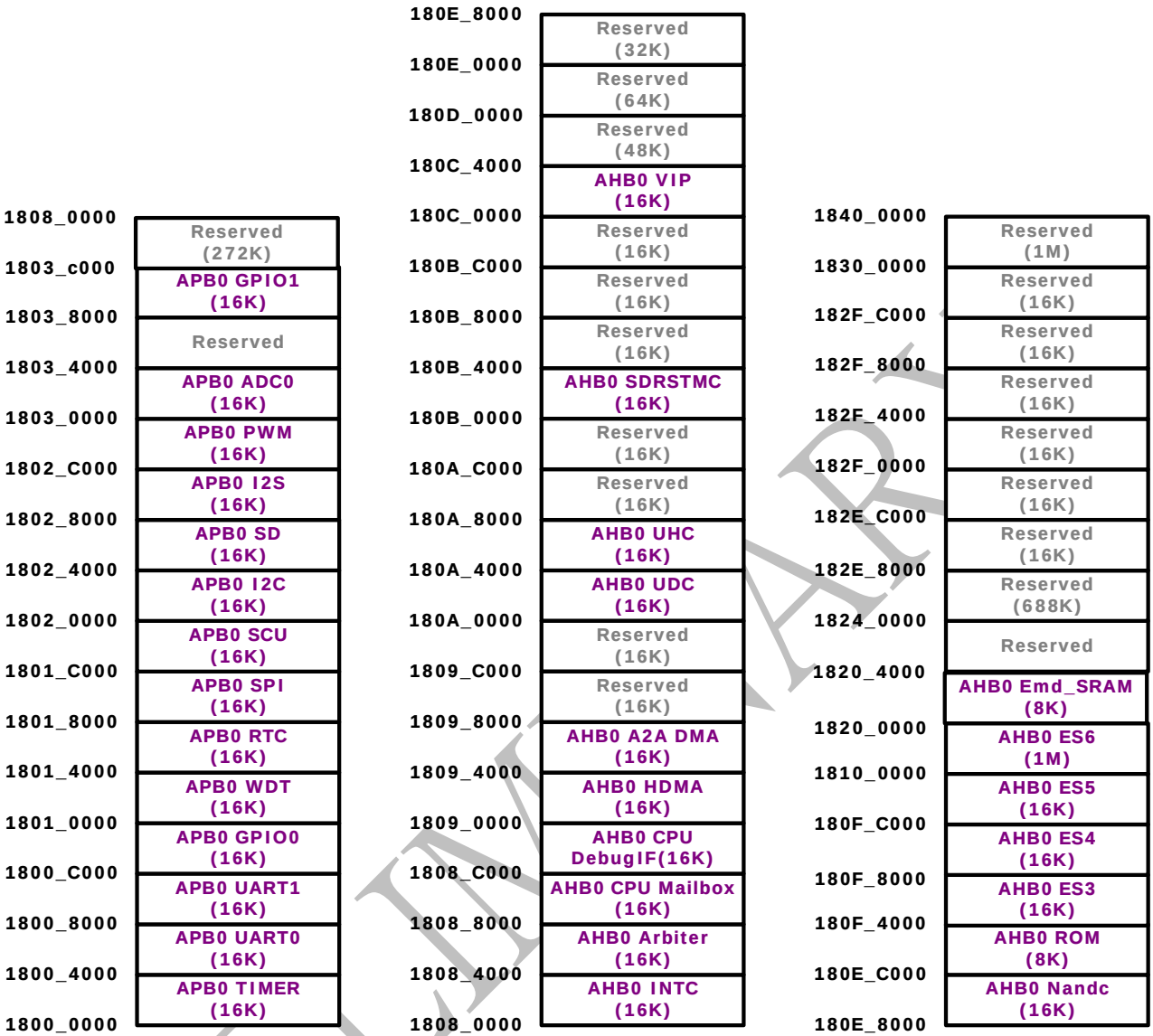
The 32-bit address bus can address up to 4GB of memory. This space is sub-divided into a numbers of memory banks and control registers. The RK27xx uses little endian only for memory and registers access.

For ARM processor, RK27xx core provides address decode re-map function. It cans speed-up whole system performance. The detail memory map is as follow.

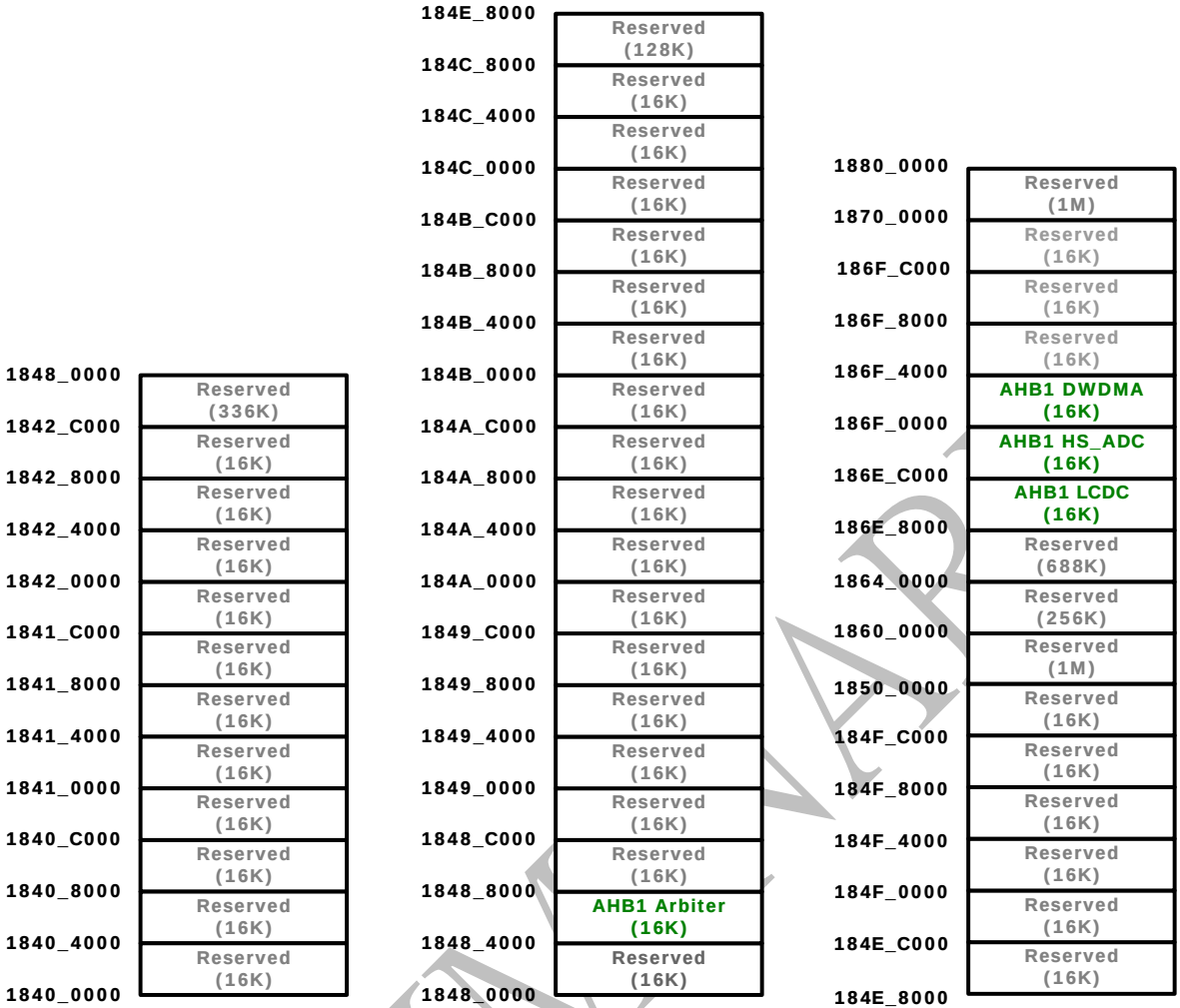
2.2.2 System Memory Map for ARM



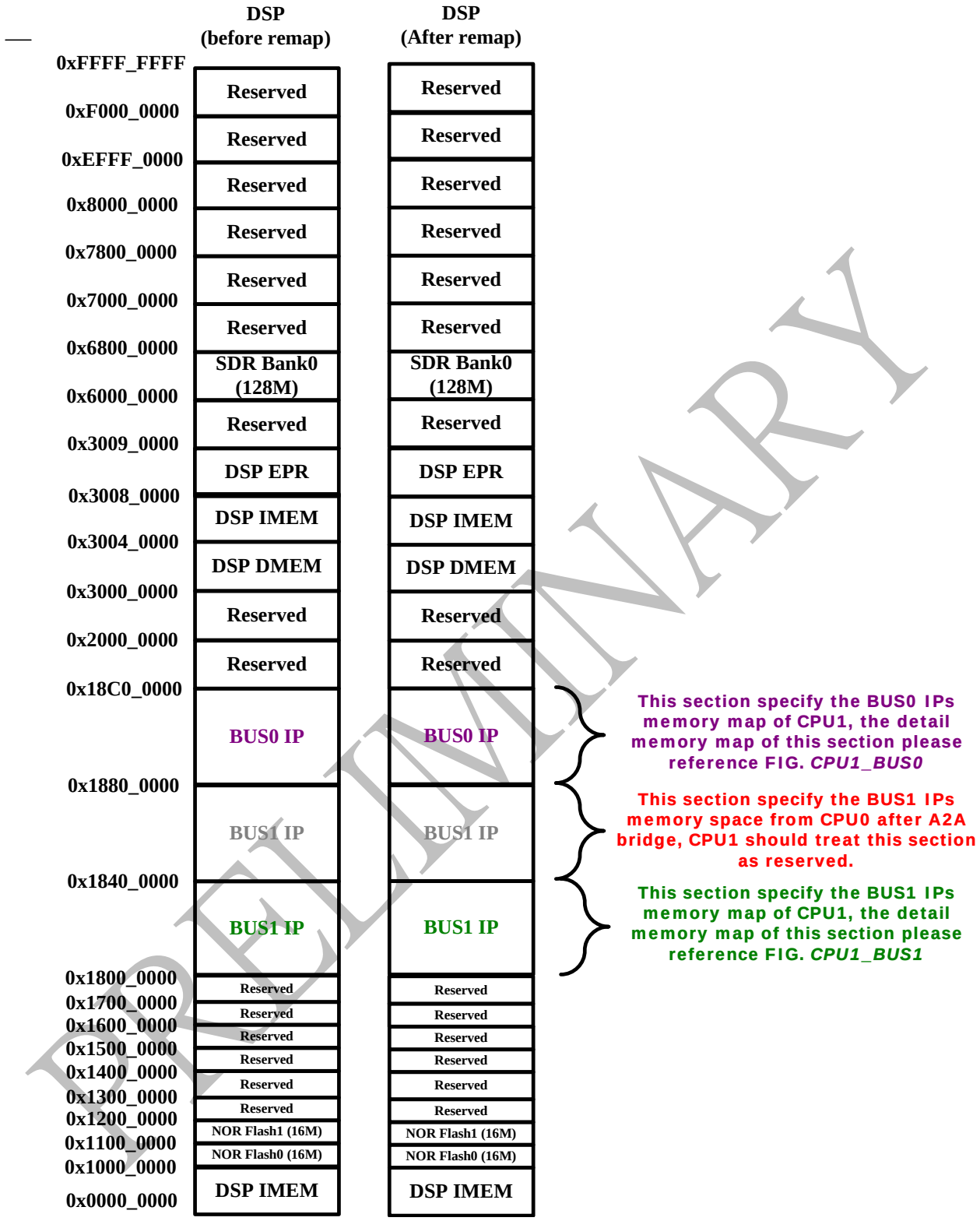
- **FIG. ARM_BUS0**



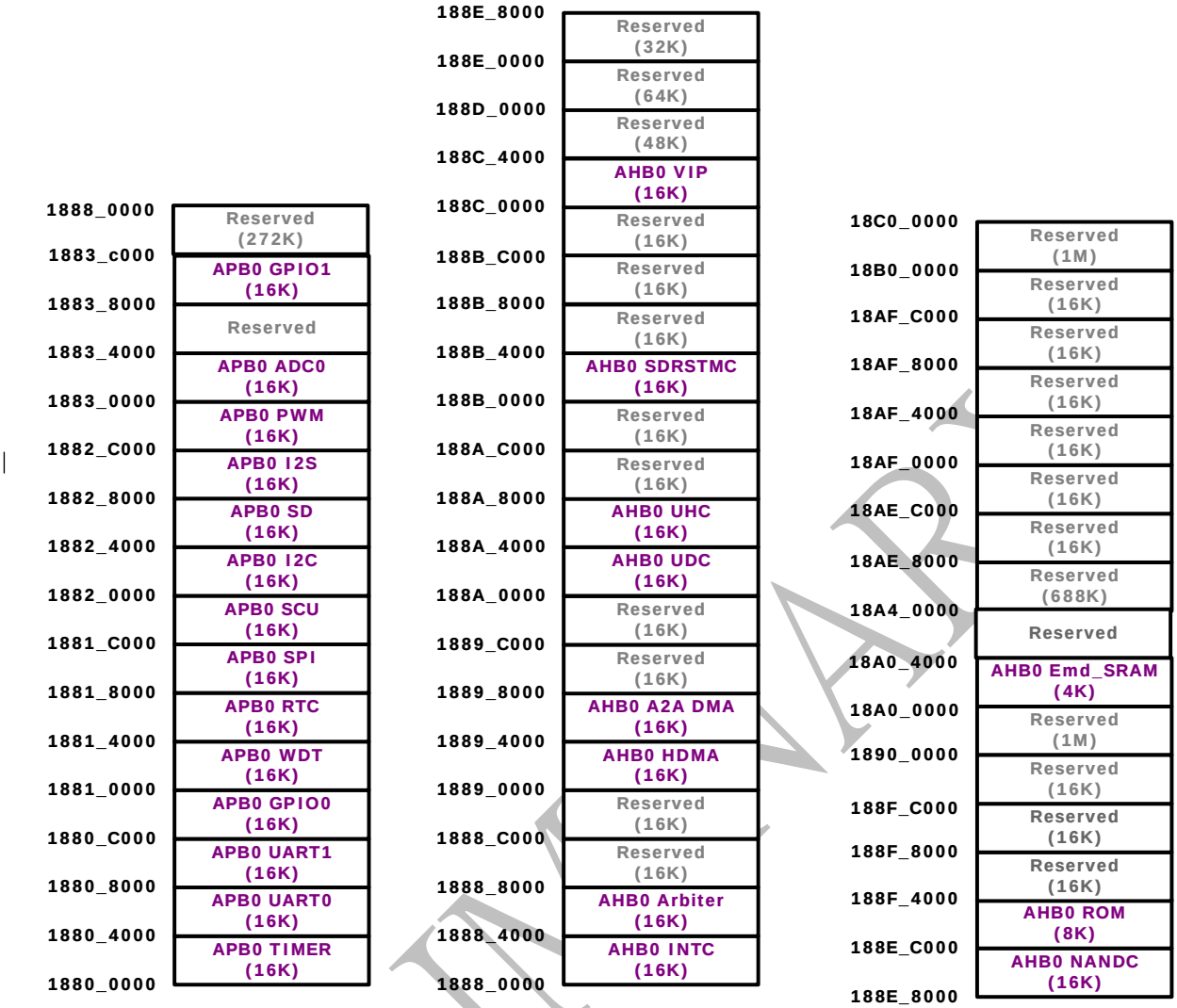
- **FIG. ARM_BUS1**



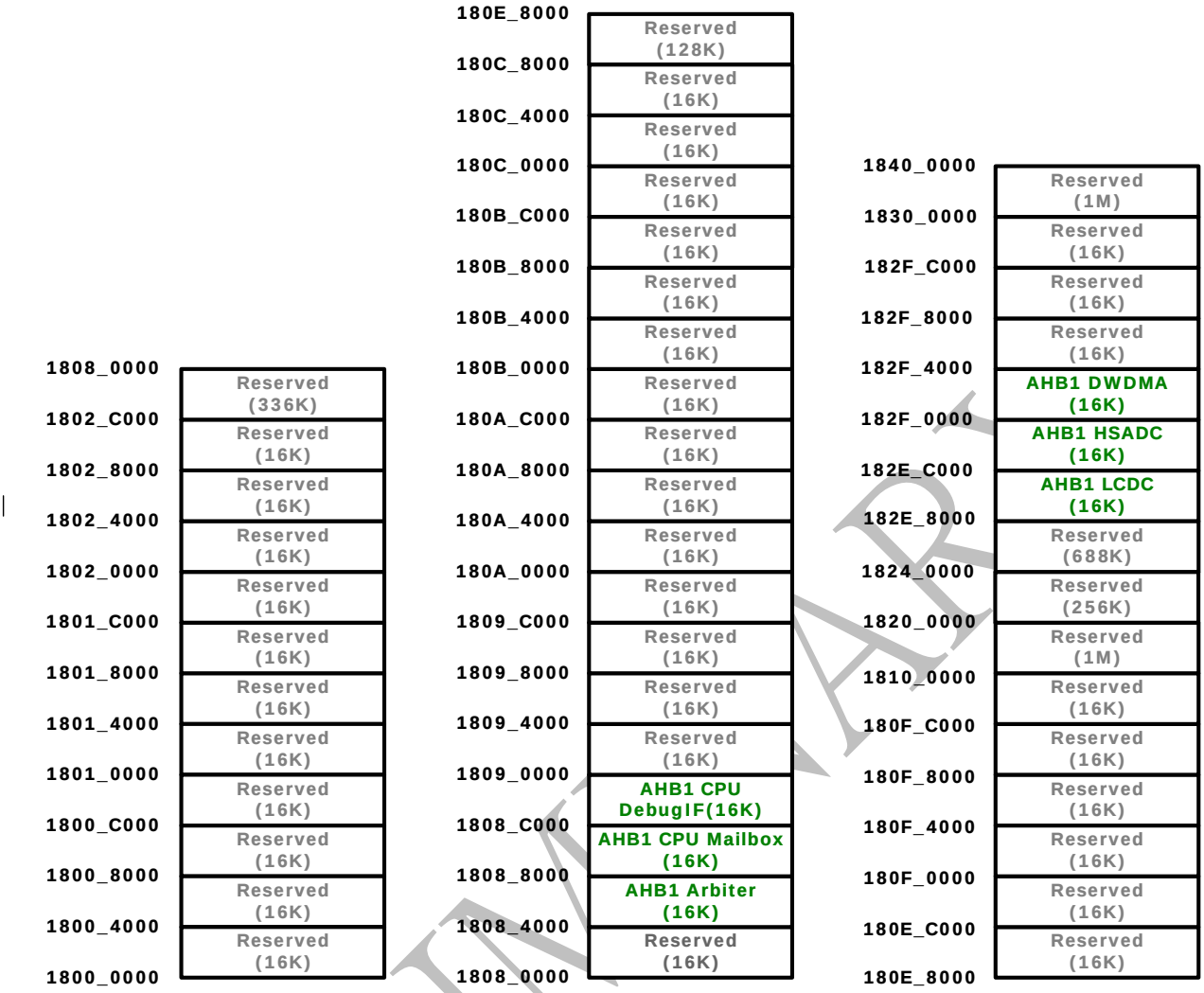
2.2.3 System Memory Map for DSP



- **FIG. DSP_BUS0**



- **FIG. DSP_BUS1**



2.3 System mode configuration

2.3.1 Debug mode

	OP_MODE[1:0]	Description
OP_MODE0	2'b00	ARM7
OP_MODE1	Reserved	Reserved
OP_MODE2	2'b10	ARM7 + DSP
OP_MODE3	2'b11	DSP

2.3.2 CPU Boot mode

	BT_MODE	Description
BOOT_MODE0	1'b0	Boot from Embedded AHB_ROM
BOOT_MODE1	1'b1	Boot from NOR Flash (SDTMC)

Chapter 3 Dual-Core Communication Unit

3.1 Design Overview

3.1.1 Overview

The Dual-Core Communication Unit contains 3 units, CPU Address Remap unit, communication unit - MAILBOX and Debug Interface unit - DBGIF. The CPU Address Remap unit provides processor address translation function that inter-processors will not conflict on execution the start address 0x00000000. The MAILBOX provides inter-processor an effective communication channel with the interrupt mechanism. The Debug Interface unit provides Dual-Core system debugging capability with one EJTAG on Multi-ICE of ARM.

3.1.2 Features

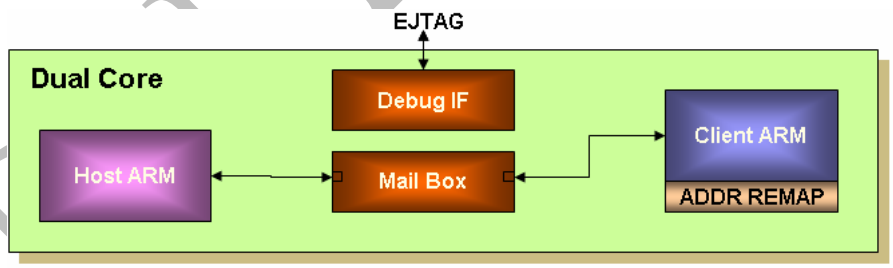
- Support dual cores system
- MAILBOX communication unit
- Support two AHB slave ports
- Each mailbox element includes one data word register, one command word register, and one flag bit that can represent one interrupt
- 4 interrupts to Host Interrupt Controller
- 4 interrupts to Client Interrupt Controller

3.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

The Dual-Core Unit includes one CPU Address Remap unit and communication unit MAILBOX.

3.2.1 Block Diagram

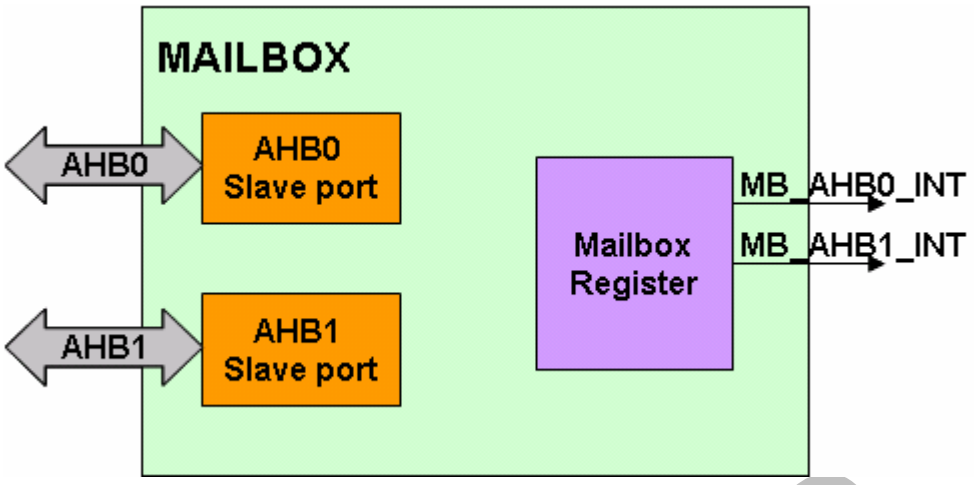


3.2.2 CPU_ADDR_REMAP Descriptions

CPU_ADDR_REMAP– CPU Address Remap

The CPU Address Remap implements CPU address translation to support dual-core architecture. While booting process, both CPU copy its own ROM code to private memory area. After booting process done, CPU begins to execute its task. But both CPUs' execution address starts from 0x00000000, the same address. They will point to the same ROM code. CPU address remap will let CPU's address to map to different memory block to support dual-core system.

3.2.3 MAILBOX Block Diagram



3.2.4 MAILBOX Block Descriptions

MAILBOX – Dual-Core communication unit

The MAILBOX implements 2 AHB slave ports that are compliance with AHB2.0. The two slave ports 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. The MAILBOX have 4 Host2Client MAILBOX elements that can generate 4 interrupts to Client Vectored Interrupt Controller and 4 Client2Host MAILBOX elements that can generate 4 interrupts to Client Vectored Interrupt Controller. It operates synchronously with the HCLK clock from the AHB bus.

3.3 Registers

3.3.1 MAILBOX Registers Summary

Name	Offset	Size	Reset Value	Description
MAILBOX_ID	0x0000	R	0x00000000	MAILBOX AHB Bus ID
H2C_STA	0x0010	RW	0x00000000	Host CPU to client CPU MAILBOX status
H2C0	0x0020	RW	0x00000000	Data word for the H2C0 MAILBOX
H2C0b	0x0024	RW	0x00000000	Command word for the H2C0 MAILBOX
H2C1	0x0028	RW	0x00000000	Data word for the H2C1 MAILBOX
H2C1b	0x002C	RW	0x00000000	Command word for the H2C1 MAILBOX
H2C2	0x0030	RW	0x00000000	Data word for the H2C2 MAILBOX
H2C2b	0x0034	RW	0x00000000	Command word for the H2C2 MAILBOX
H2C3	0x0038	RW	0x00000000	Data word for the H2C3 MAILBOX
H2C3b	0x003C	RW	0x00000000	Command word for the H2C3 MAILBOX
C2H_STA	0x0040	RW	0x00000000	Client CPU to host CPU MAILBOX status

C2H0	0x0050	RW	0x00000000	Data word for the C2H0 MAILBOX
C2H0b	0x0054	RW	0x00000000	Command word for the C2H0 MAILBOX
C2H1	0x0058	RW	0x00000000	Data word for the C2H1 MAILBOX
C2H1b	0x005C	RW	0x00000000	Command word for the C2H1 MAILBOX
C2H2	0x0060	RW	0x00000000	Data word for the C2H2 MAILBOX
C2H2b	0x0064	RW	0x00000000	Command word for the C2H2 MAILBOX
C2H3	0x0068	RW	0x00000000	Data word for the C2H3 MAILBOX
C2H3b	0x006C	RW	0x00000000	Command word for the C2H3 MAILBOX

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

3.3.2 MAILBOX Detail Register Description

MAILBOX_ID
Address: Operational Base + offset(0x00)
MAILBOX AHB Bus ID

bit	Attr	Reset Value	Description
31:1	R	0x00000000	Reserved
0	R	0x00000000	If read from AHB0, MAILBOX_ID[0]=0x0. If read from AHB1, MAILBOX_ID[0]=0x1.

H2C_STA
Address: Operational Base + offset(0x10)
Host CPU to client CPU mailbox status

bit	Attr	Reset Value	Description
31:4	R	0x00000000	Reserved
3	R	0x00000000	H2C3_FLAG Indicate that the H2C3 mailbox interrupt has been generated. Set by host CPU writing to H2C3b; cleared by client CPU reading of H2C3b. 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 H2C2b; cleared by client CPU reading of H2C2b. 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 H2C1b; cleared by client CPU reading of H2C1b. 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 H2C0b; cleared by client CPU reading of H2C0b. 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.

H2Cx (x = 0, 1, 2, 3)
Address: Operational Base + offset(0x20,0x28,0x30,0x38)
Data word for the H2C0 MAILBOX

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

H2Cx (x = 0, 1, 2, 3)
Address: Operational Base + offset(0x24,0x2C,0x34,0x3C)
Command word for the H2C0 MAILBOX

bit	Attr	Reset Value	Description
31:0	RW	0x00000000	Command word for the H2Cx MAILBOX. 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. When client CPU reads this register, the interrupt and flag are cleared.

C2H_STA
Address: Operational Base + offset(0x40)
Client CPU to host CPU mailbox status

bit	Attr	Reset Value	Description
31:4	R	0x00000000	Reserved
3	R	0x00000000	C2H3_FLAG Indicate that the C2H3 mailbox interrupt has been generated. Set by client CPU writing to C2H3b; cleared by host CPU reading of C2H3b. Before client CPU write C2H3 mailbox data word and command word, client CPU need to check if C2H3_FLAG is clear. Before host CPU read out C2H3 mailbox, host CPU need to check if C2H3_FLAG is set.
2	R	0x00000000	C2H2_FLAG Indicate that the C2H2 mailbox interrupt has been generated. Set by client CPU writing to C2H2b; cleared by host CPU reading of C2H2b. Before client CPU write C2H2 mailbox data word and command word, client CPU need to check if C2H2_FLAG is clear. Before host CPU read out C2H2 mailbox, host CPU need to check if C2H2_FLAG is set.
1	R	0x00000000	C2H1_FLAG Indicate that the C2H1 mailbox interrupt has been generated. Set by client CPU writing to C2H1b;

			cleared by host CPU reading of C2H1b. Before client CPU write C2H1 mailbox data word and command word, client CPU need 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 C2H0b; cleared by host CPU reading of C2H0b. 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.

C2Hx (x = 0, 1, 2, 3)
Address: Operational Base + offset (0x50, 0x58, 0x60, 0x58)
Data word for the C2H0 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.

C2Hxb (x = 0, 1, 2, 3)
Address: Operational Base + offset (0x54, 0x5C, 0x64, 0x6C)
Command word for the C2H0 MAILBOX

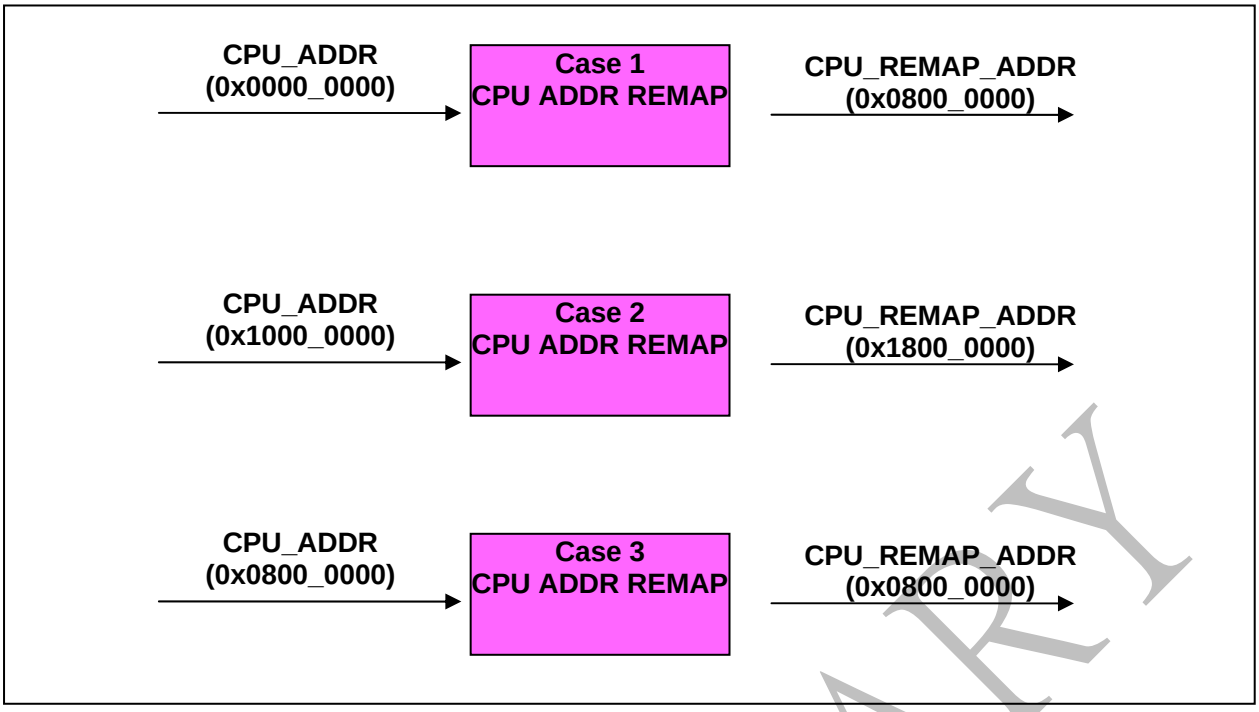
bit	Attr	Reset Value	Description
31:0	RW	0x00000000	Command word for the C2Hx MAILBOX. It can be written only by client CPU and read by host CPU. Writing to this register generates the C2Hx mailbox interrupt and sets C2Hx_FLAG. When host CPU reads this register, the interrupt and flag are cleared.

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

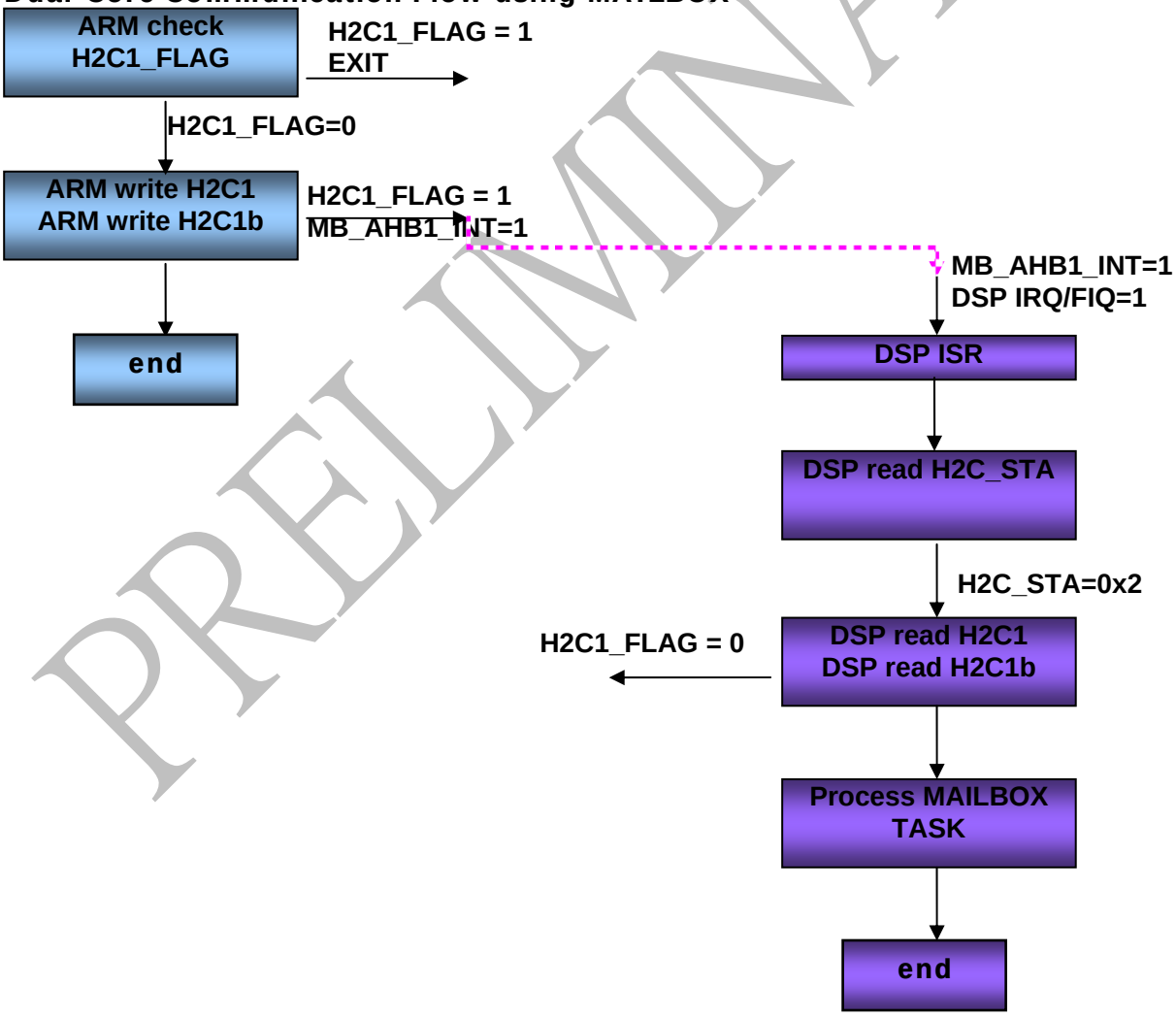
3.4 Function Description

3.4.1 Operation

CPU Address Remap



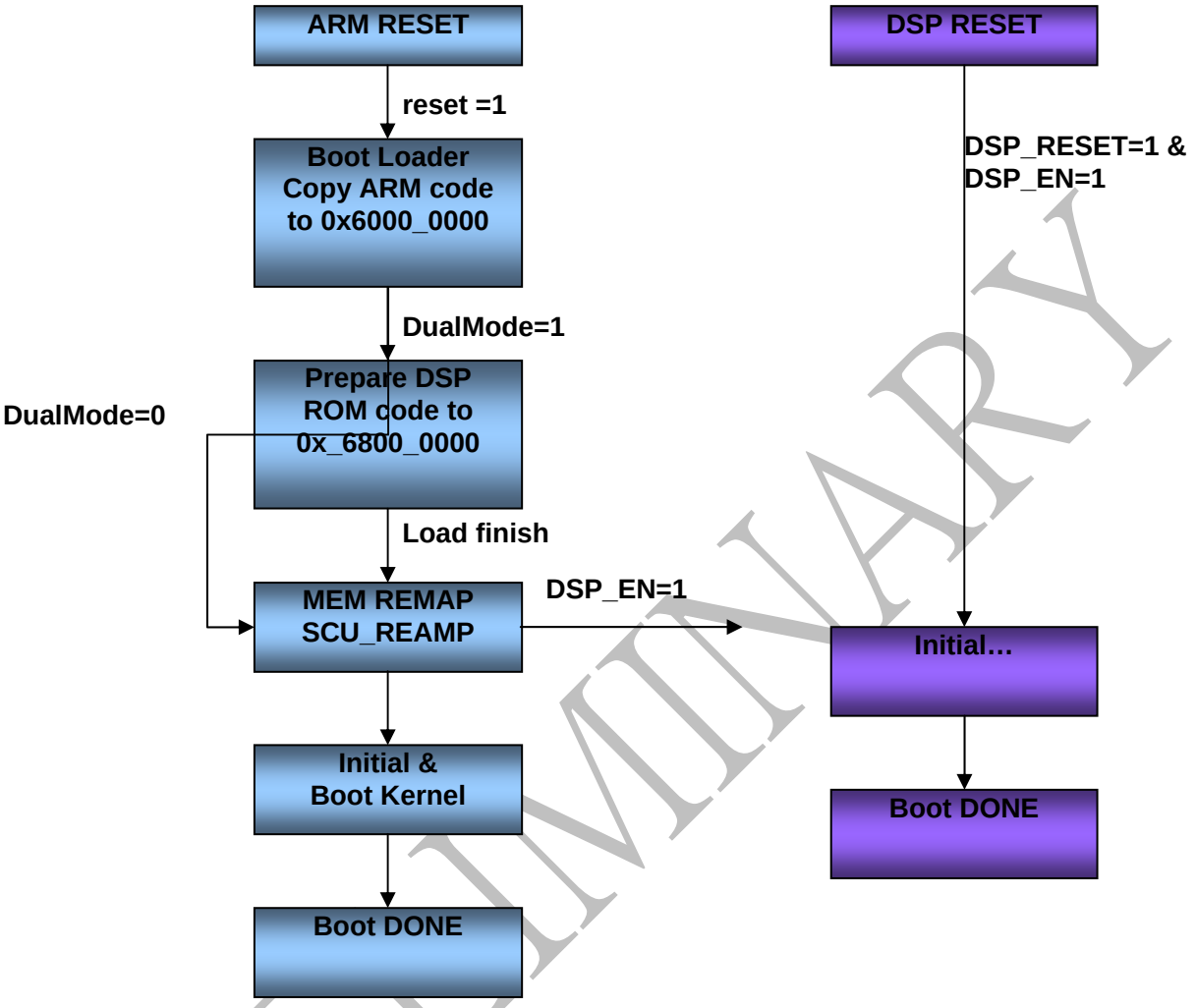
Dual-Core Communication Flow using MAILBOX



3.4.2 Programming sequence

Sequential booting

After power-on reset, the DSP will still on-reset6state until DSP enable DSP. Then DSP will leave reset state to active



ARM7EJS Power Down Flow

Set ARM7EJS CLKEN LOW, then stop CLK, and sustain at HIGH

ARM7EJS Wake-up Flow

Enable CLK, then set ARM7EJS CLKEN HIGH.

Chapter 4 AHB Bus Arbiter

4.1 Design Overview

4.1.1 Overview

The AHB bus arbiter can arbitrate 16 AHB masters including default master. Each bus master requests control of the bus and the arbiter decides which has the highest priority and issues a grant signal accordingly if fixed arbitration scheme is selected. The arbitration scheme is designed to be 16-stage programmable priority. The following table gives the default arbitration priority. If users wish to change the arbitration priority, they may program the corresponding register bit.

In order to fully comply with the AMBA specification 2.0, the arbiter must provide the following functions:

LOCK transfer control
Fixed arbitration scheme

4.1.2 Features

- Provide round robin, fixed priority, programmable arbitration scheme
- Provide 2-stage, 16-level programmable for fixed priority arbitration
- Capable to handle up to 16 AHB masters

4.2 Registers

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

4.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
ARB_MODE	0x0000	W	0x00000001	Arbiter mode selection.
ARB_PRIOx	0x0004 ~ 0x003C	W	0x00000000	Arbiter priority encoder register for Master number x (x= 1 ~ 15).

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

4.2.2 Detail Register Description

ARB_MODE
Address: Operational Base + offset(0x00)
Brief function description of register ARB_MODE

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x1	Arbitration scheme mode. 1: 2-stage, fixed arbitration. 0: Round robin.

ARB_PRIOx (x=1 ~ 15)
Address: Operational Base + offset(0x4 * x)
Brief function description of register ARB_PRIOx (x=1 ~ 15)

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.

3:0	RW	0x0	Priority level for Master x. The priority level can be between 0 (lowest) and 15 (highest).

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

4.3 Functional Description

4.3.1 Operation

A bus master may request the bus during any cycle by setting its HBUSREQ HIGH. This is then sampled by the arbiter on the rising edge of HCLK, and passed through the priority algorithm to decide which master will have access to the bus during the next cycle. The HGRANT then goes HIGH to indicate which master is currently granted. Also the HMASTER will indicate the granted master number. The HLOCK may be used to ensure that, during an indivisible transfer, the current grant outputs do not change. HLOCK must be asserted at least one cycle before the locked transfer to prevent the arbiter from changing the HGRANT.

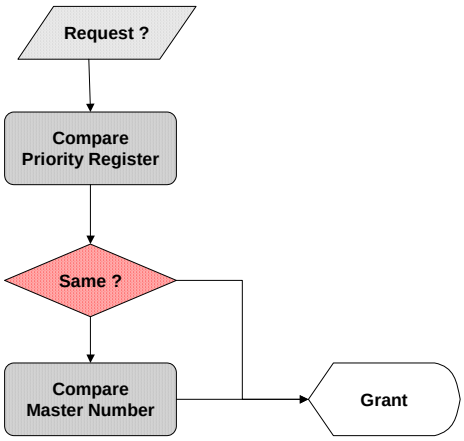
During reset, the CPU core is selected as the currently granted master.

The Leopard Generic Core chip requires a default master, which is selected when no masters are granted to the bus. The default master always performs IDLE transfers while it is granted to the bus. The default master is also selected during IDLE mode.

2-Stage fixed arbitration

During reset, the programmable priority register of each master is set to the default priority level. If the default priority level is same, the master number determines the arbitration priority. Please refer to **AHB Master Priority** in Description section of Chapter 3 Architecture for the AHB master number.

When the priority register is changed, arbiter will compare the contents of the priority register. If the corresponding priority is higher than other masters, arbiter will grant bus to that master. If the priority contents are the same, arbiter will compare the master number as given in the previous table. Figure below shows the arbitration procedure.



Round Robin arbitration

When the arbitration scheme used is the round-robin style. The round robin arbiter grants access in a manner that is meant to allow some degree of fairness among the various masters. Conceptually, each master is assigned a position on a wheel. When the highest priority master is granted, the wheel is moving such that the master with the next lower priority level is placed as the highest priority device. Initially, priorities are assigned such that Master 1 has the highest priority followed by Master 2 and so forth. When current granted master finishes its transfer, arbiter will watch the next token to see if it

wishes to request bus. If yes, arbiter grants bus to that master. Otherwise, arbiter will continue to look at the master request after next. When all masters do not request bus, arbiter will park bus to default master.

Chapter 5 Static/SDRAM Memory Controller

5.1 Design Overview

5.1.1 Overview

The Static/SDRAM memory controller is an AHB slave that provides an AHB interface to external SDRAM, ROM, SRAM and FLASH memories. It provides up to 3 AHB ports including one AHB register access port and the other two AHB data access ports. The interface pins with different memory type can be shared. The access timing of each memory type can be configured through the AHB access port.

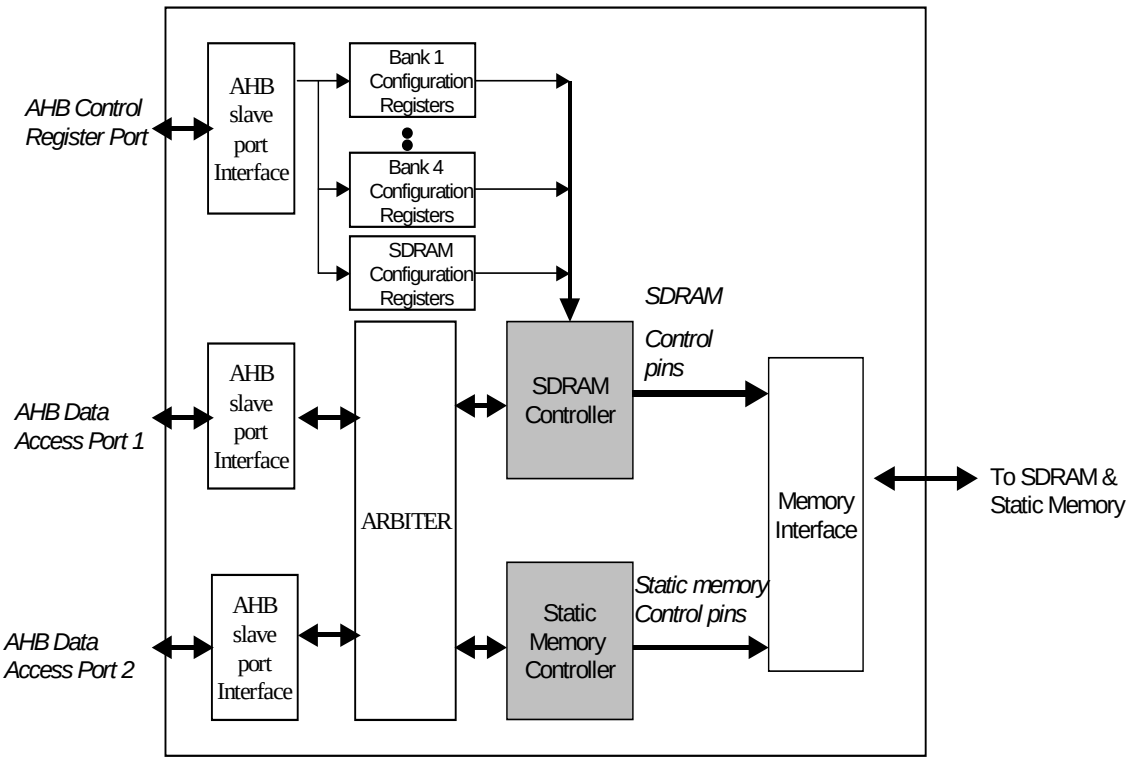
5.1.2 Features

- AMBA 32-bit AHB compliant
- Dynamic memory interface support
- Asynchronous static memory device support including SRAM, ROM and Flash with or without asynchronous page mode
- Support 1 independent AHB slave port for control register, and up to 2 individual AHB slave port for data access
- Dual write buffers for simultaneous write posting and SDRAM access by each AHB port
- 8, 16 wide static memory support
- Dedicated read buffer with data width matching
- Zero wait state burst data transfer on both AHB interfaces and SDRAM
- Early burst termination and AHB master busy are supported
- Static memory features include:
  - Programmable wait states
  - Bus turnaround delay
  - Output enable and write enable delays
- Support 1 chip selects for synchronous memory and 2 chip selects for static memory
- Pipeline access allows continuous data transfer without wasted cycles
- Fast page access on row addressing matching
- Independent row address matching for each of the SDRAM banks
- Programmable memory size: 4, 8 and 16-bit per SDRAM
- Dynamic memory self-refresh mode supported by software
- Supports industrial standard SDRAM from 64Mbits to 512 Mbits devices
- Operates on both discrete SDRAM chips and SDRAM DIMM
- Support for AHB burst types

5.2 Architecture

This section provides a description about the functions and behavior under various conditions.

5.2.1 Block Diagram



5.2.2 Block Descriptions

The Static/SDRAM memory controller consist of one AHB control register port used for memory configuration and two AHB Data access port used for AMBA component memory data access. The SDRAM controller generate SDRAM signals for external SDRAM access and the Static Memory Controller generate static memory signals that support SRAM, ROM and Flash.

5.3 Registers

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

5.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
MCSDR_MODE	0x0100	W	0x00000030	Setting Mode Register of the SDRAM.
MCSDR_ADDMAP	0x0104	W	0x000004FF	Memory size, banks setting.
MCSDR_ADDCFG	0x0108	W	0x00002222	Row address/Col address bits setting.
MCSDR_BASIC	0x010C	W	0x0000000A	SDRAM basic setting.
MCSDR_T_REF	0x0110	W	0x0000030C	SDRAM periodic refresh

				interval.
MCSDR_T_RFC	0x0114	W	0x00000008	SDRAM auto refresh period.
MCSDR_T_MRD	0x0118	W	0x00000001	SDRAM mode register timing.
MCSDR_T_RP	0x0120	W	0x00000001	SDRAM pre-charge command timing.
MCSDR_T_RCD	0x0124	W	0x00000001	SDRAM active to read or write delay.
MCST0_T_CEWD	0x0200	W	0x0000000F	Static memory 0 write CE width.
MCST0_T_CE2WE	0x0204	W	0x0000000F	Static memory 0 CE to WE timing.
MCST0_T_WEWD	0x0208	W	0x0000000F	Static memory 0 WE width.
MCST0_T_WE2CE	0x020C	W	0x0000000F	Static memory 0 WE to CE timing.
MCST0_T_CEWDR	0x0210	W	0x00000000	Static memory 0 read CE width.
MCST0_T_CE2RD	0x0214	W	0x0000000F	Static memory 0 CE to RD timing.
MCST0_T_RDWD	0x0218	W	0x00000015	Static memory 0 RD width.
MCST0_T_RD2CE	0x021C	W	0x00000000	Static memory 0 RD to CE timing.
MCST0_BASIC	0x0220	W	0x00000001	Static memory 0 data width and protection setting.
MCST1_T_CEWD	0x0300	W	0x0000000F	Static memory 1 write CE width.
MCST1_T_CE2WE	0x0304	W	0x0000000F	Static memory 1 CE to WE timing.
MCST1_T_WEWD	0x0308	W	0x0000000F	Static memory 1 WE width.
MCST1_T_WE2CE	0x030C	W	0x0000000F	Static memory 1 WE to CE timing.
MCST1_T_CEWDR	0x0310	W	0x00000000	Static memory 1 read CE width.
MCST1_T_CE2RD	0x0314	W	0x0000000F	Static memory 1 CE to RD timing.
MCST1_T_RDWD	0x0318	W	0x0000000F	Static memory 1 RD width.
MCST1_T_RD2CE	0x031C	W	0x00000000	Static memory 1 RD to CE timing.
MCST1_BASIC	0x0320	W	0x00000000	Static memory 1 data width and protection setting.

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

5.3.2 Detail Register Description

MCSDR_MODE
Address: Operational Base + offset(0x0100)
SDRAM CAS latency and burst length

Bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6:4	RW	0x3	SDRAM CAS latency. 0x2: CAS latency = 2. 0x3: CAS latency = 3. Others: Reserved.
3	-	-	Reserved.

2:0	RW	0x0	SDRAM burst length. 0x0: Burst length = 1. 0x1: Burst length = 2. 0x2: Burst length = 4. 0x3: Burst length = 8. Others: Reserved.

MCSDR_ADDMAP

Address: Operational Base + offset(0x0104)

Memory modules address mapping

Bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5:4	RW	0x3	The module 0 memory size. 0x0: 64M Bytes. 0x1: 128M Bytes. 0x2: 256M Bytes. 0x3: 512M Bytes.
3:2	RW	0x3	Reserved.
1:0	RW	0x3	The number of banks in module 0. 0x0 : Reserved.. 0x1: 1 bank, CS0 is used. 0x2~0x3: Reserved.

MCSDR_ADDCFG

Address: Operational Base + offset(0x0108)

Memory modules address config

Bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5:4	RW	0x2	Set number of bits of SDRAM module 0 row address. 0x0: 11 bits. 0x1: 12 bits. 0x2: 13 bits. 0x3: Reserved.
3	-	-	Reserved.
2:0	RW	0x2	Set number of bits of SDRAM module 0 column address. 0x0: 8 bits. 0x1: 9 bits. 0x2: 10 bits. 0x3: 11 bits. 0x4: 12 bits. 0x5~0x7: Reserved.

MCSDR_BASIC

Address: Operational Base + offset(0x010C)

SDRAM Basic setting

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:5	RW	0x0	The priority bits will control the ARBITER priority strategy of different slave port for read/write operation. 3'b000: Round-robin (RR) priority. The write priority is higher than read priority. It will change priority after read/write operation. (Either S2 write > S1 write > S2 read > S1 read, or S1 write > S2 write > S1 read > S2 read)

			3'b001: Round-robin (RR) priority. The write priority is the same as read priority. It will change priority after read/write operation. (Either S2 write > S2 read > S1 write > S1 read, or S1 write > S1 read > S2 write >S2 read) 3'b010: S1 has higher priority. The write priority is higher than read priority. (S1 write > S2 write > S1 read > S2 read) 3'b011: S1 has higher priority. The write priority is the same as read priority. (S1 write > S1 read >S2 write > S2 read) 3'b100: S2 has higher priority. The write priority is higher than read priority. (S2 write > S1 write > S2 read > S1 read) 3'b101: S2 has higher priority. The write priority is the same as read priority. (S2 write > S2 read > S1 write >S1 read) 3'b110~ 3'b111: Reserved There is no priority in the same slave port. It is first-in-first-out service.
4	-	-	Reserved.
3:2	RW	0x2	SDRAM module 1 data width size. 0x0: Reserved. 0x1: 16 bits. 0x2 ~ 0x3: Reserved.
1:0	RW	0x2	SDRAM module 0 data width size. 0x0: Reserved. 0x1: 16 bits. 0x2 ~ 0x3: Reserved.

MCSDR_T_REF
Address: Operational Base + offset(0x0110)
Average periodic refresh interval

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11:0	RW	0x30C	Average periodic AUTO REFRESH interval. If SDRAM requires AUTO REFRESH cycles at an average interval of 15.625μs, the T_REF value must be set no greater than 1560 at 100MHz.

MCSDR_T_RFC
Address: Operational Base + offset(0x0114)
Auto refresh period

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x8	Auto refresh period. Time = (T_RFC + 1) × clock cycles

MCSDR_T_MRD
Address: Operational Base + offset(0x0118)
Command to ACTIVE or REFRESH period

bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x1	Command to ACTIVE or REFRESH period. Time = (T_MRD + 1) × clock cycles

MCSDR_T_RP
Address: Operational Base + offset(0x0120)
Pre-charge command period

bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x1	Pre-charge command period. Time = (T_RP + 1) × clock cycles

MCSDR_T_RCD
Address: Operational Base + offset(0x0124)
Active to Read or Write delay

bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x1	Active to Read or Write delay. Time = (T_RCD + 1) × clock cycles

MCSTx_T_CEWD(1) (x=0 ~ 1)
Address: Operational Base + offset(0x0200, 0x0300)
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 Time = (t_cewd) × clock cycles

MCSTx_T_CE2WE(1) (x=0 ~ 1)
Address: Operational Base + offset(0x0204, 0x0304)
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 Time = (t_ce2we) × clock cycles

MCSTx_T_WEWD (1) (x=0 ~ 1)
Address: Operational Base + offset(0x0208, 0x0308)
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 Time = (t_wewd) × clock cycles

MCSTx_T_WE2CE (1) (x=0 ~ 1)
Address: Operational Base + offset(0x020C, 0x030C)
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 Time = (t_we2c) × clock cycles

MCSTx_T_CEWDR (1) (x=0 ~ 1)
Address: Operational Base + offset(0x0210, 0x0310)
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 Time = (t_cewdr) × clock cycles

MCSTx_T_CE2RD (1) (x=0 ~ 1)
Address: Operational Base + offset(0x0214, 0x0314)
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 Time = (t_ce2rd) × clock cycles

MCSTx_T_RDWD (1) (x=0 ~ 1)
Address: Operational Base + offset(0x0218, 0x0318)
Static memory timing control register for RD width

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	X=0, 0x15 X=1~3, 0xF	Static memory timing control register for RD width Time = (t_rdwd) × clock cycles

MCSTx_T_RD2CE (1) (x=0 ~ 1)
Address: Operational Base + offset(0x021C, 0x031C)
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 Time = (t_rd2ce) × clock cycles

MCSTx_BASIC (1) (x=0 ~ 1)
Address: Operational Base + offset(0x0220, 0x0320)
Static memory basic setting

bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5	RW	0x0	Write protect bit0: Writable 1: Write protect
4	RW	0x0	Byte Enable Option 0: For READ all the bits in BE_n are HIGH. For WRITE the respective active bits in BE_n are LOW. 1: For READ all the bits in BE_n are LOW. For WRITE the respective active bits in BE_n are LOW.
3:2	-	-	Reserved.
1:0	RW	X=0, 0x1 X=1, 0x0	Static memory data size 0x0 : 8 bits 0x1 : 16 bits 0x2~0x3: Reserved.

Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only
Note 1: x indicate the control registers for static memory x

5.4 Functional Description

5.4.1 Memory Integration

The memory controller provides 1 control register programming AHB slave port (port

S0) and 2 data access AHB slave ports (Port S1 and S2). The control register port is used to CPU programming the memory configuration and setting. The data access ports are used to access the SDRAM or static memory data. There are 3 HSEL signals in each data access port. One is decoded for SDRAM address space; the other 2 HSELs are decoded in individual static memory banks. Each data access ports are able to access the same 2 external static memory banks and 1 SDRAM memory modules. For example, if you want to access the static memory bank 0, the HSEL[1] must be asserted. The FLASH_CE [1:0] are chip select signals used to select static memory devices from bank 1 to bank 0. But in SDRAM, only HSEL[0] is used for decoding SDRAM memory. There are 1 memory modules for SDRAM integrate. You can specify the number of banks and size for each module in Basic2 register. The CS_n [2:0] are chip select signals used to select SDRAM modules and banks.

Select	Memory Type	Device	Chip Select
Sx_HSEL[0]	SDRAM	SDRAM Module0	CS_n [0] for 1 bank
Sx_HSEL[1]	Static Memory	Static memory 0	FLASH_CE [0]
Sx_HSEL[2]		Static memory 1	FLASH_CE [1]

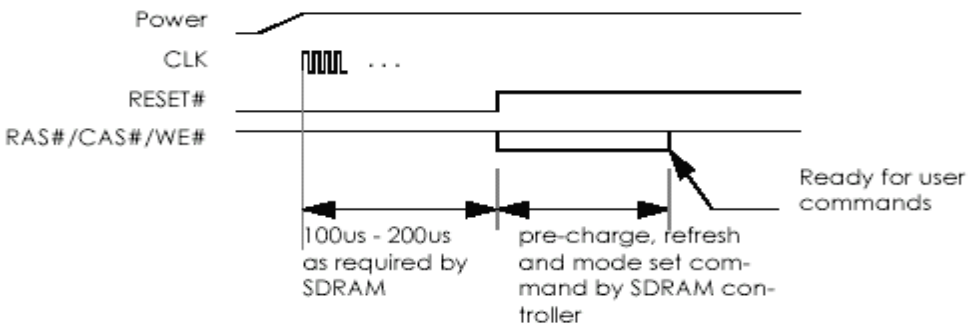
The following figure shows the transfer mechanism of address mapping from AHB bus to SDRAM interface.

Bit #	31	← Controlled by → ROW_BITS[2:0]	← Controlled by → COL_BITS[2:0]	1:0
HADDR	BA[1:0]	Row Address	Column Address	Byte

5.4.2 Initialization

The Power-on sequence of most SDRAMs requires that the system power be applied to the system with the clock running for 100us to 200us. During this time, HRESET_N must be asserted to the SDRAM controller to prevent if from accessing the SDRAMs. The user needs to control the duration of the HRESET_N input to meet SDRAM requirements.

Once HRESET_N is de-asserted, the SDRAM controller first issues a pre-charge command and then issues 15 auto-refresh commands. Most SDRAMs require 2 to 8 auto-refresh commands after reset. At the completion of the refresh sequence, the SDRAM controller issues a mode set command to initialize the SDRAM mode register. The following is a timing diagram showing the initialization sequence.



When the HRESET_N signal is asserted, the SDRAM controller automatically resets all its control registers into their default values. Timing of the initial accesses including the mode set command is based on the default timing parameters. After initialization, the user can modify the timing parameters and other settings of the control registers by configuring SDRAM registers via AHB slave interface Port S0. The user must make sure the default value of the control registers are within the operation range of the SDRAMs used.

The control registers contain the timing parameters of the SDRAM. With careful

selection of the configuration value, the memory controller would provide optimum performance after system reset and the system can be operational without any run time configuration. However, a configuration port (Port S0) is also provided in case it becomes necessary for the user to change the control register settings during run time.

5.4.3 Operation

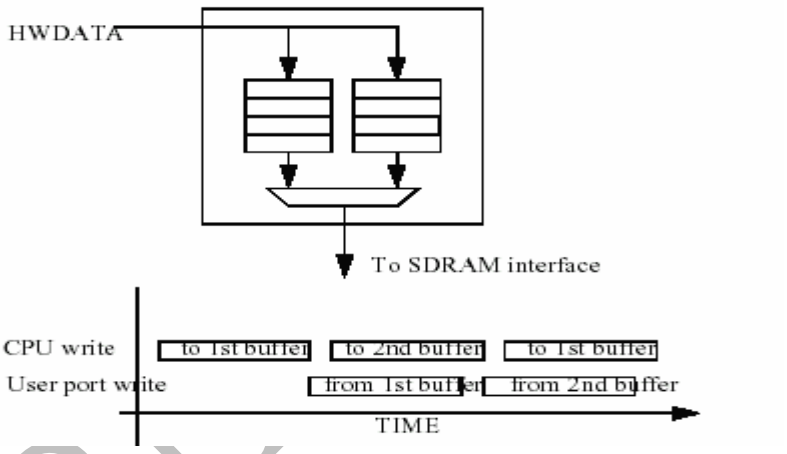
The memory controller can be read or written by data access AHB port. The AHB bus can read the data from read-buffer or write data into write-buffers. The buffers can issue request command to SDRAM controller to write or read data. In the write transfer, if the SDRAM controller is not busy, then write-buffer can get the data from AHB bus, and write data from write-buffers to SDRAM. In the read transfer, the SDRAM controller can get the data from memory to read-buffers. Then, the AHB bus can get the data from read-buffers.

AHB write to SDRAM

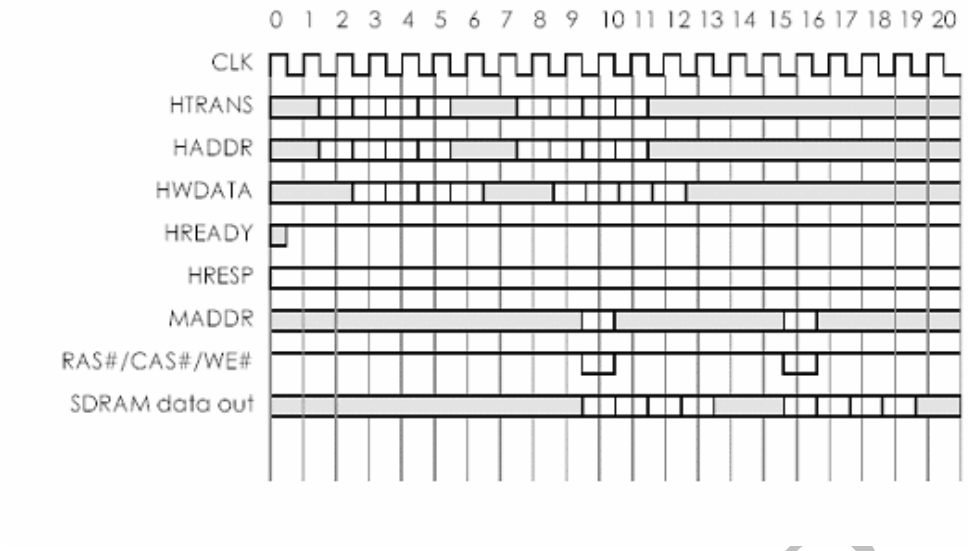
The AHB write data to SDRAM. First, the data are written into write-buffers by the data access AHB port. Then, the buffers controller can issue a request to the SDRAM controller. If the SDRAM is not busy, then the SDRAM controller can write data from write-buffers into SDRAMs. Each data access AHB interface contains 2 write buffers for posting write data to the SDRAM. Each write buffer can store up to 64 bytes of data.

Data can be written from the AHB interface in 8-bit, 16-bit or 32-bit mode. The write buffer is organized in 32-bit mode so that can be transferred to the SDRAM in burst mode regardless of the AHB transfer size.

The following diagram illustrates the concept of double write buffers.



The AHB bus interface contains two write buffers for posting write data. Data can be written to one write buffer from the AHB bus while posted data is written to the SDRAM. The following example shows a back-to-back write access sequence by the AHB Bus.



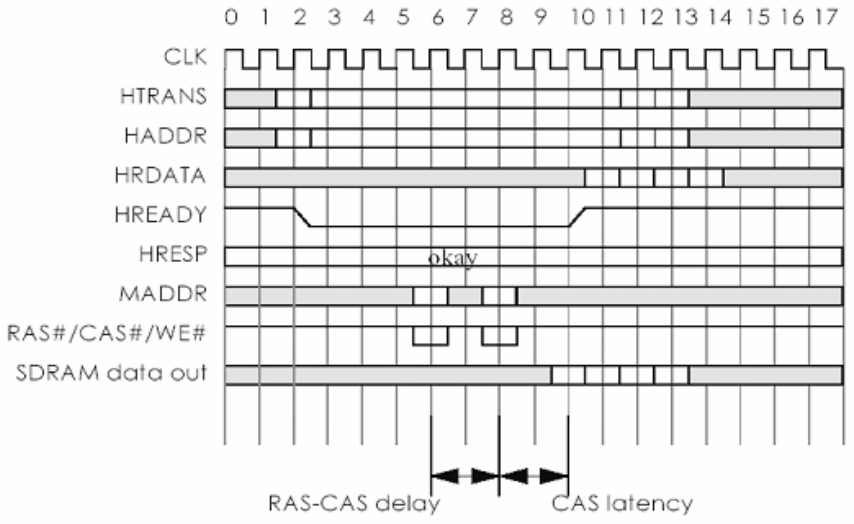
AHB read from SDRAM

The AHB bus initiates a SDRAM access by issuing a NONSEQ request on the HTRANS signal. At the same time if HSEL is asserted, the SDRAM controller accepts the request, otherwise, it assumes the request is for other devices on the bus and ignores it. If HWRITE is low, it indicates a read access is requested.

After the request is accepted and if the SDRAM is not currently servicing other request, a memory read access is initiated. The row address is first sent to the SDRAM.

Read data is available from the SDRAM after the CAS latency delay from the column address. Read data is first sampled by the SDRAM controller and then driven on to the AHB bus one cycle later.

If the controller is in pre-charge or refresh mode when the read access is requested, the row active command will be delayed until the pre-charge or refresh cycle is finished. The following timing diagram shows a SDRAM read access of 4 data. The RAS-CAS latency and CAS-data latency are programmable timing parameters.



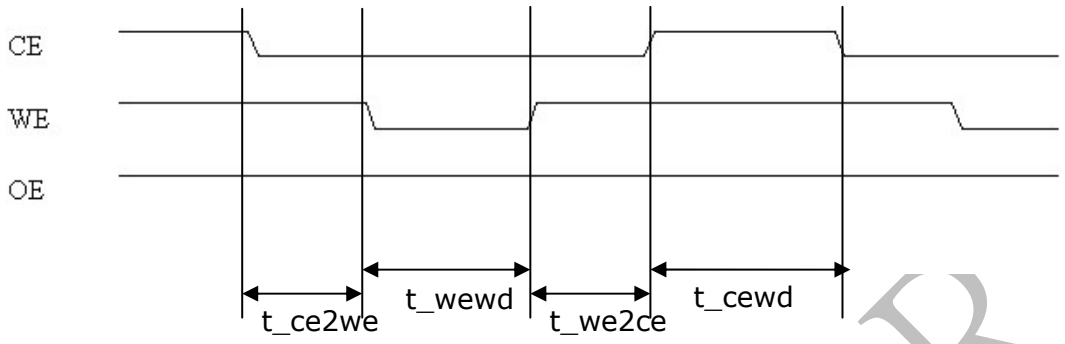
AHB read/ write from Flash/ROM

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 provide as a reference to the internal operation of the core.

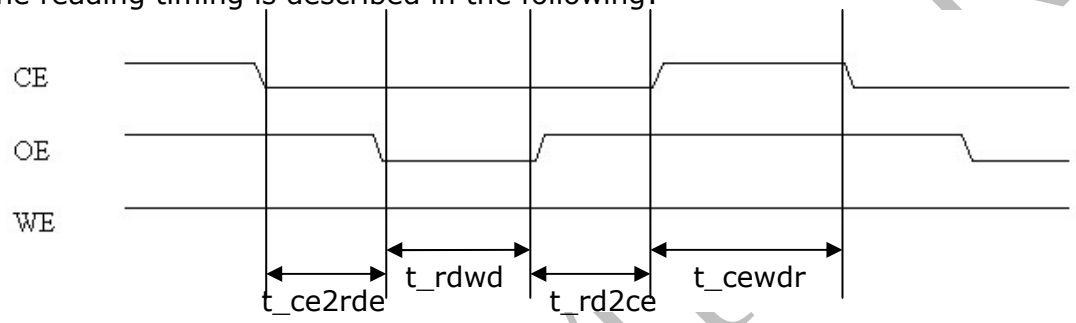
Memory access is started by the address (ADDR) and chip select signal. Once asserted, the memory will provide the read/write data after the access time. The memory manufacturer in unit of nanosecond provides access time. The memory controller can be

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 6 Interrupt Controller

6.1 Design Overview

6.1.1 Overview

The interrupt controller is an AHB slave, providing software interface to interrupt system. The interrupt controller can accept 32 interrupt sources from devices. The interrupt controller will determine which is the most urgent interrupt then issue an interrupt, IRQ, to CPU core. At any given time, when CPU receives an interrupt, it will follow the priority to handle the requests.

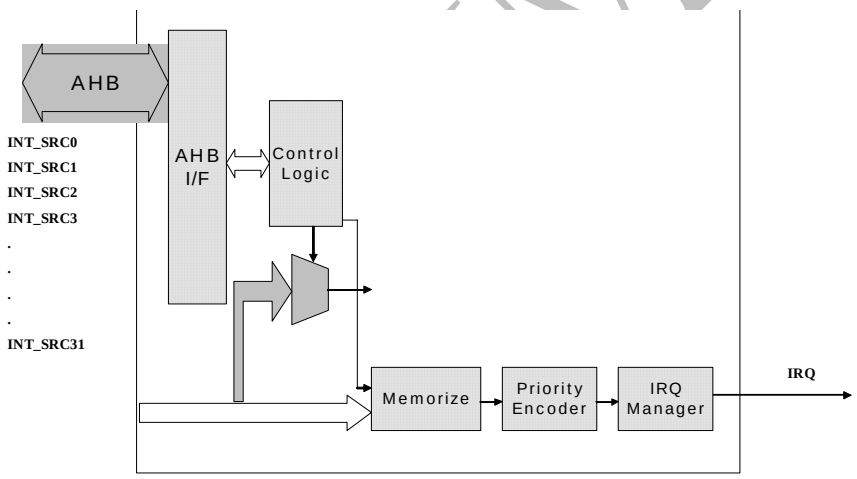
6.1.2 Features

- Source status and interrupt request status
- Separate enable set and enable clear registers to allow independent bit enable of interrupt sources
- Programmable hardware priority handling
- Programmable High/Low Level sensitive or Negative/Positive edge triggered interrupts

6.2 Architecture

This section provides a description about the functions and behavior under various conditions.

6.2.1 Block Diagram



6.2.2 Block Descriptions

Interrupt consist of an AHB slave interface for interrupt attribute setting, and the priority encoder will decided which interrupt request will be served next by detecting the priority of each interrupt.

6.3 Registers

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

6.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
INTC_SCRx	0x0000 ~ 0x007C	W	0x00000040	Source Control Register x.
INTC_ISR	0x0104	W	0x00000000	Interrupt Status Register.
INTC_IPR	0x0108	W	0x00000000	Interrupt Pending Register.
INTC_IMR	0x010C	W	0x00000000	Interrupt Mask Register.
INTC_IECR*	0x0114	W	0x00000000	Interrupt Enable Command Register.
INTC_ICCR	0x0118	W	0x00000000	Interrupt Clear Command Register.
INTC_ISCR	0x011C	W	0x00000000	Interrupt Set Command Register.
INTC_TEST	0x0124	W	0x00000000	Test Mode Register.

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

Interrupts (ARM)

Source #	Source Description	Polarity
31 (High)	SoftWare Interrupt	-
30	SoftWare Interrupt	-
29	SoftWare Interrupt	-
28	DSP	High level
27	LCDC	High level
26	NANDC	High level
25	DWDMA	High level
24	VIP	High level
23	GPIO1 (a/b group)	High level
22	ADC	High level
21	PWM3	High level
20	PWM2	High level
19	PWM1	High level
18	PWM0	High level
17	UHC	LOW level
16	UDC	High level
15	I2S	High level
14	I2C	High level
13	A2A bridge & DMA	High level
12	HDMA	High level
11	SPI master controller	High level
10	SD	High level
9	SCU	High level
8	RTC	High level
7	AHB0 MAILBOX	High level
6	SoftWare Interrupt	-
5	GPIO0 (a/b group)	High level
4	Timer0_2	High level
3	Timer0_1	High level
2	Timer0_0	High level

1	UART1	High level
0 (Low)	UART0	High level

Interrupts (DSP)

Source #	Source Description	Polarity
nmi	ARM7EJC	High level
13	Software Interrupt	High level
12	Software Interrupt	High level
11	PWM0	High level
10	GPIO0 (c/d group)	High level
9	HS_ADC	High level
8	LCDC	High level
7	VIP	High level
6	Timer1 (internal)	High level
5	Timer0 (internal)	High level
4	AHB1 MAILBOX	High level
3	DWDMA	High level
2	A2A bridge&DMA	High level
1	HDMA	High level
0	UHC or UDC (combinational)	High level

6.3.2 Detail Register Description

INTC_SCRx (x=0 ~ 31)

Address: Operational Base + offset(x*4)

For example, the address of source number 11 (INTC_SCR11)
= INTC_BASE + 11*4 = INTC_BASE + 0x002C

These registers are used for interrupt sources configuration. Now it supports trigger type and priority level control for each interrupt.

Source Control Registers

Bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:6	RW	0x1	Program the input sources to be high/low sensitive or negative/positive triggered. 0x0: Low-Level sensitive. 0x1: High-Level sensitive. 0x2: Negative-Edge triggered. 0x3: Positive-Edge triggered.
5	-	-	Reserved.
4:0	RW	0x0	The priority level can be between 0 (lowest) and 31 (highest).

INTC_ISR

Address: Operational Base + offset(0x104)

This register records the index of current highest priority interrupt. The 5-bits index supports 32 interrupt sources.

Interrupt Status Register

bit	Attr	Reset Value	Description
31:5	-	-	Reserved.
4:0	R	0x0	Current IRQ ID. The Interrupt Status Register returns the current interrupt source number. The interrupt source number is given in the end of this section.

INTC_IPR

Address: Operational Base + offset(0x108)

The register will keep the record that interrupt is pending. Each bit is referred to

corresponding interrupt source.
Interrupt Pending Register

bit	Attr	Reset Value	Description
31	R	0x0	Interrupt source 31 interrupt status. 0:inactive 1:Pending
30	R	0x0	Interrupt source 30 interrupt status. 0:inactive 1:Pending
29	R	0x0	Interrupt source 29 interrupt status. 0:inactive 1:Pending
28	R	0x0	Interrupt source 28 interrupt status. 0:inactive 1:Pending
27	R	0x0	Interrupt source 27 interrupt status. 0:inactive 1:Pending
26	R	0x0	Interrupt source 26 interrupt status. 0:inactive 1:Pending
25	R	0x0	Interrupt source 25 interrupt status. 0:inactive 1:Pending
24	R	0x0	Interrupt source 24 interrupt status. 0:inactive 1:Pending
23	R	0x0	Interrupt source 23 interrupt status. 0:inactive 1:Pending
22	R	0x0	Interrupt source 22 interrupt status. 0:inactive 1:Pending
21	R	0x0	Interrupt source 21 interrupt status. 0:inactive 1:Pending
20	R	0x0	Interrupt source 20 interrupt status. 0:inactive 1:Pending
19	R	0x0	Interrupt source 19 interrupt status. 0:inactive 1:Pending
18	R	0x0	Interrupt source 18 interrupt status. 0:inactive 1:Pending
17	R	0x0	Interrupt source 17 interrupt status. 0:inactive 1:Pending
16	R	0x0	Interrupt source 16 interrupt status. 0:inactive 1:Pending
15	R	0x0	Interrupt source 15 interrupt status. 0:inactive 1:Pending
14	R	0x0	Interrupt source 14 interrupt status. 0:inactive 1:Pending
13	R	0x0	Interrupt source 13 interrupt status. 0:inactive 1:Pending
12	R	0x0	Interrupt source 12 interrupt status. 0:inactive 1:Pending
11	R	0x0	Interrupt source 11 interrupt status. 0:inactive 1:Pending
10	R	0x0	Interrupt source 10 interrupt status. 0:inactive 1:Pending
9	R	0x0	Interrupt source 9 interrupt status. 0:inactive 1:Pending
8	R	0x0	Interrupt source 8 interrupt status. 0:inactive 1:Pending
7	R	0x0	Interrupt source 7 interrupt status. 0:inactive 1:Pending
6	R	0x0	Interrupt source 6 interrupt status. 0:inactive 1:Pending
5	R	0x0	Interrupt source 5 interrupt status.

			0:inactive 1:Pending
4	R	0x0	Interrupt source 4 interrupt status. 0:inactive 1:Pending
3	R	0x0	Interrupt source 3 interrupt status. 0:inactive 1:Pending
2	R	0x0	Interrupt source 2 interrupt status. 0:inactive 1:Pending
1	R	0x0	Interrupt source 1 interrupt status. 0:inactive 1:Pending
0	R	0x0	Interrupt source 0 interrupt status. 0:inactive 1:Pending

INTC_IMR

Address: Operational Base + offset(0x10C)

The register is used to mask the pending interrupt to get into the priority encoder, and then the highest priority interrupt will be re-selected. The hardware default will mask all interrupts. Software needs to set appropriate bit to high to pass interrupt to priority encoder.

Interrupt Mask Register

bit	Attr	Reset Value	Description
31	RW	0x0	Interrupt source 31 mask register. 0:Masked 1:Un-masked
30	RW	0x0	Interrupt source 30 mask register. 0:Masked 1:Un-masked
29	RW	0x0	Interrupt source 29 mask register. 0:Masked 1:Un-masked
28	RW	0x0	Interrupt source 28 mask register. 0:Masked 1:Un-masked
27	RW	0x0	Interrupt source 27 mask register. 0:Masked 1:Un-masked
26	RW	0x0	Interrupt source 26 mask register. 0:Masked 1:Un-masked
25	RW	0x0	Interrupt source 25 mask register. 0:Masked 1:Un-masked
24	RW	0x0	Interrupt source 24 mask register. 0:Masked 1:Un-masked
23	RW	0x0	Interrupt source 23 mask register. 0:Masked 1:Un-masked
22	RW	0x0	Interrupt source 22 mask register. 0:Masked 1:Un-masked
21	RW	0x0	Interrupt source 21 mask register. 0:Masked 1:Un-masked
20	RW	0x0	Interrupt source 20 mask register. 0:Masked 1:Un-masked
19	RW	0x0	Interrupt source 19 mask register. 0:Masked 1:Un-masked
18	RW	0x0	Interrupt source 18mask register. 0:Masked 1:Un-masked
17	RW	0x0	Interrupt source 17 mask register. 0:Masked 1:Un-masked
16	RW	0x0	Interrupt source 16 mask register. 0:Masked 1:Un-masked
15	RW	0x0	Interrupt source 15 mask register. 0:Masked 1:Un-masked
14	RW	0x0	Interrupt source 14 mask register. 0:Masked 1:Un-masked
13	RW	0x0	Interrupt source 13 mask register.

			0:Masked 1:Un-masked
12	RW	0x0	Interrupt source 12 mask register. 0:Masked 1:Un-masked
11	RW	0x0	Interrupt source 11 mask register. 0:Masked 1:Un-masked
10	RW	0x0	Interrupt source 10 mask register. 0:Masked 1:Un-masked
9	RW	0x0	Interrupt source 9 mask register. 0:Masked 1:Un-masked
8	RW	0x0	Interrupt source 8 mask register. 0:Masked 1:Un-masked
7	RW	0x0	Interrupt source 7 mask register. 0:Masked 1:Un-masked
6	RW	0x0	Interrupt source 6 mask register. 0:Masked 1:Un-masked
5	RW	0x0	Interrupt source 5 mask register. 0:Masked 1:Un-masked
4	RW	0x0	Interrupt source 4 mask register. 0:Masked 1:Un-masked
3	RW	0x0	Interrupt source 3 mask register. 0:Masked 1:Un-masked
2	RW	0x0	Interrupt source 2 mask register. 0:Masked 1:Un-masked
1	RW	0x0	Interrupt source 1 mask register. 0:Masked 1:Un-masked
0	RW	0x0	Interrupt source 0 mask register. 0:Masked 1:Un-masked

INTC_IECR

Address: Operational Base + offset(0x114)

The register is used to enable the interrupt source to get into the pending register. If the corresponding bit is disabled, and then the detector will not issue interrupt event to pending register. Software needs to set appropriate bit to high to enable interrupt sources at initial stage.

Interrupt Enable Command Register

bit	Attr	Reset Value	Description
31	RW	0x0	Interrupt source 31 interrupt enable register. 0:Disabled 1:Enabled
30	RW	0x0	Interrupt source 30 interrupt enable register. 0:Disabled 1:Enabled
29	RW	0x0	Interrupt source 29 interrupt enable register. 0:Disabled 1:Enabled
28	RW	0x0	Interrupt source 28 interrupt enable register. 0:Disabled 1:Enabled
27	RW	0x0	Interrupt source 27 interrupt enable register. 0:Disabled 1:Enabled
26	RW	0x0	Interrupt source 26 interrupt enable register. 0:Disabled 1:Enabled
25	RW	0x0	Interrupt source 25 interrupt enable register. 0:Disabled 1:Enabled
24	RW	0x0	Interrupt source 24 interrupt enable register. 0:Disabled 1:Enabled
23	RW	0x0	Interrupt source 23 interrupt enable register. 0:Disabled 1:Enabled
22	RW	0x0	Interrupt source 22 interrupt enable register. 0:Disabled 1:Enabled
21	RW	0x0	Interrupt source 21 interrupt enable register.

			0:Disabled 1:Enabled
20	RW	0x0	Interrupt source 20 interrupt enable register. 0:Disabled 1:Enabled
19	RW	0x0	Interrupt source 19 interrupt enable register. 0:Disabled 1:Enabled
18	RW	0x0	Interrupt source 18 interrupt enable register. 0:Disabled 1:Enabled
17	RW	0x0	Interrupt source 17 interrupt enable register. 0:Disabled 1:Enabled
16	RW	0x0	Interrupt source 16 interrupt enable register. 0:Disabled 1:Enabled
15	RW	0x0	Interrupt source 15 interrupt enable register. 0:Disabled 1:Enabled
14	RW	0x0	Interrupt source 14 interrupt enable register. 0:Disabled 1:Enabled
13	RW	0x0	Interrupt source 13 interrupt enable register. 0:Disabled 1:Enabled
12	RW	0x0	Interrupt source 12 interrupt enable register. 0:Disabled 1:Enabled
11	RW	0x0	Interrupt source 11 interrupt enable register. 0:Disabled 1:Enabled
10	RW	0x0	Interrupt source 10 interrupt enable register. 0:Disabled 1:Enabled
9	RW	0x0	Interrupt source 9 interrupt enable register. 0:Disabled 1:Enabled
8	RW	0x0	Interrupt source 8 interrupt enable register. 0:Disabled 1:Enabled
7	RW	0x0	Interrupt source 7 interrupt enable register. 0:Disabled 1:Enabled
6	RW	0x0	Interrupt source 6 interrupt enable register. 0:Disabled 1:Enabled
5	RW	0x0	Interrupt source 5 interrupt enable register. 0:Disabled 1:Enabled
4	RW	0x0	Interrupt source 4 interrupt enable register. 0:Disabled 1:Enabled
3	RW	0x0	Interrupt source 3 interrupt enable register. 0:Disabled 1:Enabled
2	RW	0x0	Interrupt source 2 interrupt enable register. 0:Disabled 1:Enabled
1	RW	0x0	Interrupt source 1 interrupt enable register. 0:Disabled 1:Enabled
0	RW	0x0	Interrupt source 0 interrupt enable register. 0:Disabled 1:Enabled

INTC_ICCR

Address: Operational Base + offset(0x118)

The register allows software programmer to clear pending interrupts. Only at the write cycle, the clear action will be taken. The software programmer can use this register to clear interrupt source that is serviced.

Interrupt Clear Command Register

bit	Attr	Reset Value	Description
31	W	0x0	Interrupt source 31 interrupt clear register. 0:No effect 1:Cleared
30	W	0x0	Interrupt source 30 interrupt clear register. 0:No effect 1:Cleared
29	W	0x0	Interrupt source 29 interrupt clear register. 0:No effect 1:Cleared

28	W	0x0	Interrupt source 28 interrupt clear register. 0:No effect 1:Cleared
27	W	0x0	Interrupt source 27 interrupt clear register. 0:No effect 1:Cleared
26	W	0x0	Interrupt source 26 interrupt clear register. 0:No effect 1:Cleared
25	W	0x0	Interrupt source 25 interrupt clear register. 0:No effect 1:Cleared
24	W	0x0	Interrupt source 24 interrupt clear register. 0:No effect 1:Cleared
23	W	0x0	Interrupt source 23 interrupt clear register. 0:No effect 1:Cleared
22	W	0x0	Interrupt source 22 interrupt clear register. 0:No effect 1:Cleared
21	W	0x0	Interrupt source 21 interrupt clear register. 0:No effect 1:Cleared
20	W	0x0	Interrupt source 20 interrupt clear register. 0:No effect 1:Cleared
19	W	0x0	Interrupt source 19 interrupt clear register. 0:No effect 1:Cleared
18	W	0x0	Interrupt source 18 interrupt clear register. 0:No effect 1:Cleared
17	W	0x0	Interrupt source 17 interrupt clear register. 0:No effect 1:Cleared
16	W	0x0	Interrupt source 16 interrupt clear register. 0:No effect 1:Cleared
15	W	0x0	Interrupt source 15 interrupt clear register. 0:No effect 1:Cleared
14	W	0x0	Interrupt source 14 interrupt clear register. 0:No effect 1:Cleared
13	W	0x0	Interrupt source 13 interrupt clear register. 0:No effect 1:Cleared
12	W	0x0	Interrupt source 12 interrupt clear register. 0:No effect 1:Cleared
11	W	0x0	Interrupt source 11 interrupt clear register. 0:No effect 1:Cleared
10	W	0x0	Interrupt source 10 interrupt clear register. 0:No effect 1:Cleared
9	W	0x0	Interrupt source 9 interrupt clear register. 0:No effect 1:Cleared
8	W	0x0	Interrupt source 8 interrupt clear register. 0:No effect 1:Cleared
7	W	0x0	Interrupt source 7 interrupt clear register. 0:No effect 1:Cleared
6	W	0x0	Interrupt source 6 interrupt clear register. 0:No effect 1:Cleared
5	W	0x0	Interrupt source 5 interrupt clear register. 0:No effect 1:Cleared
4	W	0x0	Interrupt source 4 interrupt clear register. 0:No effect 1:Cleared
3	W	0x0	Interrupt source 3 interrupt clear register. 0:No effect 1:Cleared
2	W	0x0	Interrupt source 2 interrupt clear register. 0:No effect 1:Cleared
1	W	0x0	Interrupt source 1 interrupt clear register. 0:No effect 1:Cleared
0	W	0x0	Interrupt source 0 interrupt clear register.

			0:No effect 1:Cleared

INTC_ISCR

Address: Operational Base + offset(0x11C)

The register allows software programmer to issue interrupts during test mode.

Its trigger type is high-level. The software programmer needs to clear the corresponding bit to stop issue interrupt after the exception test.

Interrupt Set Command Register

bit	Attr	Reset Value	Description
31	W	0x0	Interrupt source 31 interrupt SW set register in test mode. 0:No effect 1:Set
30	W	0x0	Interrupt source 30 interrupt SW set register in test mode. 0:No effect 1:Set
29	W	0x0	Interrupt source 29 interrupt SW set register in test mode. 0:No effect 1:Set
28	W	0x0	Interrupt source 28 interrupt SW set register in test mode. 0:No effect 1:Set
27	W	0x0	Interrupt source 27 interrupt SW set register in test mode. 0:No effect 1:Set
26	W	0x0	Interrupt source 26 interrupt SW set register in test mode. 0:No effect 1:Set
25	W	0x0	Interrupt source 25 interrupt SW set register in test mode. 0:No effect 1:Set
24	W	0x0	Interrupt source 24 interrupt SW set register in test mode. 0:No effect 1:Set
23	W	0x0	Interrupt source 23 interrupt SW set register in test mode. 0:No effect 1:Set
22	W	0x0	Interrupt source 22 interrupt SW set register in test mode. 0:No effect 1:Set
21	W	0x0	Interrupt source 21 interrupt SW set register in test mode. 0:No effect 1:Set
20	W	0x0	Interrupt source 20 interrupt SW set register in test mode. 0:No effect 1:Set
19	W	0x0	Interrupt source 19 interrupt SW set register in test mode. 0:No effect 1:Set
18	W	0x0	Interrupt source 18 interrupt SW set register in test mode. 0:No effect 1:Set
17	W	0x0	Interrupt source 17 interrupt SW set register in test mode. 0:No effect 1:Set
16	W	0x0	Interrupt source 16 interrupt SW set register in test mode. 0:No effect 1:Set

15	W	0x0	Interrupt source 15 interrupt SW set register in test mode. 0:No effect 1:Set
14	W	0x0	Interrupt source 14 interrupt SW set register in test mode. 0:No effect 1:Set
13	W	0x0	Interrupt source 13 interrupt SW set register in test mode. 0:No effect 1:Set
12	W	0x0	Interrupt source 12 interrupt SW set register in test mode. 0:No effect 1:Set
11	W	0x0	Interrupt source 11 interrupt SW set register in test mode. 0:No effect 1:Set
10	W	0x0	Interrupt source 10 interrupt SW set register in test mode. 0:No effect 1:Set
9	W	0x0	Interrupt source 9 interrupt SW set register in test mode. 0:No effect 1:Set
8	W	0x0	Interrupt source 8 interrupt SW set register in test mode. 0:No effect 1:Set
7	W	0x0	Interrupt source 7 interrupt SW set register in test mode. 0:No effect 1:Set
6	W	0x0	Interrupt source 6 interrupt SW set register in test mode. 0:No effect 1:Set
5	W	0x0	Interrupt source 5 interrupt SW set register in test mode. 0:No effect 1:Set
4	W	0x0	Interrupt source 4 interrupt SW set register in test mode. 0:No effect 1:Set
3	W	0x0	Interrupt source 3 interrupt SW set register in test mode. 0:No effect 1:Set
2	W	0x0	Interrupt source 2 interrupt SW set register in test mode. 0:No effect 1:Set
1	W	0x0	Interrupt source 1 interrupt SW set register in test mode. 0:No effect 1:Set
0	W	0x0	Interrupt source 0 interrupt SW set register in test mode. 0:No effect 1:Set

INTC_TEST

Address: Operational Base + offset(0x124)

This register is used by software to control the operation mode of interrupt. It allows software to issue interrupts by programming INTC_ISCR register.

Test Mode Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved.
5	RW	0x0	FIQ level.

			1: Low active 0: High active
4	RW	0x0	IRQ level 1: Low active 0: High active
3:1	-	-	Reserved.
0	RW	0x0	Test Mode Register. 1: Test mode enable; 0: Test mode disable. When the TESTMODE is enabled, it allows software to issue interrupts by programming INTC_ISCR register. It is useful for software testing and debug.

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

6.4 Functional Description

6.4.1 Operation

The interrupt controller provides a simple software interface to the interrupt system. Certain interrupt bits must be defined according to the system requirements.

Priority Controller

A 32-level priority encoder controls the output of IRQ. Each source has a programmable priority level of 31 to 0. Level 31 is the highest priority and level 0 the lowest. When the interrupt controller receives more than one unmasked interrupt at a time, the interrupt with the highest priority is serviced first. If both interrupts have equal priority, the interrupt with the highest interrupt source number (see the Table given below) is serviced first.

Interrupt Enabling and Masking

Each interrupt source can be enabled or disabled using the command register INTC_IECR. The interrupt mask can be masked or unmasked by configuring the register INTC_IMR. A disabled interrupt does not affect the servicing of other interrupts.

Interrupt Clearing and Setting

All interrupt sources that are programmed to be edge triggered can be individually set or cleared by respectively writing to the registers INTC_ISCR and INTC_ICCR. This function of the interrupt controller is available for auto-test or software debug purposes.

Chapter 7 AHB DMA (HDMA)

7.1 Design Overview

7.1.1 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.

7.1.2 Features

- 2 x AHB master and 1 x slave interfaces
- 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

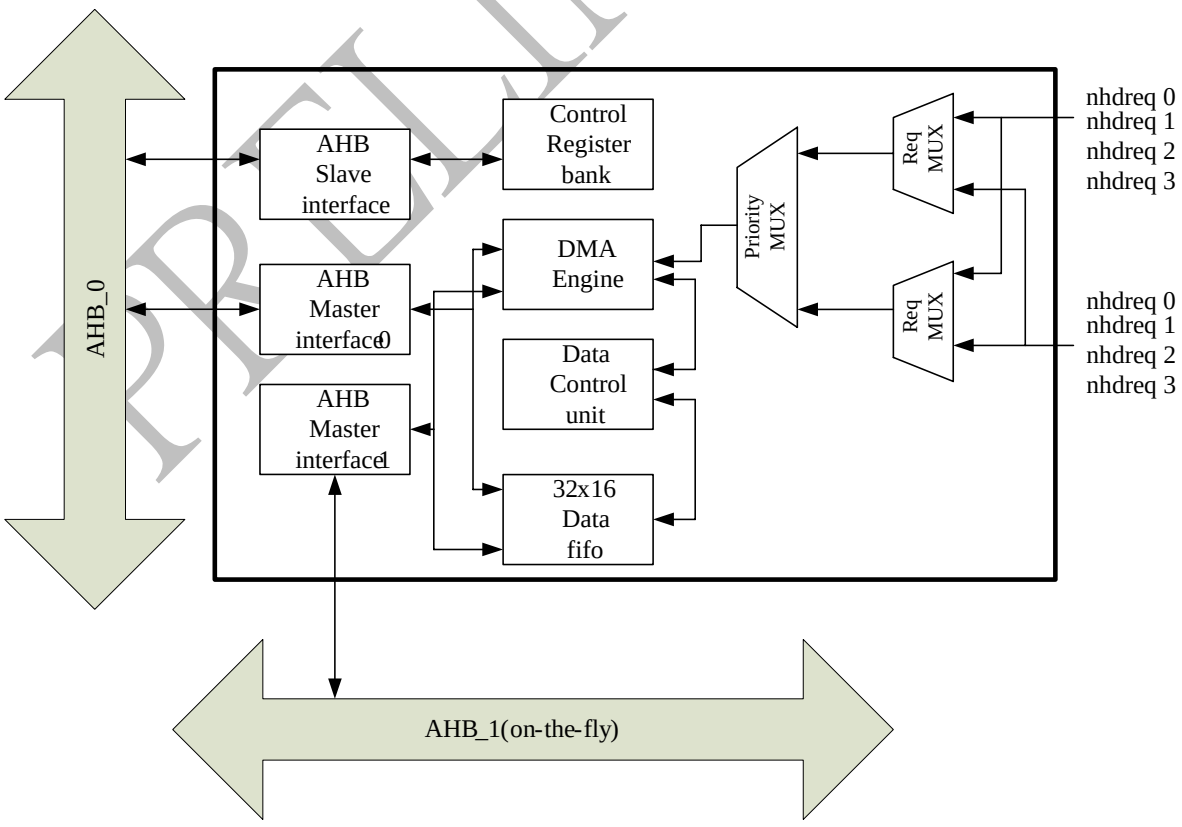
  Support slice transfer base on peripheral FIFO depth

  Support turnaround start address when over address buffer size

7.2 Architecture

This section provides a description about the functions and behavior under various conditions.

7.2.1 Block Diagram



7.2.2 Block Descriptions

AHB DMA consists of one AHB slave interface and two AHB master interfaces.
AHB slave interface is used for registers setting and AHB master interface 0 issues command from DMA engine to AHB_0, and AHB master interface 1 issues command from DMA engine when on-the-fly function is used, two request MUXs select the corresponding hardware request port and the priority MUX selects which channel is served by DMA engine.

7.3 Registers

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

7.3.1 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_ISR	0x0038	W	0x00000000	HDMA interrupt status register
HDMA_DSR	0x003C	W	0x00000000	HDMA DMA 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_BS1	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_BS0	0x0060	W	0x00000000	HDMA channel 0 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_BS1	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

7.3.2 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, set this bit to 1 also disable page mode 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	-	-	Reserved.
19:18	RW	0x0	Interrupt Mode Set. 00: Polling mode 01: Interrupt mode
17:16	RW	0x0	On the fly mode. 0x0: Disable on the fly 0x1: Read on the fly(Read from OFM bus) 0x2: Write on the fly(Write to OFM bus) 0x3: 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. 0x0: From UART0 rxd 0x1: From UART0 txd 0x2: From UART1 rxd 0x3: From UART1 txd 0x4: From SPI rxd 0x5: From SPI txd 0x6: From I2S txd 0x7: From I2S rxd
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 ox2 ~ 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_ISRCx(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_CSRCx(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_IDSTx(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_CDSTx(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_ICNTx(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. (Unit: data size) the transfer length is depend the Terminate count and Data size of HDMA_CONx 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. (Unit : data size)

HDMA_ISR
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
7:6	-	-	Reserved.
5	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
4	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
3	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
2	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
1	RW	0x0	Channel 1 Interrupt active, 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_DSR
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

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) Slice count must be divided by terminate count.

HDMA_CSCNTx(x=0, 1)
Address: Operational Base + offset(058x, 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.

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) This value must be divided by silice count register.

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

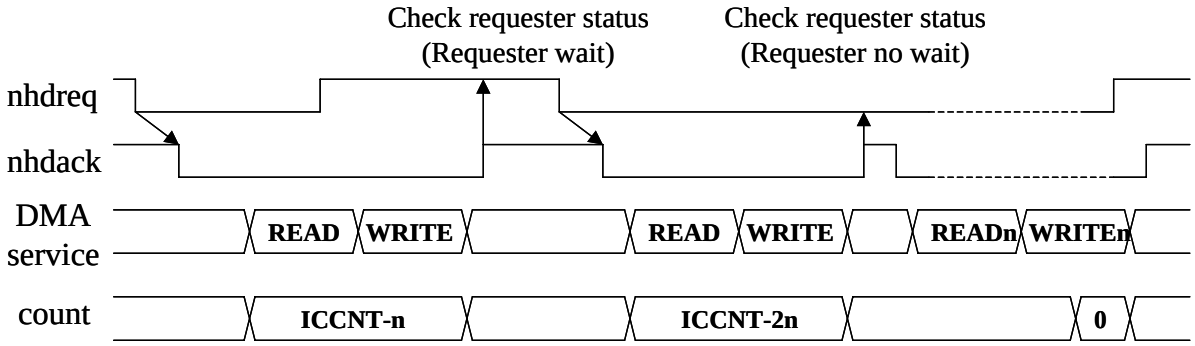
7.4 Functional Description

7.4.1 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.

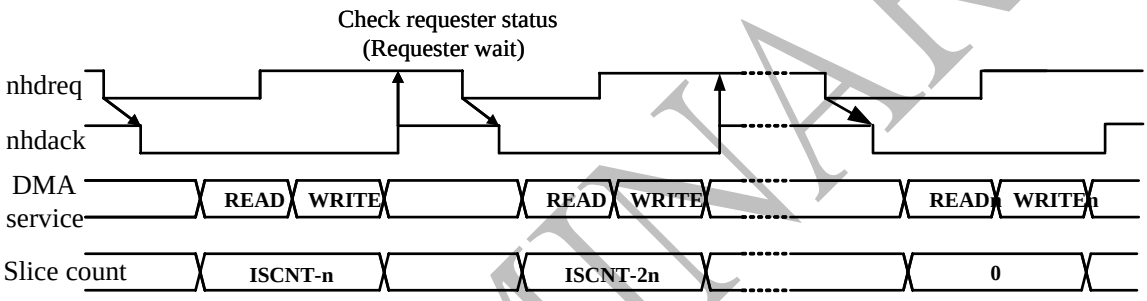
7.4.2 H/W Trigger DMA Mode

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.

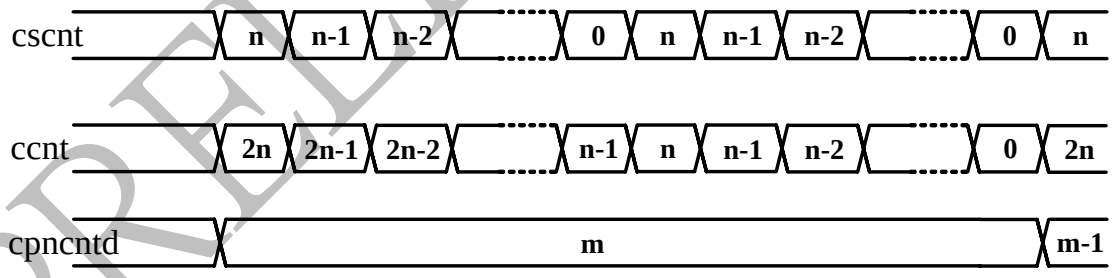


H/W Trigger DMA with Slice Mode

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. The transfer will auto reload if source address buffer size is reach and page count number is no zero.



7.4.3 On-the-Fly DMA Mode

To enhance AHB bus performance, the HDMA supports On-the-Fly data transfer of DMA operation at both SW and HW mode. When On-the-Fly read bit is set in DMA channel control register, HDMA will read data from On-the-Fly AHB bus and write data to AHB bus. When On-the-Fly write bit is set in DMA channel control register, HDMA will read data from AHB bus and write data to On-the-Fly AHB bus.

7.5 Application Notes

- The A2A DMA only support the same size for source and destination peripheral.
- In RK27xx, on-the-fly function of HDMA is disabled.
- The following is example step by step for software :

```
*(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_ISRC0_REG) = sar;    //source address
*(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_IDST0_REG) = dst;   //destination address
*(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_ICNT0_REG) = 0xf;   //transfer length
*(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_ISR_REG) = (((unsigned long )0xf3)<<8);
        //两个通道的中断不mask
*(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_IPNCNTD0_REG) = 0x1;
*(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_CON0_REG) =
((0x2)|((0x2)<<3)|(0x3<<13)|(0x1<<18)|(0x3<<21));                    // enable dma

 Wait generation of interrupt……
 Read interrupt source to decide which channel is interrupt from as follows
 temp = *(unsigned long volatile *)(HDMA_ARM_BASE + HDMA_ISR_REG);
 if(temp&0x4){
        channel 1 ….
     }
```

Chapter 8 DW DMA

8.1 Design Overview

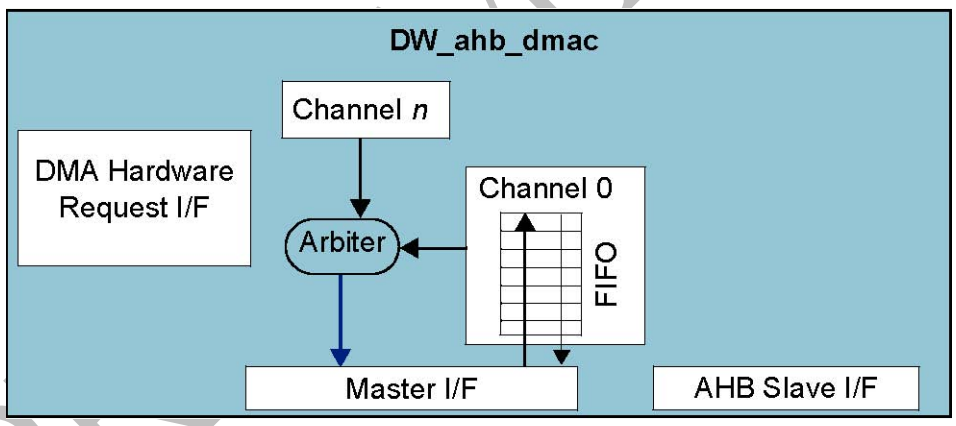
8.1.1 Overview

The DW DMA controller provides an interface to translate data between SDR/ST and DSP subsystem. It can support all kinds of burst type and data size, which define in AHB protocol. It can support on-the-fly DMA transfer on DSP AHB bus devices and SDRAM controller or DSP AHB bus data transfer.

8.1.2 Features

- Provide AHB-to-AHB bus protocol translation
- Support AHB-to-AHB DMA or Single AHB DMA
- Four DMA channels supported
- Support byte, half word and word data transfer sizes
- Support incremental and fixed addressing mode
- Support block and software DMA transfer mode
- Support all burst type transfer for bridge
- Support SINGLE and all incremental burst type transfer for DMA
- Support fixed channel priority arbitration
- Scatter/Gather transfer support
- LLP transfer support

8.2 Architecture



8.3 Registers

8.3.1 Registers Summary

Name	Address Offset	Width	R/W	Description
Channel Registers, x = channels number , (0 ~3)				
SARx	0x000	64	R/W	Channel x Source Address Register Reset Value: 0x0
DARx	0x008	64	R/W	Channel x Destination Address Register Reset Value: 0x0
LLPx	0x010	64	R/W	Channel x Linked List Pointer Register Reset Value: 0x0
CTLx	0x018	64	R/W	Channel x Control Register Reset Value: Configuration dependent

SSTATx	0x020	64	R/W	Channel x Source Status Register Reset Value: 0x0
DSTATx	0x028	64	R/W	Channel x Destination Status Register Reset Value: 0x0
SSTATARx	0x030	64	R/W	Channel x Source Status Address Register Reset Value: 0x0
DSTATARx	0x038	64	R/W	Channel x Destination Status Address Register Reset Value: 0x0
CFGx	0x040	64	R/W	Channel x Configuration Register Reset Value: 0x0000000400000c00
SGRx	0x048	64	R/W	Channel x Source Gather Register Reset Value: 0x0
DSRx	0x050	64	R/W	Channel x Destination Scatter Register Reset Value: 0x0
Interrupt Registers				
RawTfr	0x2c0	64	R	Raw Status for IntTfr Interrupt Reset Value: 0x0
RawBlock	0x2c8	64	R	Raw Status for IntBlock Interrupt Reset Value: 0x0
RawSrcTran	0x2d0	64	R	Raw Status for IntSrcTran Interrupt Reset Value: 0x0
RawDstTran	0x2d8	64	R	Raw Status for IntDstTran Interrupt Reset Value: 0x0
RawErr	0x2e0	64	R	Raw Status for IntErr Interrupt Reset Value: 0x0
StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr				
StatusTfr	0x2e8	64	R	Status for IntTfr Interrupt Reset Value: 0x0
StatusBlock	0x2f0	64	R	Status for IntBlock Interrupt Reset Value: 0x0
StatusSrcTran	0x2f8	64	R	Status for IntSrcTran Interrupt Reset Value: 0x
StatusDstTran	0x300	64	R	Status for IntDstTran Interrupt Reset Value: 0x0
StatusErr	0x308	64	R	Status for IntErr Interrupt Reset Value: 0x0
MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr				
MaskTfr	0x310	64	R/W	Mask for IntTfr Interrupt Reset Value: 0x0
MaskBlock	0x318	64	R/W	Mask for IntBlock Interrupt Reset Value: 0x0
MaskSrcTran	0x320	64	R/W	Mask for IntSrcTran Interrupt Reset Value: 0x0
MaskDstTran	0x328	64	R/W	Mask for IntDstTran Interrupt Reset Value: 0x0
MaskErr	0x330	64	R/W	Mask for IntErr Interrupt Reset Value: 0x0
ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr				
ClearTfr	0x338	64	W	Clear for IntTfr Interrupt Reset Value: 0x0
ClearBlock	0x340	64	W	Clear for IntBlock Interrupt Reset Value: 0x0
ClearSrcTran	0x348	64	W	Clear for IntSrcTran Interrupt Reset Value: 0x0
ClearDstTran	0x350	64	W	Clear for IntDstTran Interrupt Reset Value: 0x0
ClearErr	0x358	64	W	Clear for IntErr Interrupt Reset Value: 0x0

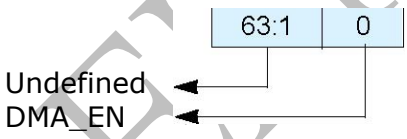
StatusInt	0x360	64	W	Status for each interrupt type Reset Value: 0x0
Software Handshaking Registers				
ReqSrcReg	0x368	64	R/W	Source Software Transaction Request Register Reset Value: 0x0
ReqDstReg	0x370	64	R/W	Destination Software Transaction Request Register Reset Value: 0x0
SglReqSrcReg	0x378	64	R/W	Single Source Transaction Request Register Reset Value: 0x0
SglReqDstReg	0x380	64	R/W	Single Destination Transaction Request Register Reset Value: 0x0
LstSrcReg	0x388	64	R/W	Last Source Transaction Request Register Reset Value: 0x0
LstDstReg	0x390	64	R/W	Last Destination Transaction Request Register Reset Value: 0x0
Miscellaneous Registers				
DmaCfgReg	0x398	64	R/W	DMA Configuration Register Reset Value: 0x0
ChEnReg	0x3a0	64	R/W	DMA Channel Enable Register Reset Value: 0x0

8.3.2 Configuration and Channel Enable Registers

DmaCfgReg

- **Name:** DW_ahb_dmac Configuration Register
- **Size:** 64 bits
- **Address Offset:** 0x398
- **Read / Write Access:** Read/Write

This register is used to enable the DW_ahb_dmac, which must be done before any channel activity can begin.



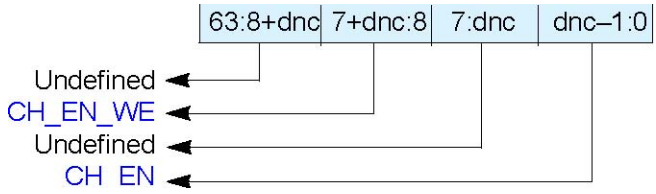
Bits	Name	R/W	Reset	Description
63:1	Undefined	N/A	0x0	Reserved
0	DMA_EN	R/W	0x0	**DW_ahb_dmac Enable bit.** 0 = DW_ahb_dmac Disabled 1 = DW_ahb_dmac Enabled.

If the global channel enable bit is cleared while any channel is still active, then DmaCfgReg.DMA_EN still returns 1 to indicate that there are channels still active until hardware has terminated all activity on all channels, at which point the DmaCfgReg.DMA_EN bit returns 0.

ChEnReg

- **Name:** DW_ahb_dmac Channel Enable Register
- **Size:** 64 bits
- **Address Offset:** 0x3a0
- **Read / Write Access:** Read/Write

This is the DW_ahb_dmac Channel Enable Register. If software needs to set up a new channel, then it can read this register in order to find out which channels are currently inactive; it can then enable an inactive channel with the required priority.



Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	CH_EN_WE	W	0x0	**Channel enable write enable.**
7:4	Undefined	N/A	0x0	Reserved
3:0	CH_EN	R/W	0x0	**Enables/Disables the channel.** 0 = Disable the Channel 1 = Enable the Channel The ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer.

All bits of this register are cleared to 0 when the global DW_ahb_dmac channel enable bit, DmaCfgReg[0], is 0. When the global channel enable bit is 0, then a write to the ChEnReg register is ignored and a read will always read back 0.

The channel enable bit, ChEnReg.CH_EN, is written only if the corresponding channel write enable bit, ChEnReg.CH_EN_WE, is asserted on the same AHB write transfer. For example, writing hex 01x1 writes a 1 into ChEnReg[0], while ChEnReg[7:1] remains unchanged. Writing hex 00xx leaves ChEnReg[7:0] unchanged. Note that a read-modified write is not required.

8.3.3 Channel Registers

The channel registers consist of the following, where x = 0 to 3:

- **CFGx** – Configuration register for channel x
- **CTLx** – Control register for channel x
- **DARx** – Destination address register for channel x
- **DSRx** – Destination scatter register for channel x
- **DSTATx** – Destination status register for channel x
- **DSTATARx** – Destination status address register for channel x
- **LLPx** – Linked list pointer register for channel x
- **SARx** – Source address register for channel x
- **SGRx** – Source gather register for channel x
- **SSTATx** – Source status register for channel x
- **SSTATARx** – Source status address register for channel x

The SARx, DARx, LLPx, CTLx, and CFGx channel registers should be programmed prior to enabling the channel. However, if an LLI update occurs before commencing data transfer, SARx and DARx may not need to be programmed prior to enabling the channel; refer to rows 6 to 10 in Table 5. It is an Illegal Register Access when a write to the SARx, DARx, LLPx, CTLx, SSTATx, DSTATx, SSTATARx, DSTATARx, SGRx, or DSRx registers occurs when the channel is enabled.

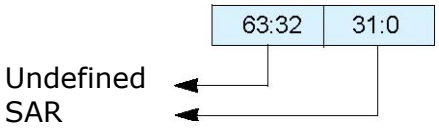
SARx

- Name: Source Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  SAR0 – 0x000
  SAR1 – 0x058
  SAR2 – 0x0b0
  SAR3 – 0x108
- Read/Write Access: Read/Write

The starting source address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the source address of the current AHB transfer.

Note

You must program the SAR address to be aligned to CTLx.SRC_TR_WIDTH.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31:0	SAR	R/W	0x0	**Current Source Address of DMA transfer.** Updated after each source transfer. The SINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

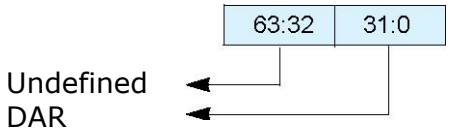
DARx

- Name: Destination Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 7:
  DAR0 – 0x008
  DAR1 – 0x060
  DAR2 – 0x0b8
  DAR3 – 0x110
- Read/Write Access: Read/Write

The starting destination address is programmed by software before the DMA channel is enabled, or by an LLI update before the start of the DMA transfer. While the DMA transfer is in progress, this register is updated to reflect the destination address of the current AHB transfer.

Note

You must program the DAR to be aligned to CTLx.DST_TR_WIDTH.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved

31:0	DAR	R/W	0x0	**Current Destination address of DMA transfer.** Updated after each destination transfer. The DINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

Hardware Realignment of SAR/DAR Registers

In a particular cicumstance, during contiguous multi-block DMA transfers, the destination address can become misaligned between the end of one block and the start of the next block. When this situation occurs, DW_ahb_dmac re-aligns the destination address before the start of the next block.

Consider the following example. If the block length is 9, the source transfer width is 16 (halfword), and the destination transfer width is 32 (word)—the destination is programmed for contiguous block transfers—then the destination performs four word transfers followed by a halfword transfer to complete the block transfer to the destination. At the end of the destination block transfer, the address is aligned to a 16-bit transfer as the last AMBA transfer is halfword. This is misaligned to the programmed transfer size of 32 bits for the destination. However, for contiguous destination multi-block transfers, DW_ahb_dmac re-aligns the DAR address to the nearest 32-bit address (next 32-bit address upwards if address control is incrementing or next address downwards if address control is decrementing).

This only occurs when the following is the DMA transfer setup:

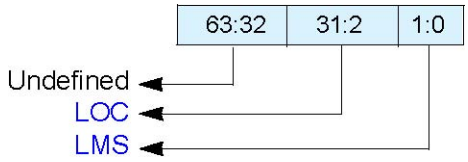
- When on the DAR address, OR
- Contiguous multi-block transfers on destination side, OR
- DST_TR_WIDTH > SRC_TR_WIDTH, OR
- (BLOCK_TS * SRC_TR_WIDTH)/DST_TR_WIDTH != integer (where SRC_TR_WIDTH, DST_TR_WIDTH is byte width of transfer)

LLPx

- Name: Linked List Pointer Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  LLP0 – 0x010
  LLP1 – 0x068
  LLP2 – 0x0c0
  LLP3 – 0x118
- Read/Write Access: Read/Write

Note

You need to program this register to point to the first Linked List Item (LLI) in memory prior to enabling the channel if block chaining is enabled.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31:2	LOC	R/W	0x0	**Starting Address** In Memory of next LLI if block chaining is enabled. Note that the two LSBs of the starting address are not stored because the address is assumed to be aligned to a 32-bit boundary.

1:0	LMS	R/W	0x0	**List Master Select.** Identifies the AHB layer/interface where the memory device that stores the next linked list item resides. 00 = AHB master 1 01 = AHB master 2

The LLP register has two functions:

- The logical result of the equation LLP.LOC != 0 is used to set up the type of DMA transfer—single or multi-block. If LLP.LOC is set to 0x0, then transfers using linked lists are not enabled. This register must be programmed prior to enabling the channel in order to set up the transfer type.
- LLP.LOC != 0 contains the pointer to the next LLI for block chaining using linked lists; The LLPx register can also point to the address where write-back of the control and source/destination status information occur after block completion.

CTLx

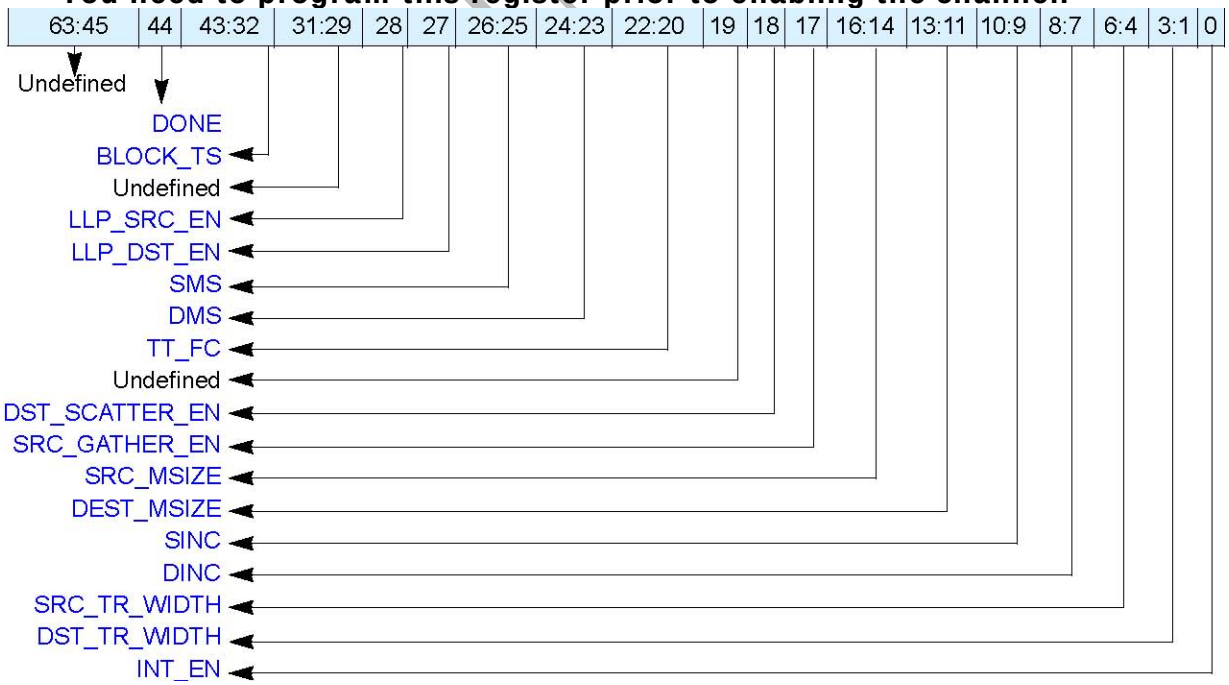
- Name: Control Register for Channel x
- Size: 64 bits
- Address Offset: for x = 0 to 7:
  CTL0 – 0x018
  CTL1 – 0x070
  CTL2 – 0x0c8
  CTL3 – 0x120
- Read/Write Access: Read/Write

This register contains fields that control the DMA transfer.

The CTLx register is part of the block descriptor (linked list item – LLI) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled.If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the LLI in system memory at the end of the block transfer.

Note

You need to program this register prior to enabling the channel.



Bits	Name	R/W	Description
63:45	Undefined	N/A	Reserved

44	DONE	R/W	**Done bit** If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set. Software can poll the LLI CTLx.DONE bit to see when a block transfer is complete. The LLI CTLx.DONE bit should be cleared when the linked lists are set up in memory prior to enabling the channel.
b:32	BLOCK_TS	R/W	**Block Transfer Size.** When the DW_ahb_dmac is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat. Width: The width of the single transaction is determined by CTLx.SRC_TR_WIDTH. Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller. When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at DMAH_CHx_MAX_BLK_SIZE, but the actual block size can be greater. b = log2(DMAH_CHx_MAX_BLK_SIZE + 1) + 31 Bits 43:b+1 do not exist and return 0 on a read. **Reset Value:** 0x2
31:29	Undefined	N/A	Reserved
28	LLP_SRC_EN	R/W	Block chaining is enabled on the source side only if the LLP_SRC_EN field is high and LLPx.LOC is non-zero; **Reset Value:** 0x0
27	LLP_DST_EN	R/W	Block chaining is enabled on the destination side only if the LLP_DST_EN field is high and LLPx.LOC is non-zero. **Reset Value:** 0x0
26:25	SMS	R/W	Source Master Select. Identifies the Master Interface layer from which the source device (peripheral or memory) is accessed. 00 = AHB master 1 01 = AHB master 2 **Reset Value:** DMAH_CHx_SMS[1:0]
24:23	DMS	R/W	Destination Master Select. Identifies the Master Interface layer where the destination device (peripheral or memory) resides. 00 = AHB master 1 01 = AHB master 2 **Reset Value:** DMAH_CHx_DMS[1:0]
22:20	TT_FC	R/W	Transfer Type and Flow Control. The following transfer types are supported. • Memory to Memory • Memory to Peripheral • Peripheral to Memory • Peripheral to Peripheral Flow Control can be assigned to the DW_ahb_dmac, the source peripheral, or the destination peripheral. Table 3 lists the decoding for this field. **Reset Value:** Configuration dependent: TT_FC[0] = 1'b1 TT_FC[1] = DMAH_CHx_FC[1] || (!DMAH_CHx_FC[0]) TT_FC[2] = DMAH_CHx_FC[1] ^ DMAH_CHx_FC[0] **Dependencies:** If the configuration parameter DMAH_CHx_FC is set to DMA_FC_ONLY, then TT_FC[2] does not exist and TT_FC[2] always reads back 0. If DMAH_CHx_FC is set to SRC_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b10. If DMAH_CHx_FC is set to DST_FC_ONLY, then TT_FC[2:1] does not exist and TT_FC[2:1] always reads back 2'b11.

19	Undefined	N/A	Reserved
18	DST_SCATTER_EN	R/W	Destination scatter enable bit: 0 = Scatter disabled 1 = Scatter enabled Scatter on the destination side is applicable only when the CTLx.DINC bit indicates an incrementing or decrementing address control. **Reset Value:** 0x0
17	SRC_GATHER_EN	R/W	Source gather enable bit: 0 = Gather disabled 1 = Gather enabled Gather on the source side is applicable only when the CTLx.SINC bit indicates an incrementing or decrementing address control. **Reset Value:** 0x0
16:14	SRC_MSIZE	R/W	Source Burst Transaction Length. Number of data items, each of width CTLx.SRC_TR_WIDTH, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface. Table 1 lists the decoding for this field; **NOTE:** This value is not related to the AHB bus master HBURST bus. **Reset Value:** 0x1
13:11	DEST_MSIZE	R/W	Destination Burst Transaction Length. Number of data items, each of width CTLx.DST_TR_WIDTH, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface. **NOTE:** This value is not related to the AHB bus master HBURST bus. **Reset Value:** 0x1
10:9	SINC	R/W	Source Address Increment. Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change **NOTE:** Incrementing or decrementing is done for alignment to the next CTLx.SRC_TR_WIDTH boundary. **Reset Value:** 0x0
8:7	DINC	R/W	Destination Address Increment. Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change **NOTE:** Incrementing or decrementing is done for alignment to the next CTLx.DST_TR_WIDTH boundary. **Reset Value:** 0x0
6:4	SRC_TR_WIDTH	R/W	Source Transfer Width. Table 2 lists the decoding for this field. Mapped to AHB bus "hsize." For a non-memory peripheral, typically the peripheral (source) FIFO width. This value must be less than or equal to DMAH_Mx_HDATA_WIDTH, where x is the AHB layer 1 to 2 where the source resides. **Reset Value**: Encoded value; refer to Table 2.

3:1	DST_TR_WIDTH	R/W	Destination Transfer Width. Table 2 lists the decoding for this field. Mapped to AHB bus "hsize." For a non-memory peripheral, typically rgw peripheral (destination) FIFO width. This value must be less than or equal to DMAH_Mk_HDATA_WIDTH, where k is the AHB layer 1 to 2 where the destination resides. Reset Value: Encoded value; refer to Table 2.
0	INT_EN	R/W	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. Reset Value: 0x1

Table 1: CTLx.SRC_MSIZE and DEST_MSIZE Decoding

CTLx.SRC_MSIZE / CTLx.DEST_MSIZE	Number of data items to be transferred (of width CTLx.SRC_TR_WIDTH or CTLx.DST_TR_WIDTH)
000	1
001	4
010	8
011	16
100	32

Table 2: CTLx.SRC_TR_WIDTH and CTLx.DST_TR_WIDTH Decoding

CTLx.SRC_TR_WIDTH / CTLx.DST_TR_WIDTH	Size (bits)
000	8
001	16
010	32

Table 3: CTLx.TT_FC Field Decoding

CTLx.TT_FC Field	Transfer Type	Flow Controller
000	Memory to Memory	DW_ahb_dmac
001	Memory to Peripheral	DW_ahb_dmac
010	Peripheral to Memory	DW_ahb_dmac
011	Peripheral to Peripheral	DW_ahb_dmac
100	Peripheral to Memory	Peripheral
101	Peripheral to Peripheral	Source Peripheral
110	Memory to Peripheral	Peripheral
111	Peripheral to Peripheral	Destination Peripheral

SSTATx

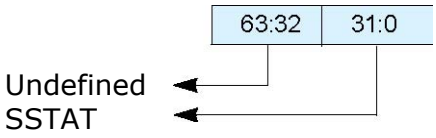
- Name: Source Status Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  SSTAT0 – 0x020
  SSTAT1 – 0x078
  SSTAT2 – 0x0d0
  SSTAT3 – 0x128
- Read/Write Access: Read/Write

After each block transfer completes, hardware can retrieve the source status information from the address pointed to by the contents of the SSTATARx register. This status information is then stored in the SSTATx register and written out to the SSTATx register location of the LLI before the start of the next block. For conditions under which the source status information is fetched.

Note

This register is a temporary placeholder for the source status information on

its way to the SSTATx register location of the LLI. The source status information should be retrieved by software from the SSTATx register location of the LLI, and not by a read of this register over the DW_ahb_dmac slave interface.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31:0	SSTAT	R/W	0x0	Source status information retrieved by hardware from the address pointed to by the contents of the SSTATARx register.

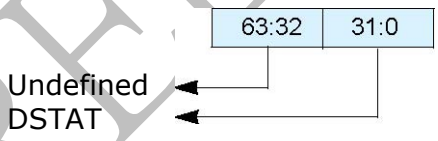
DSTATx

- Name: Destination Status Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  DSTAT0 – 0x028
  DSTAT1 – 0x080
  DSTAT2 – 0x0d8
  DSTAT3 – 0x130
- • Read/Write Access: Read/Write

After the completion of each block transfer, hardware can retrieve the destination status information from the address pointed to by the contents of the DSTATARx register. This status information is then stored in the DSTATx register and written out to the DSTATx register location of the LLI before the start of the next block. if the parameter DMAH_CHx_STAT_DST is set to False; in this case, the read-back value is always 0.

Note

This register is a temporary placeholder for the destination status information on its way to the DSTATx register location of the LLI. The destination status information should be retrieved by software from the DSTATx register location of the LLI and not by a read of this register over the DW_ahb_dmac slave interface.



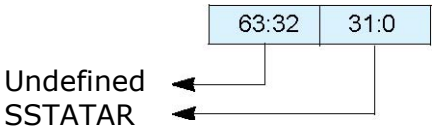
Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31:0	DSTAT	R/W	0x0	Destination status information retrieved by hardware from the address pointed to by the contents of the DSTATARx register.

SSTATARx

- Name: Source Status Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  SSTATAR0 – 0x030
  SSTATAR1 – 0x088
  SSTATAR2 – 0x0e0
  SSTATAR3 – 0x138
- Read/Write Access: Read/Write

After the completion of each block transfer, hardware can retrieve the source status information from the address pointed to by the contents of the SSTATARx register.

This register does not exist if the configuration parameter DMAH_CHx_STAT_SRC is set to False; in this case, the read-back value is always 0.

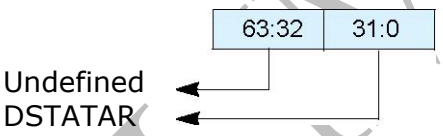


Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31-1:0	SSTATAR	R/W	0x0	Pointer from where hardware can fetch the source status information, which is registered in the SSTATx register and written out to the SSTATx register location of the LLI before the start of the next block.

DSTATARx

- Name: Destination Status Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  DSTATAR0 – 0x038
  DSTATAR1 – 0x090
  DSTATAR2 – 0x0e8
  DSTATAR3
- Read/Write Access: Read/Write

After the completion of each block transfer, hardware can retrieve the destination status information from the address pointed to by the contents of the DSTATARx register. This register does not exist if the configuration parameter DMAH_CHx_STAT_DST is set to False; in this case, the read-back value is always 0.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved
31-1:0	DSTATAR	R/W	0x0	Pointer from where hardware can fetch the destination status information, which is registered in the DSTATx register and written out to the DSTATx register location of the LLI before the start of the next block.

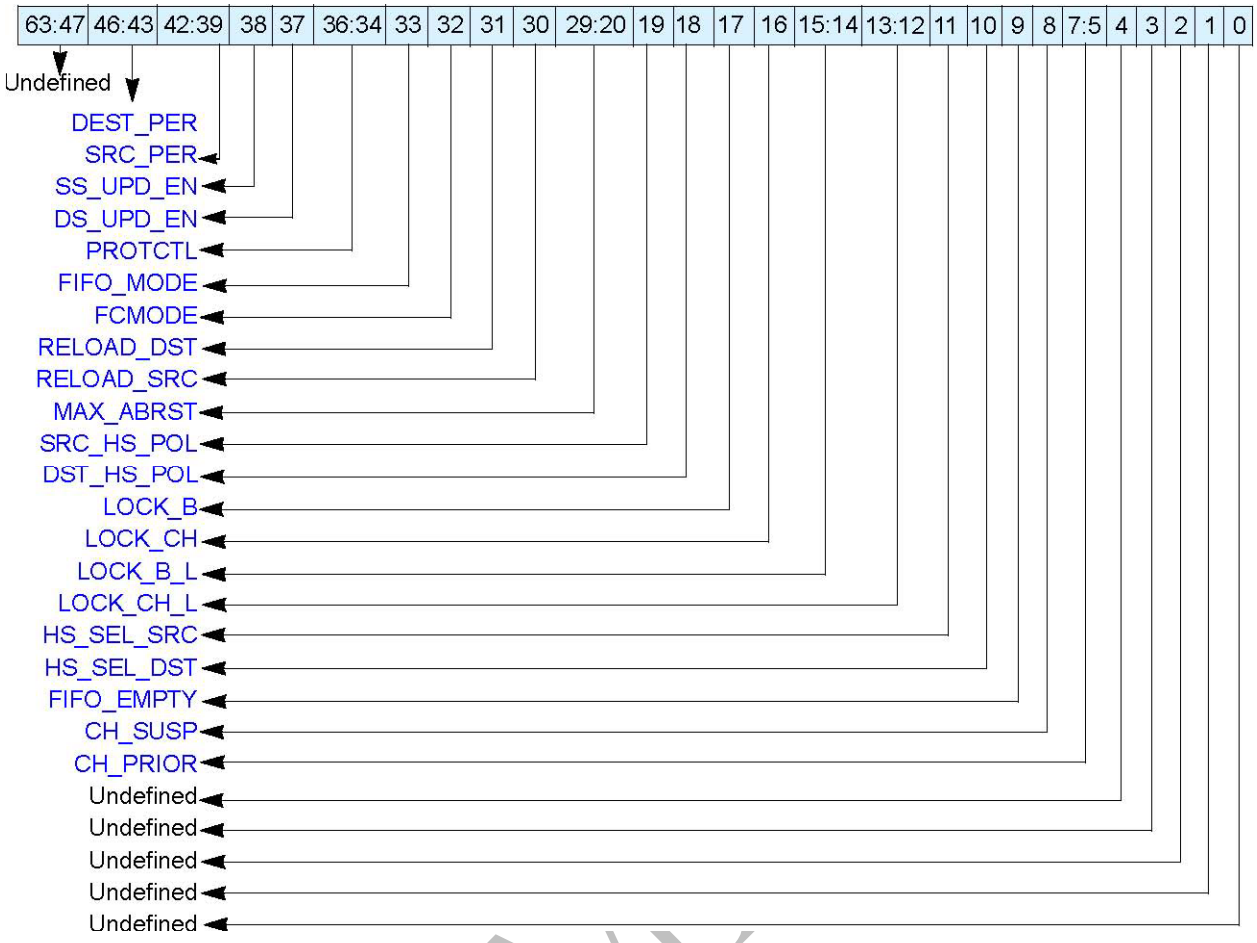
CFGx

- Name: Configuration Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  CFG0 – 0x040
  CFG1 – 0x098
  CFG2 – 0x0f0
  CFG3 – 0x148
- Read/Write Access: Read/Write

This register contains fields that configure the DMA transfer. The channel configuration register remains fixed for all blocks of a multi-block transfer.

Note

You need to program this register prior to enabling the channel.



Bits	Name	R/W	Reset	Description
63:47	Undefined	N/A	0x0	Reserved
46:43	DEST_PER	R/W	0x0	Assigns a hardware handshaking interface (0 -DMAH_NUM_HS_INT-1) to the destination of channel x if the CFGx.HS_SEL_DST field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. **NOTE:** For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
42:39	SRC_PER	R/W	0x0	Assigns a hardware handshaking interface (0 -DMAH_NUM_HS_INT-1) to the source of channel x if the CFGx.HS_SEL_SRC field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface. **NOTE:** For correct DW_ahb_dmac operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
38	SS_UPD_EN	R/W	0x0	**Source Status Update Enable.** Source status information is fetched only from the location pointed to by the SSTATARx register, stored in the SSTATx register and written out to the SSTATx location of the LLI if SS_UPD_EN is high. **NOTE:** This enable is applicable only if DMAH_CHx_STAT_SRC is set to True.

37	DS_UPD_EN	R/W	0x0	**Destination Status Update Enable.** Destination status information is fetched only from the location pointed to by the DSTATARx register, stored in the DSTATx register and written out to the DSTATx location of the LLI if DS_UPD_EN is high.
36:34	PROTCTL	R/W	0x1	Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. Table 4 shows the mapping of bits in this field to the AHB HPROT[3:1] bus.
33	FIFO_MODE	R/W	0x0	**FIFO Mode Select.** Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Space/data available is greater than or equal to half the FIFO depth for destination transfers and less than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.
32	FCMODE	R/W	0x0	**Flow Control Mode.** Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller. 0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled. 1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.
31	RELOAD_DST	R/W	0x0	**Automatic Destination Reload.** The DARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. For conditions under which this occurs, refer to Table 5.
30	RELOAD_SRC	R/W	0x0	**Automatic Source Reload.** The SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated. For conditions under which this occurs.
29:20	MAX_ABRST	R/W	0x0	**Maximum AMBA Burst Length.** Maximum AMBA burst length that is used for DMA transfers on this channel. A value of 0 indicates that software is not limiting the maximum AMBA burst length for DMA transfers on this channel.
19	SRC_HS_POL	R/W	0x0	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
18	DST_HS_POL	R/W	0x0	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
17	LOCK_B	R/W	0x0	**Bus Lock Bit.** When active, the AHB bus master signal hlock is asserted for the duration specified in CFGx.LOCK_B_L.

16	LOCK_CH	R/W	0x0	**Channel Lock Bit.** When the channel is granted control of the master bus interface and if the CFGx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in CFGx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in CFGx.LOCK_CH_L.
15:14	LOCK_B_L	R/W	0x0	**Bus Lock Level.** Indicates the duration over which CFGx.LOCK_B bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction
13:12	LOCK_CH_L	R/W	0x0	**Channel Lock Level.** Indicates the duration over which CFGx.LOCK_CH bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction
11	HS_SEL_SRC	R/W	0x1	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
10	HS_SEL_DST	R/W	0x1	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware- initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
9	FIFO_EMPTY	R	0x0	Indicates if there is data left in the channel FIFO. Can be used in conjunction with CFGx.CH_SUSP to cleanly disable a channel. 1 = Channel FIFO empty 0 = Channel FIFO not empty
8	CH_SUSP	R/W	0x0	Channel Suspend. Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMA transfer from the source.
7:5	CH_PRIOR	R/W	Channel Number example: Chan0=0 Chan1=1	Channel priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range: 0: (DMAH_NUM_CHANNELS – 1) A programmed value outside this range will cause erroneous behavior.
4:0	Undefined	N/A	0x0	Reserved

Table 4: PROTCTL field to HPROT Mapping

1'b1	HPROT[0]
CFGx.PROTCTL[1]	HPROT[1]

CFGx.PROTCTL[2] ->	HPROT[2]
CFGx.PROTCTL[3] ->	HPROT[3]

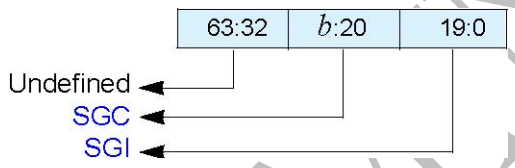
SGRx

- Name: Source Gather Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  SGR0 – 0x048
  SGR1 – 0x0a0
  SGR2 – 0x0f8
  SGR3 – 0x150
- Read/Write Access: Read/Write

The Source Gather register contains two fields:

- Source gather count field (SGRx.SGC) – Specifies the number of contiguous source transfers of CTLx.SRC_TR_WIDTH between successive gather intervals. This is defined as a gather boundary.
- Source gather interval field (SGRx.SGI) – Specifies the source address increment/decrement in multiples of CTLx.SRC_TR_WIDTH on a gather boundary when gather mode is enabled for the source transfer.

The CTLx.SINC field controls whether the address increments or decrements. When the CTLx.SINC field indicates a fixed-address control, then the address remains constant throughout the transfer and the SGRx register is ignored. This register does not exist if the configuration parameter DMAH_CHx_SRC_GAT_EN is set to False.



Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved.
b:20 See description	SGC	R/W	0x0	Source gather count. Source contiguous transfer count between successive gather boundaries. b = log2 (DMAH_CHx_MAX_BLK_SIZE + 1) + 19 Bits 31:b+1 do not exist and read back as 0.
19:0	SGI	R/W	0x0	Source gather interval.

DSRx

- Name: Destination Scatter Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 3:
  DSR0 – 0x050
  DSR1 – 0x0a8
  DSR2 – 0x100
  DSR3 – 0x158

- Read/Write Access: Read/Write

The Destination Scatter register contains two fields:

- Destination scatter count field (DSRx.DSC) – Specifies the number of contiguous destination transfers of CTLx.DST_TR_WIDTH between successive scatter boundaries.
- Destination scatter interval field (DSRx.DSI) – Specifies the destination address increment/ decrement in multiples of CTLx.DST_TR_WIDTH on a scatter boundary when scatter mode is enabled for the destination transfer.

The CTLx.DINC field controls whether the address increments or decrements. When the CTLx.DINC field indicates a fixed address control, then the address remains constant throughout the transfer and the DSRx register is ignored. This register does not exist if the configuration parameter DMAH_CHx_DST_SCA_EN is set to False.

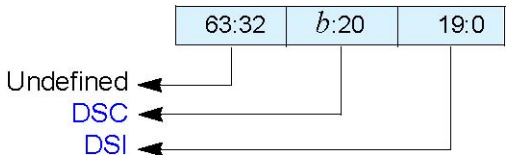


Figure 1: Destination Scatter Register Bit Fields for Channel x

Table 6: Description of Destination Scatter Register for Channel x

Bits	Name	R/W	Reset	Description
63:32	Undefined	N/A	0x0	Reserved.
b:20 See description	DSC	R/W	0x0	Destination scatter count. Destination contiguous transfer count between successive scatter boundaries. b = log2 (DMAH_CHx_MAX_BLK_SIZE + 1) + 19 Bits 31:b+1 do not exist and read 0.
19:0	DSI	R/W	0x0	Destination scatter interval.

8.3.4 Interrupt Registers

The following sections describe the registers pertaining to interrupts, their status, and how to clear them. For each channel, there are five types of interrupt sources:

- IntBlock – Block Transfer Complete Interrupt This interrupt is generated on DMA block transfer completion to the destination peripheral.
- IntDstTran – Destination Transaction Complete Interrupt

This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the destination side.

Note

If the destination for a channel is memory, then that channel will never generate the IntDstTran interrupt. Because of this, the corresponding bit in this field will not be set.

- IntErr – Error Interrupt
  This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.
- IntSrcTran – Source Transaction Complete Interrupt

This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the source side.

Note

If the source for a channel is memory, then that channel will never generate a IntSrcTran interrupt. Because of this, the corresponding bit in this field will not be set.

- IntTfr – DMA Transfer Complete Interrupt

This interrupt is generated on DMA transfer completion to the destination peripheral. There are several groups of interrupt-related registers:

- RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr
- StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr

- MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr
- ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr
- StatusInt

When a channel has been enabled to generate interrupts, the following is true:

- Interrupt events are stored in the Raw Status registers.
- The contents of the Raw Status registers are masked with the contents of the Mask registers.
- The masked interrupts are stored in the Status registers.
- The contents of the Status registers are used to drive the int_* port signals.
- Writing to the appropriate bit in the Clear registers clears an interrupt in the Raw Status registers and the Status registers on the same clock cycle.

The contents of each of the five Status registers is ORed to produce a single bit for each interrupt type in the Combined Status register; that is, StatusInt.

Note

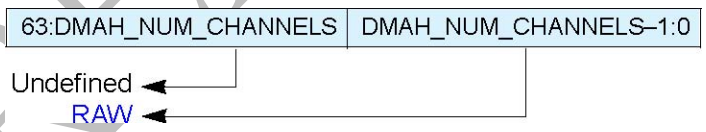
The CTLx.INT_EN bit must be set for an enabled channel to generate any interrupts.

RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr

- Name: Interrupt Raw Status Registers
- Size: 64 bits
- Address Offset:
  RawTfr – 0x2c0
  RawBlock – 0x2c8
  RawSrcTran – 0x2d0
  RawDstTran – 0x2d8
  RawErr – 0x2e0
- Read/Write Access: Read

Interrupt events are stored in these Raw Interrupt Status registers before masking: RawBlock, RawDstTran, RawErr, RawSrcTran, and RawTfr. Each Raw Interrupt Status register has a bit allocated per channel; for example, RawTfr[2] is the Channel 2 raw transfer complete interrupt.

Each bit in these registers is cleared by writing a 1 to the corresponding location in the ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr registers.

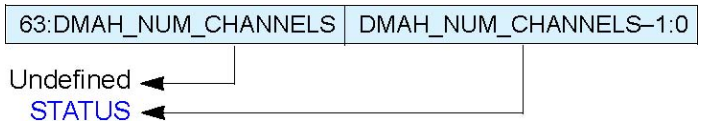


Bits	Name	R/W	Reset	Description
63:4	Undefined	N/A	0x0	Reserved
3:0	RAW	R	0x0	Raw interrupt status.

StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr

- Name: Interrupt Status Registers
- Size: 64 bits
- Address Offset:
  StatusTfr – 0x2e8
  StatusBlock – 0x2f0
  StatusSrcTran – 0x2f8
  StatusDstTran – 0x300
  StatusErr – 0x308
- Read/Write Access: Read

All interrupt events from all channels are stored in these Interrupt Status registers after masking: StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, and StatusTfr. Each Interrupt Status register has a bit allocated per channel; for example, StatusTfr[2] is the Channel 2 status transfer complete interrupt. The contents of these registers are used to generate the interrupt signals (int or int_n bus, depending on interrupt polarity) leaving the DW_ahb_dmac.



Bits	Name	R/W	Reset	Description
63:4	Undefined	N/A	0x0	Reserved
3:0	STATUS	R	0x0	Interrupt status.

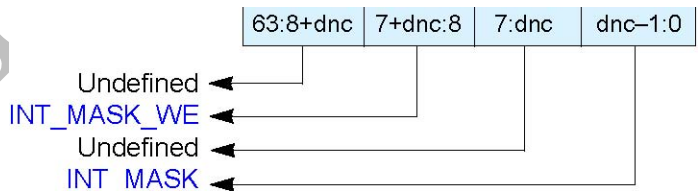
MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr

- Name: Interrupt Mask Registers
- Size: 64 bits
- Address Offset:
  MaskTfr – 0x310
  MaskBlock – 0x318
  MaskSrcTran – 0x320
  MaskDstTran – 0x328
  MaskErr – 0x330
- Read/Write Access: Read/Write

The contents of the Raw Status registers are masked with the contents of the Mask registers: MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, and MaskTfr. Each Interrupt Mask register has a bit allocated per channel; for example, MaskTfr[2] is the mask bit for the Channel 2transfer complete interrupt.

When the source peripheral of DMA channel i is memory, then the source transaction complete interrupt, MaskSrcTran[i], must be masked to prevent an erroneous triggering of an interrupt on the int_combined signal. Similarly, when the destination peripheral of DMA channel i is memory, then the destination transaction complete interrupt, MaskDstTran[i], must be masked to prevent an erroneous triggering of an interrupt on the int_combined(_n) signal.

A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation. For example, writing hex 01x1 to the MaskTfr register writes a 1 into MaskTfr[0], while MaskTfr[7:1] remains unchanged. Writing hex 00xx leaves MaskTfr[7:0] unchanged. Writing a 1 to any bit in these registers unmasks the corresponding interrupt, thus allowing the DW_ahb_dmac to set the appropriate bit in the Status registers and int_* port signals.



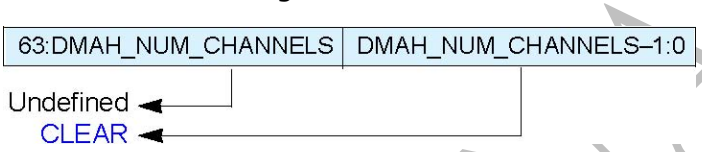
Bits	Name	R/W	Description
63:12	Undefined	N/A	**Reset Value:** 0x0
11:8	INT_MASK_WE	W	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled **Reset Value:** 0x0
7:4	Undefined	N/A	Reset Value: 0x0

3:0	INT_MASK	R/W	Interrupt Mask 0 = masked 1 = unmasked **Reset Value:** 0x0

ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr

- Name: Interrupt Clear Registers
- Size: 64 bits
- Address Offset:
  ClearTfr – 0x338
  ClearBlock – 0x340
  ClearSrcTran – 0x348
  ClearDstTran – 0x350
  ClearErr – 0x358
- Read/Write Access: Write

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, and ClearTfr. Each Interrupt Clear register has a bit allocated per channel; for example, ClearTfr[2] is the clear bit for the Channel 2 transfer complete interrupt. Writing a 0 has no effect. These registers are not readable.

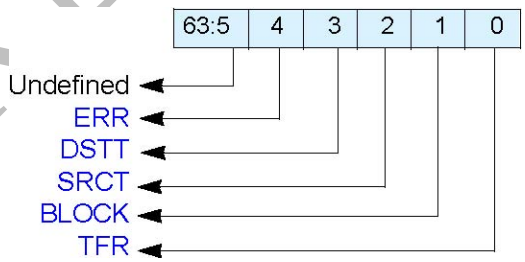


Bits	Name	R/W	Reset	Description
63:4	Undefined	N/A	0x0	Reserved
3:0	CLEAR	W	N/A	Interrupt clear. 0 = no effect 1 = clear interrupt

StatusInt

- Name: Combined Interrupt Status Register
- Size: 64 bits
- Address Offset: 0x360
- Read/Write Access: Read

The contents of each of the five Status registers – StatusTfr, StatusBlock, StatusSrcTran, StatusDstTran, StatusErr – is ORed to produce a single bit for each interrupt type in the Combined Status register(StatusInt). This register is read-only.



Bit	Name	R/W	Reset	Description
63:5	Undefined	N/A	0x0	Reserved
4	ERR	R	0x0	OR of the contents of StatusErr register.
3	DSTT	R	0x0	OR of the contents of StatusDst register.
2	SRCT	R	0x0	OR of the contents of StatusSrcTran register.
1	BLOCK	R	0x0	OR of the contents of StatusBlock register.
0	TFR	R	0x0	OR of the contents of StatusTfr register.

8.3.5 Software Handshaking Registers

The registers that comprise the software handshaking registers allow software to initiate single or burst transaction requests in the same way that handshaking interface signals do in hardware.

The software handshaking registers are:

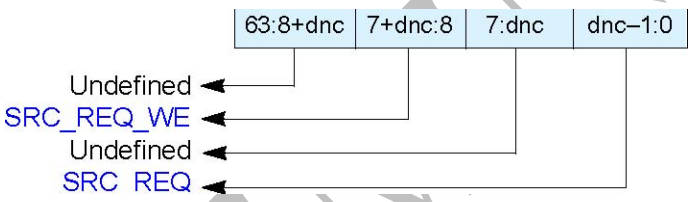
- LstDstReg – Last Destination Transaction Request Register
- LstSrcReg – Last Source Transaction Request Register
- ReqDstReg – Destination Software Transaction Request Register
- ReqSrcReg – Source Software Transaction Request Register
- SglReqDstReg – Single Destination Transaction Request Register
- SglReqSrcReg – Single Source Transaction Request Register

Setting CFGx.HS_SEL_SRC to 1 enables software handshaking on the source of channel x. Setting CFGx.HS_SEL_DST to 1 enables software handshaking on the destination of channel x.

ReqSrcReg

- Name: Source Software Transaction Request Register
- Size: 64 bits
- Address Offset: 0x368
- Read/Write Access: Read/Write

A bit is assigned for each channel in this register. ReqSrcReg[n] is ignored when software handshaking is not enabled for the source of channel n.



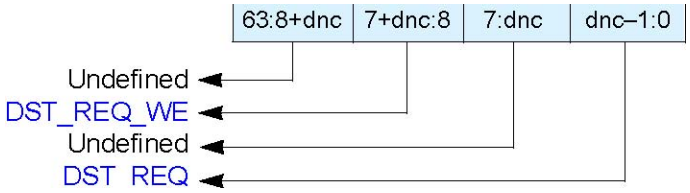
Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	SRC_REQ_WE	W	0x0	Source request write enable 0 = write disabled 1 = write enabled
7:4	Undefined	N/A	0x0	Reserved
3:0	SRC_REQ	R/W	0x0	Source request

A channel SRC_REQ bit is written only if the corresponding channel write enable bit in the SRC_REQ_WE field is asserted on the same AHB write transfer. For example, writing hex 0101 writes a 1 into ReqSrcReg[0], while ReqSrcReg[7:1] remains unchanged. Writing hex 00xx leaves ReqSrcReg[7:0] unchanged. This allows software to set a bit in the ReqSrcReg register without performing a read-modified write operation.

ReqDstReg

- Name: Destination Software Transaction Request Register
- Size: 64 bits
- Address Offset: 0x370
- Read/Write Access: Read/Write

A bit is assigned for each channel in this register. ReqDstReg[n] is ignored when software handshaking is not enabled for the source of channel n.



Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	DST_REQ_WE	W	0x0	Destination request write enable 0 = write disabled 1 = write enabled
7:4	Undefined	N/A	0x0	Reserved
3:0	DST_REQ	R/W	0x0	Destination request

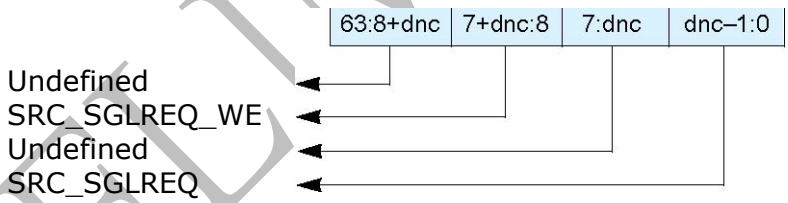
A channel DST_REQ bit is written only if the corresponding channel write enable bit in the DST_REQ_WE field is asserted on the same AHB write transfer.

The functionality of this register depends on whether the destination is a flow control peripheral or not. For a description of when the destination is not a flow controller.

SglReqSrcReg

- Name: Single Source Transaction Request Register
- Size: 64 bits
- Address Offset: 0x378
- Read/Write Access: Read/Write

A bit is assigned for each channel in this register. SglReqSrcReg[n] is ignored when softwarehandshaking is not enabled for the source of channel n.



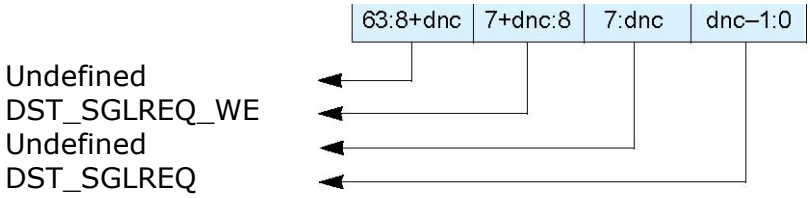
Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	SRC_SGLREQ_WE	W	0x0	Single write enable 0 = write disabled 1 = write enabled
7:4	Undefined	N/A	0x0	Reserved
3:0	SRC_SGLREQ	R/W	0x0	Source single request

A channel SRC_SGLREQ bit is written only if the corresponding channel write enable bit in the SRC_SGLREQ_WE field is asserted on the same AHB write transfer.
The functionality of this register depends on whether the source is a flow control peripheral or not.

SglReqDstReg

- Name: Single Destination Transaction Request Register
- Size: 64 bits
- Address Offset: 0x380
- Read/Write Access: Read/Write

A bit is assigned for each channel in this register. SglReqDstReg[n] is ignored when softwarehandshaking is not enabled for the destination of channel n.



Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	DST_SGLREQ_WE	W	0x0	Destination write enable 0 = write disabled 1 = write enabled
7:4	Undefined	N/A	0x0	Reserved
3:0	DST_SGLREQ	R/W	0x0	Destination single or burst request

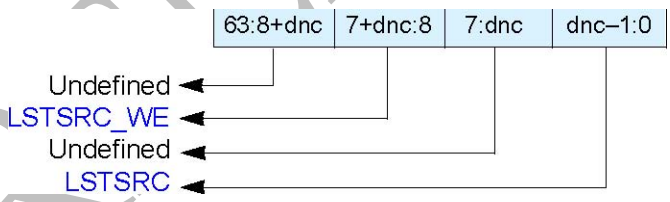
A channel DST_SGLREQ bit is written only if the corresponding channel write enable bit in the DST_SGLREQ_WE field is asserted on the same AHB write transfer. The functionality of this register depends on whether the destination is a flow control peripheral or not.

LstSrcReg

- Name: Last Source Transaction Request Register
- Size: 64 bits
- Address Offset: 0x388
- Read/Write Access: Read/Write

A bit is assigned for each channel in this register. LstSrcReg[n] is ignored when software handshaking is not enabled for the source of channel n, or when the source of channel n is not a flow controller.

A channel LSTSRC bit is written only if the corresponding channel write enable bit in the LSTSRC_WE field is asserted on the same AHB write transfer.



Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	LSTSRC_WE	W	0x0	Source last transaction request write enable 0 = write disabled 1 = write enabled
7:4	Undefined	N/A	0x0	Reserved
3:0	LSTSRC	R/W	0x0	Source last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

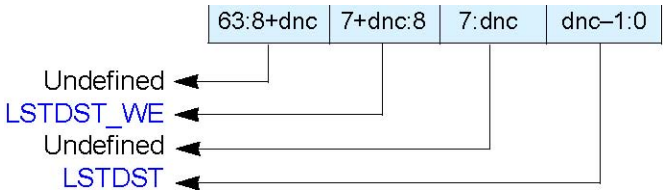
LstDstReg

- Name: Last Destination Transaction Request Register
- Size: 64 bits
- Address Offset: 0x390

- Read/Write Access: Read/Write

A bit is assigned for each channel in this register. LstDstReg[n] is ignored when software handshaking is not enabled for the destination of channel n or when the destination of channel n is not a flow controller.

A channel LSTDST bit is written only if the corresponding channel write enable bit in the LSTDST_WE field is asserted on the same AHB write transfer.



Bits	Name	R/W	Reset	Description
63:12	Undefined	N/A	0x0	Reserved
11:8	LSTDST_WE	W	0x0	Destination last transaction request write enable 0 = write disabled 1 = write enabled
7:4	Undefined	N/A	0x0	Reserved
3:0	LSTDST	R/W	0x0	Destination last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

8.4 Register Access

All registers are aligned to a 64-bit bdoundary and are 64 bits wide. In general, the upper 32 bits of a register are reserved. A write to reserved bits within the register is ignored. A read from reserved bits in the register reads back 0. To avoid address aliasing, do one of the following:

The DW_ahb_dmac should not be allocated more than 1 KB of address space in the system memory map. If it is, then addresses selected above 1 KB from the base address are aliased to an address within the 1 KB space, and a transfer takes place involving this register.

Software should not attempt to access non-register locations when hsel is asserted.

Note

The hsel signal is asserted by the system decoder when the address on the bus is within the system address assigned for DW_ahb_dmac.

8.5 Illegal Register Access

An illegal access can be any of the following:

1. A AHB transfer of hsize greater than 64 is attempted.
2. The hsel signal is asserted, but the address does not decode to a valid address.
3. A write to the SARx, DARx, LLPx, CTLx, SSTATx, DSTATx, SSTATARx, DSTATARx, SGRx, or DSRx registers occurs when the channel is enabled.
4. A read from the ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr is attempted.
5. A write to the StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr is attempted.
6. A write to the StatusInt register is attempted.
7. A write to either the DmaIdReg or DMA Component ID Register register is attempted.

The response to an illegal access is configured using the configuration parameter DMAH_RETURN_ERR_RESP. When DMAH_RETURN_ERR_RESP is set to True, an illegal

access (read/write) returns an error response.

If DMAH_RETURN_ERR_RESP is set to False, an OKAY response is returned, a read reads back 0x0, and a write is ignored.

8.6 DW_ahb_dmac Transfer Types

A DMA transfer may consist of single or multi-block transfers. On successive blocks of a multi-block transfer, the SARx/DARx register in the DW_ahb_dmac is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the CTLx register in the DW_ahb_dmac is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using Linked Lists is the multi-block method of choice. On successive blocks, the LLPx register in the DW_ahb_dmac is reprogrammed using block chaining with linked lists.

A block descriptor consists of six registers: SARx, DARx, LLPx, CTLx, SSTATx, and DSTATx. The first four registers, along with the CFG*x* register, are used by the DW_ahb_dmac to set up and describe the block transfer.

Note

The term Link List Item (LLI) and block descriptor are synonymous.

Multi-Block Transfers

Multi-block transfers are enabled by setting the DMAH_CHX_MULTI_BLK_EN configuration parameter to True.

Block Chaining Using Linked Lists

To enable multi-block transfers using block chaining, you must set the configuration parameter DMAH_CHx_MULTI_BLK_EN to True and the DMAH_CHx_HC_LLP parameter to False.

In this case, the DW_ahb_dmac reprograms the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

DW_ahb_dmac block chaining uses a Linked List Pointer register (LLPx) that stores the address in memory of the next linked list item. Each LLI contains the corresponding block descriptors:

1. SARx
2. DARx
3. LLPx
4. CTLx
5. SSTATx
6. DSTATx

To set up block chaining, you program a sequence of Linked Lists in memory.

The SARx, DARx, LLPx, and CTLx registers are fetched from system memory on an LLI update. If configuration parameter DMAH_CHx_CTL_WB_EN = True, then the updated contents of the CTLx, SSTATx, and DSTATx registers are written back to memory on block completion. Figure 2 and Figure 3 show how you use chained linked lists in memory to define multi-block transfers using block chaining.

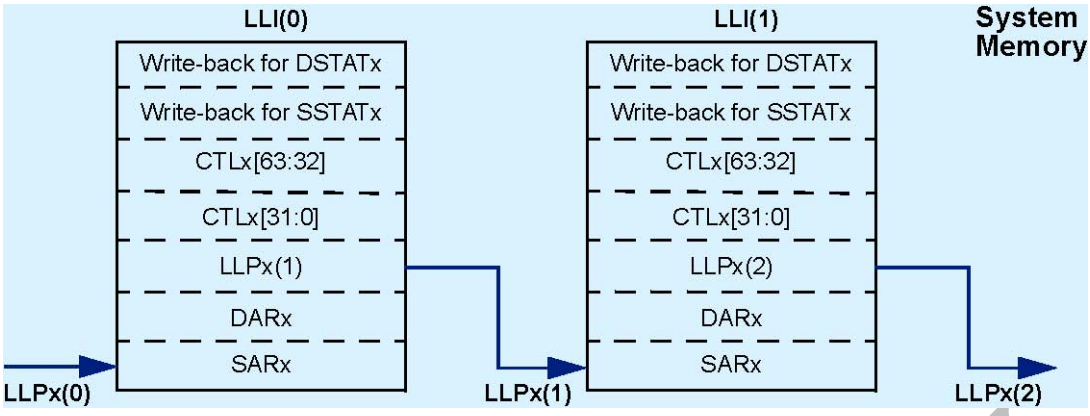


Figure 2: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to True

It is assumed that no allocation is made in system memory for the source status when the configuration parameter DMAH_CHx_STAT_SRC is set to False. If this parameter is False, then the order of a Linked List item is as follows:

1. SARx
2. DARx
3. LLPx
4. CTLx
5. DSTATx

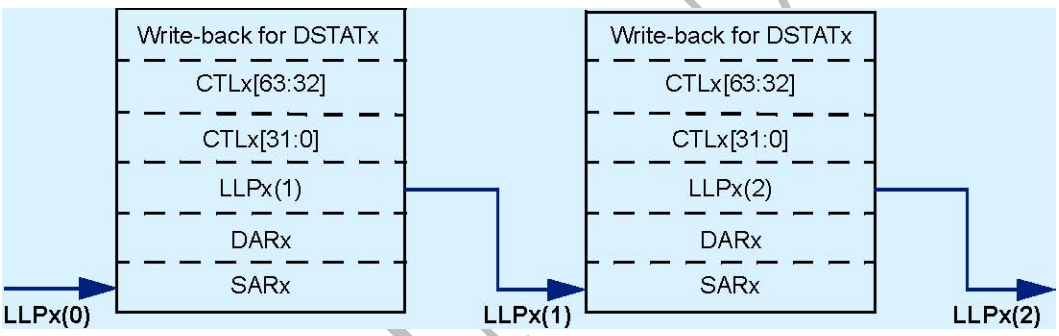


Figure 3: Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set to False

Note

In order to not confuse the SARx, DARx, LLPx, CTLx, STATx, and DSTATx register locations of the LLI with the corresponding DW_ahb_dmac memory mapped register locations, the LLI register locations are prefixed with LLI; that is, LLI.SARx, LLI.DARx, LLI.LLPx, LLI.CTLx, LLI.SSTATx, and LLI.DSTATx.

Note

For rows 6 through 10 of Table 5, the LLI.CTLx, LLI.LLPx, LLI.SARx, and LLI.DARx register locations of the LLI are always affected at the start of every block transfer. The LLI.LLPx and LLI.CTLx locations are always used to reprogram the DW_ahb_dmac LLPx and CTLx registers. However, depending on the Table 5 row number, the LLI.SARx/LLI.DARx address may or may not be used to reprogram the DW_ahb_dmac SARx/DARx registers.

Table 5: Programming of Transfer Types and Channel Register Update Method

Transfer Type	LLP.LOC =0	LLP_SRC_EN (CTLx)	RELOAD_SRC (CFGx)	LLP_DST_EN (CTLx)	RELOAD_DST (CFGx)	CTLx, LLPx Update Method	SARx Update Method	DARx Update Method	Write Backa
1. Single-block or last transfer of multi-block.	Yes	0	0	0	0	None, user reprograms	None (single)	None (single)	No
2. Auto-reload	Yes	0	0	0	1	CTLx, LLPx	Con-	Auto-	No

multi-block transfer with contiguous SAR						are reloaded from initial values.	tiguous	Reload	
3. Auto-reload multi-block transfer with contiguous DAR.	Yes	0	1	0	0	CTLx, LLPx are reloaded from initial values	Auto-Reload	Con-tiguous	No
4. Auto-reload multi-block transfer	Yes	0	1	0	1	CTLx, LLPx are reloaded from initial values	Auto-reload	Auto-Reload	No
5. Single-block or last transfer of multi-block.	No	0	0	0	0	None, user reprograms	None (single)	None (single)	Yes
6. Linked list multi-block transfer with contiguous SAR	No	0	0	1	0	CTLx, LLPx loaded from next Linked List item.	Con-tiguous	Linked List	Yes
7. Linked list multi-block transfer with auto-reload SAR	No	0	1	1	0	CTLx, LLPx loaded from next Linked List item.	Auto-Reload	Linked List	Yes
8. Linked list multi-block transfer with contiguous DAR	No	1	0	0	0	CTLx, LLPx loaded from next Linked List item.	Linked List	Con-tiguous	Yes
9. Linked list multi-block transfer with auto-reload DAR	No	1	0	0	1	CTLx, LLPx loaded from next Linked List item.	Linked List	Auto-Reload	Yes
10. Linked list multi-block transfer	No	1	0	1	0	CTLx, LLPx loaded from next Linked List item.	Linked List	Linked List	Yes

a. This column assumes that the configuration parameter DMAH_CHx_CTL_WB_EN = True. If DMAH_CHx_CTL_WB_EN = False, then there is never writeback of the control and status registers regardless of transfer type, and all rows of this column are "No".

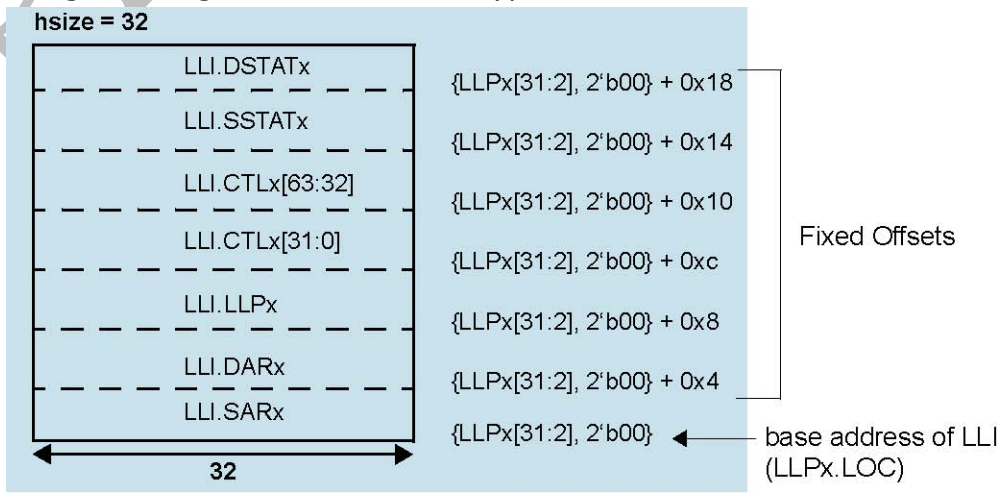


Figure 4: Mapping of Block Descriptor (LLI) in Memory to Channel Registers

Chapter 9 AHB-to-AHB Bridge

9.1 Design Overview

9.1.1 Overview

The AHB-to-AHB Bridge controller provides an interface to translate data between two AHB bus domains. It can support all kinds of burst type and data size, which define in AHB protocol. Besides bridge function, it supports DMA function also. It can support DMA transfer on one AHB bus devices or two AHB bus devices.

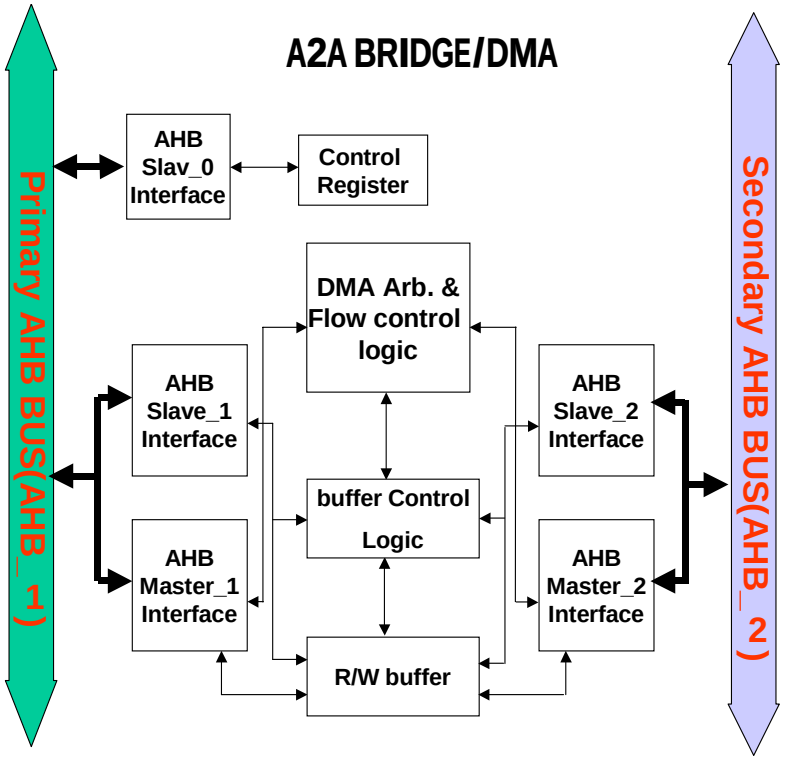
9.1.2 Features

- Provide AHB-to-AHB bus protocol translation
- Support AHB-to-AHB DMA or Single AHB DMA
- Two AHB master and two slave interfaces for bridge/DMA function
- Dedicate one AHB slave interface at primary bus domain for configuration
- Two DMA channels supported
- Support byte, half word and word data transfer sizes
- Support incremental and fixed addressing mode
- Support block and software DMA transfer mode
- Support all burst type transfer for bridge
- Support SINGLE and all incremental burst type transfer for DMA
- Support fixed channel priority arbitration
- Lock transfers are NOT permitted in bridge function
- Bus Error Recording

9.2 Architecture

This section provides a description about the functions and behavior under various conditions.

9.2.1 Block Diagram



9.2.2 Block Descriptions

AHB-to-AHB consist of one AHB slave interface(slave_0) for DMA function's register configuration and a pair of AHB master and slave interface(mastrer_1 and slave_1) for AHB_1 and a pair of AHB master and slave interface(mastrer_2 and slave_2) for AHB_2, witch can support AMBA protocol transfer from AHB_1 to AHB_2 or from AHB_2 to AHB_1.

9.3 Registers

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

9.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
A2A_CON0	0x0000	W	0x00000600	A2A channel 0 control register.
A2A_ISRC0	0x0004	W	0x20040421	A2A channel 0 initial source address register.
A2A_IDST0	0x0008	W	0x00000000	A2A channel 0 initial destination address register.
A2A_ICNT0	0x000C	W	0x00000000	A2A channel 0 initial terminates count register.
A2A_CSRC0	0x0010	W	0x20040421	A2A channel 0 current source address register.
A2A_CDST0	0x0014	W	0x00000000	A2A channel 0 current destination address register.
A2A_CCNT0	0x0018	W	0x00000000	A2A channel 0 current terminates count register.
A2A_CON1	0x001C	W	0x00000600	A2A channel 1 control register.
A2A_ISRC1	0x0020	W	0x00000000	A2A channel 1 initial source address register.

A2A_IDST1	0x0024	W	0x00000000	A2A channel 1 initial destination address register.
A2A_ICNT1	0x0028	W	0x00000000	A2A channel 1 initial terminates count register.
A2A_CSRC1	0x002C	W	0x00000000	A2A channel 1 current source address register.
A2A_CDST1	0x0030	W	0x00000000	A2A channel 1 current destination address register.
A2A_CCNT1	0x0034	W	0x00000000	A2A channel 1 current terminates count register.
A2A_INT_STS	0x0038	W	0x00000000	A2A Interrupt status register.
A2A_DMA_STS	0x003C	W	0x00000000	A2A channels status register.
A2A_ERR_ADR1	0x0040	W	0x00000000	Address of primary bus error.
A2A_ERR_OP1	0x0044	W	0x00000000	Operation of primary bus error.
A2A_ERR_ADR2	0x0048	W	0x00000000	Address of secondary bus error.
A2A_ERR_OP2	0x004C	W	0x00000000	Operation of secondary bus error.
A2A_LCNT0	0x0050	W	0x00000000	A2A channel 0 on the fly mode AHB bus Lock Count register.
A2A_LCNT1	0x0054	W	0x00000000	A2A channel 1 on the fly mode AHB bus Lock Count register.
A2A_DOMAIN	0x0058	W	0x00000000	A2A DMA Source and destination domain selection

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

9.3.2 Detail Register Description

A2A_CONx(x=0, 1)
External HDREQ source selection
Address: Operational Base + offset(0x0000, 0x001C)
DMA control register for channel x

Bit	Attr	Reset Value	Description
31:15	-	-	Reserved.
14	RW	0x0	Auto-Reload 0: Disable 1: Enable
13	RW	0x0	DMA HW enable/disable. This bit will be cleared to 0 when A2A finished DMA transfer in H/W mode. This bit has no affection in S/W mode (channel always enable in SW mode) 0: Disable HW DMA 1: Enable HW DMA
12	RW	0x0	Interrupt Mask. 0: Mask 1: Non-Mask
11	RW	0x0	On the fly mode. On the fly transfer can be applied on DMA which source and destination addresses are at the different bus domain. 0: Disable on the fly 1: Enable on the fly
10:9	RW	0x3	Transfer Mode.

			(INCR burst type will be used when remain data length less than 4,8,16 correspond to TM = 0x1, 0x2, 0x3) 0x0: Single 0x1: INCR4 0x2: INCR8 0x3: INCR16
8:7	RW	0x0	External HDREQ source selection. 0x0: From SD/MMC 0x1~0x3: Reserved.
6	RW	0x0	Direction of source address. 0: Increment 1: Fixed
5	RW	0x0	Direction of destination address. 0: Increment 1: Fixed
4:3	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
2:1	RW	0x0	Data size for transfer. 0x0: Byte 0x1: Halfword 0x2: Word 0x3: Reserved
0	RW	0x0	DMA mode. 0: Hardware block mode 1: Software mode

A2A_ISRCx(x=0, 1)

The DMA initial source address and initial destination address must have the same address alignment. It means that A2A_ISRCx[1:0] must equal to A2A_IDSTx[1:0].

Address: Operational Base + offset(0x0004, 0x0020)

DMA initial source address register for channel x

bit	Attr	Reset Value	Description
31:0	RW	0x0	A2A DMA initial source address register.

A2A_CSRCx(x=0, 1)

Address: Operational Base + offset(0x0010, 0x002C)

DMA current source address register for channel x

bit	Attr	Reset Value	Description
31:0	R	0x0	A2A DMA current source address register.

A2A_IDSTx(x=0, 1)

Address: Operational Base + offset(0x0008, 0x0024)

DMA initial destination address register for channel x

bit	Attr	Reset Value	Description
31:0	RW	0x0	A2A DMA initial destination address register.

A2A_CDSTx(x=0, 1)

Address: Operational Base + offset(0x0014, 0x0030)

DMA current destination address register for channel x

bit	Attr	Reset Value	Description
31:0	R	0x0	A2A DMA current destination address register.

A2A_ICNTx(x=0, 1)

Address: Operational Base + offset(0x000C, 0x0028)
DMA initial terminate count register for channel x

bit	Attr	Reset Value	Description
31:0	-	-	Reserved.
15:0	RW	0x0	A2A DMA initial terminate count register. It means data size (byte count) to be moved.

A2A_CCNTx(x=0, 1)
Address: Operational Base + offset(0x0018, 0x0034)
DMA current terminate count register for channel x

bit	Attr	Reset Value	Description
31:0	-	-	Reserved.
15:0	R	0x0	A2A DMA current terminate count register. It means the last data size (byte count) not yet is moved.

A2A_INT_STS
Address: Operational Base + offset(0x0038)
Interrupt status

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3	RW	0x0	Secondary Bus (AHB_2) Error Interrupt active, clear after write. 0: NO bus error 1: Bus error interrupt active
2	RW	0x0	Primary Bus (AHB_1) Error Interrupt active, clear after write. 0: NO bus error 1: Bus error interrupt active
1	RW	0x0	Channel 1 Interrupt active, clear interrupt after write. 0: not active 1: active
0	RW	0x0	Channel 0 Interrupt active, clear interrupt after write. 0: not active 1: active

A2A_DMA_STS
Address: Operational Base + offset(0x003C)
DMA status

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	R	0x0	Channel 1 DMA status 0: Channel 0 is free 1: Channel 0 is busy on DMA service
0	R	0x0	Channel 0 DMA status 0: Channel 0 is free 1: Channel 0 is busy on DMA service

A2A_ERR_ADRx(x=1,2)
Address: Operational Base + offset(0x0040, 0x0048)
Address when primary/secondary bus error

bit	Attr	Reset Value	Description
31:0	R	0x0	Address of primary/secondary bus error.

A2A_ERR_OPx(x=1,2)
Address: Operational Base + offset(0x0044, 0x004C)

Operation when primary/secondary bus error

bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0x0	Operation of primary/secondary bus error: 0: Bus error when read operation 1: Bus error when write operation

A2A_LCNTx(x=1,2)

Address: Operational Base + offset(0x0050, 0x0054)

Bus lock counts

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:0	RW	0x0	Bus lock counts at on-the-fly mode. 0x0: Never release bus until DMA process end 0x1: Release bus every 16-beats data transfer 0x2: Release bus every 32-beats data transfer 0x3: Release bus every 64-beats data transfer 0x4: Release bus every 128-beats data transfer 0x5: Release bus every 256-beats data transfer 0x6: Release bus every 512-beats data transfer 0x7: Release bus every 1024-beats data transfer

A2A_DOMAIN

Address: Operational Base + offset(0x0058)

A2A DMA Source/destination domain selection

Bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3	RW	0x0	DMA destination Domain selection for channel 1 0: Destination address is on AHB0 1: Destination address is on AHB1
2	RW	0x0	DMA Source Domain selection for channel 1 0: Source address is on AHB0 1: Source address is on AHB1
1	RW	0x0	DMA destination Domain selection for channel 0 0: Destination address is on AHB0 1: Destination address is on AHB1
0	RW	0x0	DMA Source Domain selection for channel 0 0: Source address is on AHB0 1: Source address is on AHB1

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

9.4 Functional Description

9.4.1 AHB Bridge Mode Operation

A2A provides transparent mode that is to transfer data between two AHB buses. When masters in AHB_1 (As in A2A block diagram) want to write data to slaves in AHB_2, A2A activated as an AHB slave in AHB_1 bus (slave_1). Slave_1 will capture the data and control signals from AHB_1 masters into write buffers in A2A module, and then transfer the write data from write buffer to AHB_2 slaves by master port (Master_2). When AHB_1 masters wants to read data from AHB_2 slaves, the A2A Slave_1 is designed to use retry response to AHB_1 masters until the read operation in AHB_2 is finished by Master_2 and read data is ready in read buffer, and then AHB_1 masters can complete the read transfer by Salve_1 normally. The Read/Write transfer from AHB_2 to AHB_1 is just like from

AHB_1 to AHB_2 we described above. The use of buffer and retry response in A2A module can avoid insetting wait state to drop off AHB bus utilization. The AHB transfer control signals such as HSIZE, HBURST or HPROT will be kept in across-domain transfers, but locked transfers are not permitted in bridge function to avoid deadlock condition.

9.4.2 AHB DMA Mode Operation

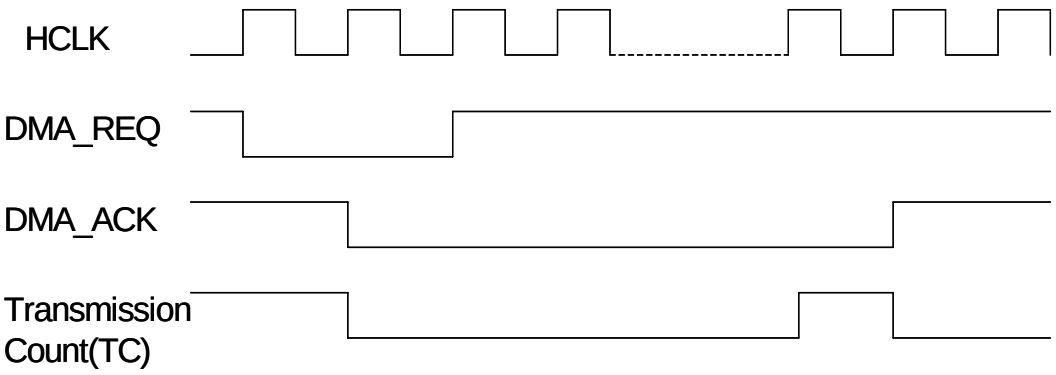
The DMA operation is to transfer data from slave devices to slave devices without interrupt CPU normal operation. There are block and software transfer modes in A2A module. Before A2A DMA start to operate in two different operation modes, we must program some information to A2A configuration register, for example, source address, destination address, terminate count (in A2A, terminate count means the data size) and other control signals… etc. The A2A will ready to receive request signal from other slave devices to start DMA operation after we provide some information to A2A registers. When DMA operate in block mode, it will be activated by H/W request, or by S/W request in software mode. There are two DMA channels in A2A DMA module and the channel priority arbitration is fixed, channel 0 always has the higher priority than channel 1. A2A DMA not only provides AHB_1-to-AHB_2 or AHB_2-to-AHB_1 but also AHB_1-to-AHB_1 and AHB_2-to-AHB_2 DMA transfers.

SW DMA Operation

After AHB reset signal (hreset_n), the A2A DMA module enters the initial state. In the initial state, the A2A DMA doesn't move data form slave devices to slave devices. When users finish programming configuration registers through AHB slave interface, the A2A enter transfer state to begin to transfer data from source address to destination address. When the TC (Terminate Count) is asserted, the A2A stops DMA operation and trigger an interrupt signal to interrupt controller.

HW DMA Operation

In HW block mode, A2A will assert acknowledge signal (DMA_ACK) when it receive request signal (DMA_REQ) from device. Device will dessert request signal if A2A assert acknowledge signal. When terminate count (TC) is zero, the A2A will de-assert acknowledge. The process of block mode DMA is just like software DMA we describe above. The only difference between block and software mode is that in software mode, DMA starts with software start bit in control registers, but in block mode DMA will start with hardware request signals. Figure below shows A2A's block mode operation from AHB_1 slave device to AHB_2 slave device. The flowing figure describes the HW DMA interface signals.



On-the-Fly DMA Mode

The A2A DMA supports On-the-Fly mode at both SW and HW mode. When On-the-Fly bit is set in DMA channel control register, A2A will request both primary and secondary AHB bus with lock signal. Once both buses are granted to A2A master ports, the A2A will start to read data from source address by one of the master ports and write data to

destination address by the other master port directly without using buffers. And due to buses are locked by A2A, A2A can transfer data without interrupt by other masters on buses until terminate count or specified data count are reached corresponds to A2ALCNTn register. Note that On-the-Fly transfer can only be applied at DMA process which source and destination addresses are at different bus segments, On-the-Fly bit will be ignored if source and destination addresses are at the same AHB bus domain.

9.5 Application Notes

- The A2A DMA only support the same size for source and destination peripheral.
- Not support byte and 16bits transfer in the condition of fixed address.

- The following is example step by step for software :

```
*(unsigned long volatile *)(A2AC_ARM_BASE + A2A_ISRC1_REG) = sar;    //source address
*(unsigned long volatile *)(A2AC_ARM_BASE + A2A_IDST1_REG) = dst;   //destination address
*(unsigned long volatile *)(A2AC_ARM_BASE + A2A_ICNT1_REG) = 256;   //transfer size
*(unsigned long volatile *)(A2AC_ARM_BASE + A2A_LCNT1_REG) = 0x01;
*(unsigned long volatile *)(A2AC_ARM_BASE + A2A_DOMAIN_REG) = 2<<2;
*(unsigned long volatile *)(A2AC_ARM_BASE + A2A_CON1_REG) =
((0x5)|((0x1)<<3)|(0x3<<9)|(0x1<<12));
```

Chapter 10 USB 2.0 Host Controller

10.1 Design Overview

10.1.1 Overview

The USB 2.0 Host Controller (UHC) includes both the Enhanced Host Controller Interface (EHCI) and Open Host Controller Interface (OHCI), and is compliant with the specifications of the EHCI and the OHCI. The UHC acts as a link between the system software and the USB 2.0/1.1 devices. The data transfer between the system software and the USB device is helped by the EHCI_OHCI function. It handles high-speed devices by direct transactions through the EHCI and non high-speed transactions through the OHCI. Transactions to a low/full speed device connected under a hub are done through split transactions. The EHCI or OHCI performs data transfer by reading the information queued by the software in the system memory in the form of data structures.

10.1.2 Features

- Compliant with USB 2.0 specification
- Compliant with Intel™ EHCI specification
- Compliant with OHCI specification
- Compliant with AMBA v2.0 AHB interface
- Supports UTMI+ specification level 3
- Supports High speed, full speed and low speed devices on root port and split transactions for full speed and low speed devices connected under a hub
- Supports Control, Bulk, Isochronous and Interrupt transfers
- Integrate Root Hub with Single port
- Fully static, synchronous, and synthesizable RTL design available in Verilog formats

For detail information about USB2.0 host controller, pleaser refer **RK27xx USB 2.0 Host Controller.pdf**

Chapter 11 USB 2.0 Device Controller

11.1 Design Overview

11.1.1 Overview

The USB 2.0 Device Controller (UDC 2.0) supports high-speed data transfer rates (up to 480Mbps), suspend / resume mode, control / bulk/ interrupt transfer and is fully compliant with the USB specification version 2.0. The AHB Bridge incorporates an AHB bus master interface, an AHB bus slave interface, buffer management logic and a DMA engine for all transfers.

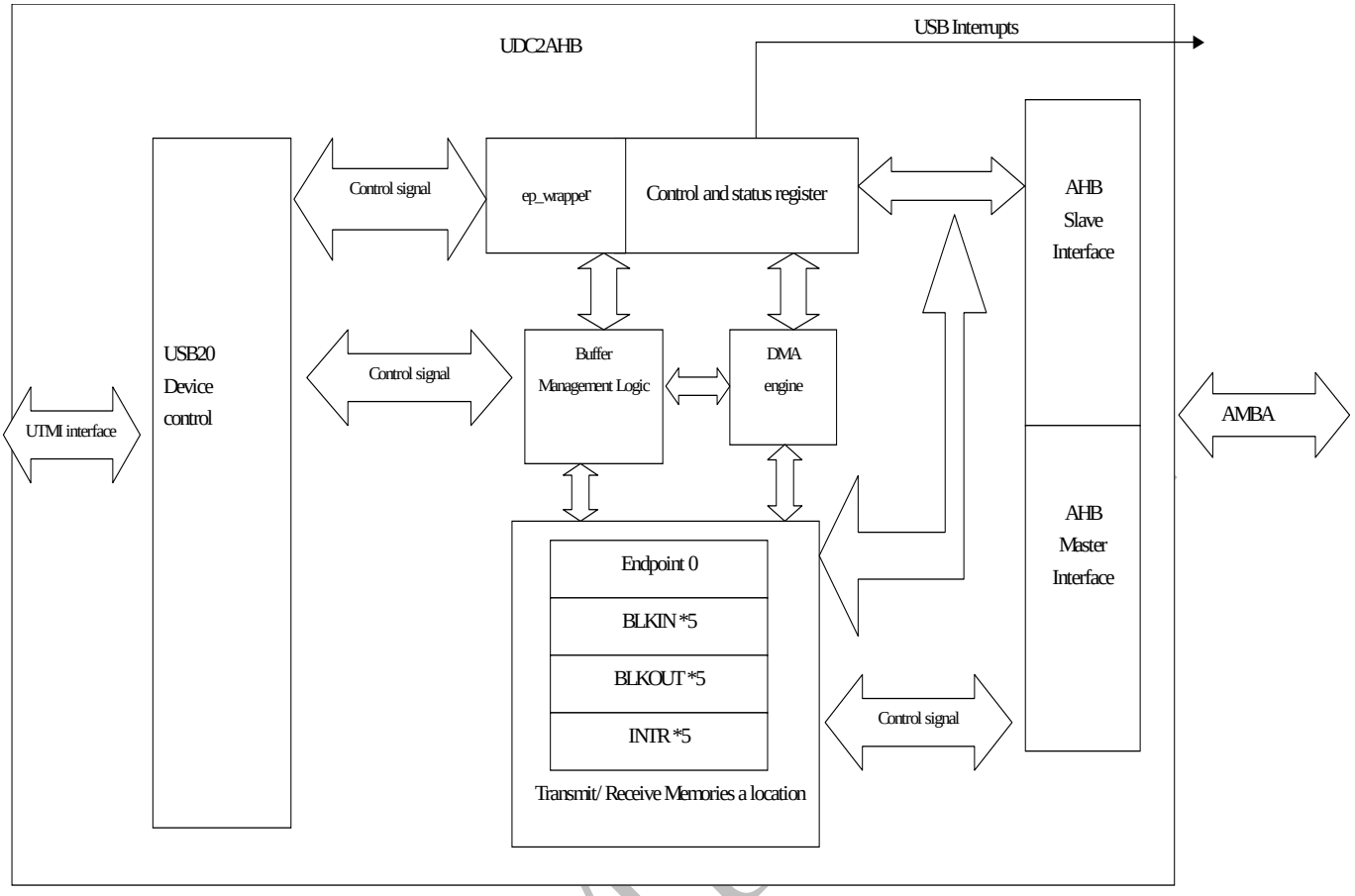
11.1.2 Features

- Complies with the Universal Serial Bus specification Rev. 2.0, Supports USB Full Speed (12Mb/sec) and High Speed (480 Mb/sec), is Backward compatible with USB1.1
- Support AHB M/S I/F operation
- Double buffering scheme for main endpoint increases throughput and eases real-time data transfer Dual-Port SDRAM: 2640 x 32-bit space
- Support Control, BULK and Interrupt transfer type
- 16 Endpoint; control ep x1, bulk-in ep x5, bulk-out ep x5, interrupt ep x5
- Automatic retry of failed packets, and PING Flow control
- Separate data buffers for the SETUP portion of a CONTROL transfer
- Support DMA Engine: to move the large block data between local memory and USB without intervene of CPU
- On-chip USB2.0 PHY and Parallel Bus Interface Engine (PIE)
- Suspend / Resume operation - Supports USB remote wake-up
- Automatic transmit/receive memory or buffer management
- One AHB bus master interface with a DMA engine for all transfers. The interface supports AHB RETRY and SPLIT operations
- One AHB bus slave interface for access of configuration registers, transmit and receive memories
- The DMA engine supports
  - Increment addressing mode
  - 32-bit wide transactions
  - DMA channels and fixed channel priority arbitration for all endpoints
  - 16-word burst transfer in DMA operation

11.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

11.2.1 Block Diagram



11.2.2 Block Descriptions

The architecture of UDC 2.0 includes AMBA interface, some masters and slaves of AMBA-interface device, and UDC 2.0 with AMBA interface and UTMI interface. The UDC 2.0 device can communicate with other master and slave device on AMBA through AMBA interface. All the devices on AMBA have an address, and UDC 2.0 can communication with any device on AMBA through an address.

In UDC 2.0 device, there are 16-endpoint, 5-BLKIN endpoint, 5-BLKOUT endpoint, 5-INTR endpoint and 1 endpoint 0. Every endpoint has its own DMA, interrupt, endpoint status register and so on. For choosing those functions (DMA, interrupt, and so on), it would be work through AMBA communicating with the AHB slave interface of UDC 2.0. The CSR (Control and Status Register) would receive the address from AHB slave interface and decodes the address. After decoding those address, the data would sent to related functions. Those devices on AMBA would know the UDC 2.0 has an event by USB interrupts and AHB master interface and they would be know whom be chosen and what to do by a address from AHB master interface.

The UDC 2.0 used UTMI to communicate with PHY and PHY used DP and DM to communicate with USB host.

11.3 Registers

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

11.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
DEV_CTL	0x0008	W	0x00000048	Device Control Register.
DEV_INFO	0x0010	W	0x00600000	Device Address Register.
EN_INT	0x0014	W	0x00000000	Interrupt Enable Register.
INT2FLAG	0x0018	W	0x00000080	Interrupt Flag Register.
INTCON	0x001C	W	0x00000005	Interrupt Control Register.
SETUP1	0x0020	W	0x00000000	Setup bmRequestType, RequestType, wValue Register.
SETUP2	0x0024	W	0x00000000	Setup wIndex, wLength Register.
AHBCON	0x0028	W	0x00000000	AHB Control Register
RX0STAT	0x0030	W	0x00000000	Endpoint 0 receive Status Register
RX0CON	0x0034	W	0x00000010	Endpoint 0 receive Control Register
DMA0CTLO	0x0038	W	0x00000001	DMA0 Control OUT Endpoint Register
DMA0LM_OADDR	0x003C	W	0x00000000	DMA0 Control OUT Endpoint local memory address Register
TX0STAT	0x0040	W	0x00000000	Endpoint 0 transmit Status Register
TX0CON	0x0044	W	0x00000008	Endpoint 0 transmit Control Register
TX0BUF	0x0048	W	0x00000000	Endpoint 0 Buffer Status Register
DMA0CTLI	0x004C	W	0x00000000	DMA0 Control IN Endpoint Register
DMA0LM_IADDR	0x0050	W	0x00000000	DMA0 Control IN Endpoint local memory address Register
RX1STAT	0x0054	W	0x00000000	Endpoint 1 receive Status Register
RX1CON	0x0058	W	0x00000010	Endpoint 1 receive Control Register
DMA1CTLO	0x005C	W	0x00000001	DMA1 Bulk OUT Endpoint local memory address Register
DMA1LM_OADDR	0x0060	W	0x00000000	DMA1 Bulk OUT Endpoint local memory address Register
TX2STAT	0x0064	W	0x00000000	Endpoint 2 transmit Status Register
TX2CON	0x0068	W	0x00000008	Endpoint 2 transmit Control Register
TX2BUF	0x006C	W	0x00000000	Endpoint 2 Buffer Status Register
DMA2CTLI	0x0070	W	0x00000000	DMA2 Bulk IN Endpoint Register
DMA2LM_IADDR	0x0074	W	0x00000000	DMA2 Bulk IN Endpoint local memory address Register
TX3STAT	0x0078	W	0x00000000	Endpoint 3 transmit Status Register

TX3CON	0x007C	W	0x00000008	Endpoint 3 transmit Control Register
TX3BUF	0x0080	W	0x00000000	Endpoint 3 Buffer Status Register
DMA3CTLI	0x0084	W	0x00000000	DMA3 Interrupt IN Endpoint Register
DMA3LM_IADDR	0x0088	W	0x00000000	DMA3 Interrupt IN Endpoint local memory address Register
RX4STAT	0x008C	W	0x00000000	Endpoint 4 receive Status Register
RX4CON	0x0090	W	0x00000010	Endpoint 4 receive Control Register
DMA4CTLO	0x0094	W	0x00000001	DMA4 Bulk OUT Endpoint Register
DMA4LM_OADDR	0x0098	W	0x00000000	DMA4 Bulk OUT Endpoint local memory address Register
TX5STAT	0x009C	W	0x00000000	Endpoint 5 transmit Status Register
TX5CON	0x00A0	W	0x00000008	Endpoint 5 transmit Control Register
TX5BUF	0x00A4	W	0x00000000	Endpoint 5 Buffer Status Register
DMA5CTLI	0x00A8	W	0x00000000	DMA5 Bulk IN Endpoint Register
DMA5LM_IADDR	0x00AC	W	0x00000000	DMA5 Bulk IN Endpoint local memory address Register
TX6STAT	0x00B0	W	0x00000000	Endpoint 6 transmit Status Register
TX6CON	0x00B4	W	0x00000008	Endpoint 6 transmit Control Register
TX6BUF	0x00B8	W	0x00000000	Endpoint 6 Buffer Status Register
DMA6CTLI	0x00BC	W	0x00000000	DMA6 Interrupt IN Endpoint Register
DMA6LM_IADDR	0x00C0	W	0x00000000	DMA6 Interrupt IN Endpoint local memory address Register
RX7STAT	0x00C4	W	0x00000000	Endpoint 7 receive Status Register
RX7CON	0x00C8	W	0x00000010	Endpoint 7 receive Control Register
DMA7CTLO	0x00CC	W	0x00000001	DMA7 Bulk OUT Endpoint Register
DMA7LM_OADDR	0x00D0	W	0x00000000	DMA7 Bulk OUT Endpoint local memory address Register
TX8STAT	0x00D4	W	0x00000000	Endpoint 8 transmit Status Register
TX8CON	0x00D8	W	0x00000008	Endpoint 8 transmit Control Register
TX8BUF	0x00DC	W	0x00000000	Endpoint 8 Buffer Status Register
DMA8CTLI	0x00E0	W	0x00000000	DMA8 Bulk IN Endpoint Register
DMA8LM_IADDR	0x00E4	W	0x00000000	DMA8 Bulk IN Endpoint local memory address Register
TX9STAT	0x00E8	W	0x00000000	Endpoint 9 transmit Status Register

TX9CON	0x00EC	W	0x00000008	Endpoint 9 transmit Control Register
TX9BUF	0x00F0	W	0x00000000	Endpoint 9 Buffer Status Register
DMA9CTLI	0x00F4	W	0x00000000	DMA9 Interrupt IN Endpoint Register
DMA9LM_IADDR	0x00F8	W	0x00000000	DMA9 Interrupt IN Endpoint local memory address Register
RX10STAT	0x00FC	W	0x00000000	Endpoint 10 receive Status Register
RX10CON	0x0100	W	0x00000010	Endpoint 10 receive Control Register
DMA10CTLO	0x0104	W	0x00000001	DMA10 Bulk OUT Endpoint Register
DMA10LM_OADDR	0x0108	W	0x00000000	DMA10 Bulk OUT Endpoint local memory address Register
TX11STAT	0x010C	W	0x00000000	Endpoint 11 transmit Status Register
TX11CON	0x0110	W	0x00000008	Endpoint 11 transmit Control Register
TX11BUF	0x0114	W	0x00000000	Endpoint 11 Buffer Status Register
DMA11CTLI	0x0118	W	0x00000000	DMA11 Bulk IN Endpoint Register
DMA11LM_IADDR	0x011C	W	0x00000000	DMA11 Bulk IN Endpoint local memory address Register
TX12STAT	0x0120	W	0x00000000	Endpoint 12 transmit Status Register
TX12CON	0x0124	W	0x00000008	Endpoint 12 transmit Control Register
TX12BUF	0x0128	W	0x00000000	Endpoint 12 Buffer Status Register
DMA12CTLI	0x012C	W	0x00000000	DMA12 Interrupt IN Endpoint Register
DMA12LM_IADDR	0x0130	W	0x00000000	DMA12 Interrupt IN Endpoint local memory address Register
RX13STAT	0x0134	W	0x00000000	Endpoint 13 receive Status Register
RX13CON	0x0138	W	0x00000010	Endpoint 13 receive Control Register
DMA13CTLO	0x013C	W	0x00000001	DMA13 Bulk OUT Endpoint Register
DMA13LM_OADDR	0x0140	W	0x00000000	DMA13 Bulk OUT Endpoint local memory address Register
TX14STAT	0x0144	W	0x00000000	Endpoint 14 transmit Status Register
TX14CON	0x0148	W	0x00000008	Endpoint 14 transmit Control Register
TX14BUF	0x014C	W	0x00000000	Endpoint 14 Buffer Status Register
DMA14CTLI	0x0150	W	0x00000000	DMA14 Bulk IN Endpoint Register
TX15STAT	0x0158	W	0x00000000	Endpoint 15 transmit Status Register
TX15CON	0x015C	W	0x00000008	Endpoint 15 transmit Control Register

TX15BUF	0x0160	W	0x00000000	Endpoint 15 Buffer Status Register
DMA15CTLI	0x0164	W	0x00000000	DMA15 Interrupt IN Endpoint Register
DMA15LM_IADDR	0x0168	W	0x00000000	DMA15 Interrupt IN Endpoint local memory address Register

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

11.3.2 Detail Register Description

PHY_TEST_EN
Address: Operational Base + offset (0x00)
PHY Test mode enable Register

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	RW	0x0	PHY_VATEST_EN: This test signal enables the input/output of analog test signals on the analog_test pin.
0	RW	0x0	PHY_TEST_CLK_EN: For sampling test data.

PHY_TEST
Address: Operational Base + offset (0x04)
PHY Test mode address and data input Register

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11:4	RW	0x0	PHY_TEST_DATA_IN[7:0]: For different test mode. It relies on PHY.
3:0	RW	0x0	PHY_TEST_ADDR[3:0]: For different test mode. It relies on PHY.

DEV_CTL
Address: Operational Base + offset (0x08)
Device Control Register

bit	Attr	Reset Value	Description
31:10	-	-	Reserved.
9	RW	0x0	TEST_MODE: TEST_MODE. This bit could reduce pattern effort during High-speed Chirp sequence. Hi-active. 0: Normal operation for High-speed Chirp sequence and vbus de-bounce time. 1: Test mode for High-speed Chirp sequence and vbus de-bounce time.
8	RW	0x0	CSR_DONE: Configure CSR done.
7	RW	0x0	SOFT_POR: Send power on reset to PHY.
6	RW	0x1	DEV_PHYBUS16_8: Enable Device data bus 8-bit mode. 0: 8-bit data path enabled. validh is undefined. udc_clk = 60MHz. 1: 16-bit data path enabled. udc_clk = 30MHz.
5	RW	0x0	DEV_RESUME: Enable Device Resume.

4	RW	0x0	DEV_SOFT_CN: Device soft disconnect.
3	RW	0x1	DEV_SELF_PWR: Device self power.
2	RW	0x0	DEV_RMTWKP: Device Remote Wakeup. Support device remote wakeup.
1:0	RW	0x0	DEV_SPEED: Device speed. 00:HS 11, 01,10 Reserved

DEV_INFO
Address: Operational Base + offset (0x10)
Device Address Register

bit	Attr	Reset Value	Description
31:23	-	-	Reserved.
22:21	R	0x3	dev_speed: Enumeration speed. 00:HS 11:FS 01,10 Reserved
20	R	0x0	vbus_sync: VBUS status. VBUS high means connection, VBUS low means disconnection.
19:16	R	0x0	dev_altintf: Alternate setting number.
15:12	R	0x0	intf_number: Interface number.
11:8	R	0x0	cfg_number: Configuration number.
7	R	0x0	DEV_EN: Device Enable. (set configuration ok).
6:0	R	0x0	dev_addr: Device Address.

EN_INT
Address: Operational Base + offset (0x14)
Enable INT Status Register

bit	Attr	Reset Value	Description
31:27	-	-	Reserved
26	RW	0x0	TEST_PKT: Enable TEST_PKT interrupt
25	RW	0x0	TEST_K: Enable TEST_K interrupt
24	RW	0x0	TEST_J: Enable TEST_J interrupt
23	RW	0x0	TEST_SE0_NAK: Enable TEST_SE0_NAK interrupt
22	RW	0x0	EN_IIN15_INTR: Enable INTR IN Token transmit Interrupt.
21	RW	0x0	EN_BIN14_INTR: Enable BULK IN Token transmit Interrupt.
20	RW	0x0	EN_BOUT13_INTR: Enable BULK OUT Token Receive Interrupt.
19	RW	0x0	EN_IIN12_INTR:

			Enable INTR IN Token transmit Interrupt.
18	RW	0x0	EN_BIN11_INTR: Enable BULK IN Token transmit Interrupt.
17	RW	0x0	EN_BOUT10_INTR: Enable BULK OUT Token Receive Interrupt.
16	RW	0x0	EN_IIN9_INTR: Enable INTR IN Token transmit Interrupt.
15	RW	0x0	EN_BIN8_INTR: Enable BULK IN Token transmit Interrupt.
14	RW	0x0	EN_BOUT7_INTR: Enable BULK OUT Token Receive Interrupt.
13	RW	0x0	EN_IIN6_INTR: Enable INTR IN Token transmit Interrupt.
12	RW	0x0	EN_BIN5_INTR: Enable BULK IN Token transmit Interrupt.
11	RW	0x0	EN_BOUT4_INTR: Enable BULK OUT Token Receive Interrupt.
10	RW	0x0	EN_IIN3_INTR: Enable INTR IN Token transmit Interrupt.
9	RW	0x0	EN_BIN2_INTR: Enable BULK IN Token transmit Interrupt.
8	RW	0x0	EN_BOUT1_INTR: Enable BULK OUT Token Receive Interrupt.
7	-	-	Reserved
6	RW	0x0	EN_SUSP_INTR: Enable Suspend Interrupt.
5	RW	0x0	EN_RSUME_INTR: Enable Resume Interrupt.
4	RW	0x0	EN_USBRST_INTR: Enable USB Reset Interrupt.
3	RW	0x0	EN_OUT0_INTR: Enable OUT Token Receive Interrupt. Endp0
2	RW	0x0	EN_IN0_INTR: Enable IN Token transmits Interrupt. Endp0
1	RW	0x0	EN_SETUP_INTR: Enable SETUP Packet Receive Interrupt.
0	RW	0x0	EN_SOF_INTR: Enable SOF Receive Interrupt.

INT2FLAG
Address: Operational Base + offset (0x18)
AP_INT2FLAG Status Register

bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26	R	0x0	TEST_PKT: Device went to TEST_PKT
25	R	0x0	TEST_K: Device went to TEST_K
24	R	0x0	TEST_J: Device went to TEST_J
23	R	0x0	TEST_SE0_NAK: Device went to TEST_SE0_NAK
22	R	0x0	IIN15_INTR: INTRIN, IN Token Endpoint Transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint.

			Read clear
21	R	0x0	BIN14_INTR: BLKIN, IN Token Endpoint transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
20	R	0x0	BOUT13_INTR: BLKOUT, OUT Token Endpoint Receive Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
19	R	0x0	IIN12_INTR: INTRIN, IN Token Endpoint Transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
18	R	0x0	BIN11_INTR: BLKIN, IN Token Endpoint transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
17	R	0x0	BOUT10_INTR: BLKOUT, OUT Token Endpoint Receive Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
16	R	0x0	IIN9_INTR: INTRIN, IN Token Endpoint Transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
15	R	0x0	BIN8_INTR: BLKIN, IN Token Endpoint transmits Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
14	R	0x0	BOUT7_INTR: BLKOUT, OUT Token Endpoint Receive Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
13	R	0x0	IIN6_INTR: INTRIN, IN Token Endpoint Transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
12	R	0x0	BIN5_INTR: BLKIN, IN Token Endpoint transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
11	R	0x0	BOUT4_INTR: BLKOUT, OUT Token Endpoint 4 Receive Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
10	R	0x0	IIN3_INTR:

			INTRIN, IN Token Endpoint Transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
9	R	0x0	BIN2_INTR: BLKIN, IN Token Endpoint transmits Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
8	R	0x0	BOUT1_INTR: BLKOUT, OUT Token Endpoint Receive Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. Read clear
7	R	0x1	VBUS_INTR: USB VBUS Interrupt. This bit indicates a connection in usb port. Read clear
6	R	0x0	SUSP_INTR: Suspend Interrupt. This bit is set when the UDC 2.0 detects a USB Suspend request from the host. The Suspend Request state cannot be set or cleared by writing this bit.
5	R	0x0	RSUME_INTR: Resume Interrupt. When set, this bit indicates that a device resume has occurred. Read clear
4	R	0x0	USBRST_INTR: USB Reset Interrupt. This bit indicates a change in state of the root port reset detector. Read clear
3	R	0x0	OUT0_INTR: OUT Token Endpoint 0 Receive Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. So, we got to read AP_RX0STAT register to decode what's next step. Read clear
2	R	0x0	IN0_INTR: IN Token Endpoint 0 transmit Interrupt. When set, this bit enables a local interrupt to be set when an interrupt is active on this endpoint. So, we got to read TX0STAT register to decode what's next step. Read clear
1	R	0x0	SETUP_INTR: SETUP Packet Interrupt. When set, this bit enables a local interrupt to be generated when a setup packet has been received from the host. Read clear
0	R	0x0	SOF_INTR: SOF Interrupt. When set, this bit enables a local interrupt to be generated when the start-of-frame packet is received.

INTCON

Address: Operational Base + offset (0x1C)

Interrupt control Register

bit	Attr	Reset Value	Description

31:3	-	-	Reserved.
2	RW	0x1	INT0MODE: 0: Low active, 1: High active.
1	RW	0x0	INT0TYPE: 0: Level Trigger, 1: Edge Trigger.
0	RW	0x1	INT0EN: Writing one to this bit to enable USB Interrupt0.

SETUP1

Address: Operational Base +offset (0x20)

Standard Setup bmRequestType, bRequest, wValue Register

bit	Attr	Reset Value	Description
31:16	R	0x0	WValue: Standard WValue setup packet received.
15:8	R	0x0	BRequest: Standard request code Value Request 0: Get status 1: Clear feature 2: Reserved 3: Set feature 4: Reserved 5: Set Address 6: Get Descriptor 7: Set Descriptor 8: Get configuration 9: Set configuration a: Get interface b: Set interface c: Sync frame
7	R	0x0	BmRequestType: Direction 0: host to device. 1: device to host.
6:5	R	0x0	BmRequestType: Type 0: Standard 1: Class 2: Vendor 3: Reserved.
4:0	R	0x0	BmRequestType: Recipient 0: Device 1: Interface 2: Endpoint 3: Other 4 – 31: Reserved.

SETUP2

Address: Operational Base +offset (0x24)

Standard Setup wIndex, wLength Register

bit	Attr	Reset Value	Description
31:16	R	0x0	WLength: Standard wLength setup packet received.
15:0	R	0x0	WIndex: Standard wIndex setup packet received.

AHBCON
Address: Operational Base + offset (0x28)
AHB Control Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	MID: AHB Master ID.

RX0STAT
Address: Operational Base + offset (0x30)
Endpoint 0 Receive status Register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25	R	0x0	RX0OVF: Set to one when endpoint 0 of the receive memory were over-flow.
24	R	0x0	RX0FULL: [31:24] RX0 BUFFER STATUS Set by UDC 2.0 to indicate at least one packet available in the endpoint1 When receive ACK from ep_wrapper, can be cleared by setting the RX0FFRC bit of RX0CON register or automatically cleared by DMA controller when the DMA operation has been done When receive DMA0_ODONE (DMA transfer done)
23:19	-	-	Reserved.
18	R	0x0	RX0ACK: Set to 1 to indicate the receive transaction was successful. DMA transfer done. Read clear
17	R	0x0	RX0ERR: Receive Error when set to indicate: Due to an error such as over-flow (RX0OVF) or UDC 2.0 Fetal error in the endpoint 0 Read clear
16	R	0x0	RX0VOID: [23:16] RX0 ENDP STATUS Set to 1 when no valid data received due to RX0STALL and RX0NAK set to 1 in RX0CON Register. Read clear
15:11	-	-	Reserved.
10:0	R	0x0	RX0LEN: [15:0] RX0 LENGTH Byte length of data in the data buffer to be received.

RX0CON
Address: Operational Base + offset (0x34)
Endpoint 0 Receive Control Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7	RW	0x0	RX0ACKINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of RX0ACK.
6	RW	0x0	RX0ERRINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of RX0ERR.
5	RW	0x0	RX0VOIDINTEN:

			Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of RX0VOID.
4	RW	0x1	EP0EN: Endpoint 0 Enable. When cleared the endpoint does not respond to an SETUP or OUT token.
3	RW	0x0	RX0NAK: Set as one to response NAK handshake about this endpoint. Set by CPU
2	RW	0x0	RX0STALL: Set as one to response STALL handshake about this endpoint. Set by CPU, Auto Clear by UDC 2.0.
1	RW	0x0	RX0CLR: Set the bit to flush the RX0 FIFO. Set by CPU.
0	RW	0x0	RX0FFRC: In non-DMA mode, user needs to set this bit to relieve the RX0 receive memory and clear RX0FULL of RX0BUFFLG. (In DMA mode, the RX0FULL flag is automatically cleared by DMA controller when the DMA operation has been done)

DMA0CTLO

Address: Operational Base + offset (0x38)

DMA0 Control OUT Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x1	DMA0OUTSTA: Set as ONE to start the DMA operation. For the data movement from the endpoint 0 receive buffer to local memory, and cleared by the DMA controller whenever operation is done.

DMA0OUTLMADDR

Address: Operational Base + offset (0x3C)

DMA0 Control OUT Local memory Register

bit	Attr	Reset Value	Description
31:2	RW	0x0	LM0OUTADDR: Set to indicate the starting address of the local memory block.
1:0	-	-	Reserved

TX0STAT

Address: Operational Base + offset (0x40)

Endpoint 0 Transmit Status Register

bit	Attr	Reset Value	Description
31:19	-	-	Reserved.
18	R	0x0	TX0ACK: Transmit Successful. Receive ACK handshake from host. Read clear
17	R	0x0	TX0ERR: Transmit Error when set to indicate: Due to an error such as under-run (TX0URF) or UDC 2.0 Fetal error in the endpoint 0. Read clear
16	R	0x0	TX0VOID: [23:16] TX0 Status

			The Transmit Void Condition which is associated with NAK/STALL (TX0CON Register) handshake returned by the function after a valid IN token, due to the conditions that cause the data register to be un-enabled or not ready to transmit. Read clear
15:11	-	-	Reserved.
10:0	RW	0x0	TX0LEN: [15:0] TX0 Length Byte length of data in the data buffer to be transmitted.

TX0CON
Address: Operational Base + offset (0x44)
Endpoint 0 Transmit Control Register

bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0x0	TX0ACKINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TX0ACK.
5	RW	0x0	TX0ERRINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TX0ERR.
4	RW	0x0	TX0VOIDINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TX0VOID.
3	-	-	Reserved.
2	RW	0x0	TX0NAK: Set as one to response NAK handshake about this endpoint. Set by CPU
1	RW	0x0	TX0STALL: Set as one to response STALL handshake about this endpoint. Set by CPU, Auto Clear by UDC 2.0.
0	RW	0x0	TX0CLR: Set this bit to flush the endpoint 0 transmit memory. Set by CPU (RW).

TX0BUF
Address: Operational Base + offset (0x48)
Endpoint 0 Buffer Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	R	0x0	TX0URF: Set to one when the transmit memory were under-run.
0	R	0x0	TX0FULL: Transmit Buffer status Indicate one packet has been written and available to be transmitted in the data registers, will be cleared by the UDC 2.0 for a successful IN transaction when receive ACK from UDC 2.0.

DMA0INCTL
Address: Operational Base + offset (0x4C)
DMA0 Control IN Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved

0	RW	0x0	DMA0INSTA: Set as ONE to start the DMA operation. For the data movement from the local memory to the endpoint 0 transmit buffer, and cleared by the DMA controller whenever operation is done.

DMA0LM_IADDR

Address: Operational Base + offset (0x50)

DMA0 Control IN Local memory Register

bit	Attr	Reset Value	Description
31:2	RW	0x0	LM0INADDR: Set to indicate the starting address of the local memory block.
1:0	-	-	Reserved

(RXSTAT) For RX1, 4, 7, 10, 13

Address: Operational Base + offset (0x54, 0x8C, 0xC4, 0xfc, 0x134)

Receiver status Register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved.
25	R	0x0	RXOVF: Set to one when endpoint of the receive memory were over-flow.
24	R	0x0	RXFULL: Set by UDC 2.0 to indicate at least one packet available in the endpoint When receive ACK from ep_wrapper, can be cleared by setting the RXFFRC bit of RXCON register or automatically cleared by DMA controller when the DMA operation has been done When receive DMA DONE (DMA transfer done)
23:20	-	-	Reserved.
19	R	0x0	RX_CF_INT: Set to 1 to indicate clear feature sent by UHC. Read clear
18	R	0x0	RXACK: Set to 1 to indicate the receive transaction was successful. DMA transfer done. Read clear
17	R	0x0	RXERR: Receive Error when set to indicate: Due to an error such as over-flow (RXOVF) or UDC 2.0 Fetal error in the endpoint. Read clear
16	R	0x0	RXVOID: Set to 1 when no valid data received due to RXSTALL and RXNAK set to 1 in RXCON Register. Read clear
15:11	-	-	Reserved.
10:0	R	0x0	RXCNT: Byte length of data for the endpoint 1. For receive memory from USB Host side.

(RXCON) for RX1, 4, 7, 10, 13

Address: Operational Base + offset (0x58, 0x90, 0xC8, 0x100, 0x138)

Receiver control Register

bit	Attr	Reset Value	Description
31:14	-	-	Reserved.

13	RW	0x0	RXSTALL_AUTOCLR: Here are the first priority compare with function stall. Set as one to response STALL handshake for current endpoint. Set by CPU, Auto Clear by UDC 2.0.
12	RW	0x0	RX_CF_INTE: Set as one to enable the INTERRUPT OUTPUT for RX_CF_INT (Clear Feature).
11:8	RW	0x0	RXENDP_NUM: Endpoint number setting.
7	RW	0x0	RXACKINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of RXACK.
6	RW	0x0	RXERRINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of RXERR.
5	RW	0x0	RXVOIDINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of RXVOID.
4	RW	0x1	EPEN: default enable. Endpoint Enable. When cleared the endpoint does not respond to an SETUP or OUT token. (USBrst, PowerOnrst)
3	RW	0x0	RXNAK: Set as one to response NAK handshake about this endpoint. Set by CPU.
2	RW	0x0	RXSTALL: Set as one to response STALL handshake about this endpoint. Set by CPU.
1	RW	0x0	RXCLR: Set the bit to flush the Receive FIFO. Set by CPU (RW).
0	RW	0x0	RXFFRC: In non-DMA mode, CPU needs to set this bit to relieve the RX receive memory and clear RXFULL of RXBUFFLG. (In DMA mode, the RXFULL flag is automatically cleared by DMA controller when the DMA operation has been done)

(DMACTLO) For RX1, 4, 7, 10, 13

Address: Operational Base + offset (0x5C, 0x94, 0xCC, 0x104, 0x13C)

DMA Control Register

bit	Attr	Reset Value	Description
31:1	RW	0x1	DMAOUTSTA: Set as ONE to start the DMA operation. For the data movement from the endpoint receive buffer to local memory, and cleared by the DMA controller whenever operation is done.
0	-	-	Reserved

(DMAOUTLMADDR) For RX1, 4, 7, 10, 13

Address: Operational Base + offset (0x60, 0x98, 0xD0, 0x108, 0x140)

DMA Local memory setting Register

bit	Attr	Reset Value	Description
31:2	RW	0x0	LMOUTADDR: Set to indicate the starting address of the local memory block.

1:0	-	-	Reserved

(TXSTAT) For TX2, 5, 8, 11, 14
Address: Operational Base + offset (0x64, 0x9C, 0xD4, 0x010C, 0x0144)
Transmitter status Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	R	0x0	TX_CF_INT: Set to 1 to indicate clear feature sent by UHC. Read clear
19	R	0x0	TXDMA_DN: Transmit Successful. Receive DMA DONE handshake from host. Read clear
18	R	0x0	TXACK: Transmit Successful. Receive ACK handshake from host. Read clear
17	R	0x0	TXERR: Transmit Error when set to indicate: Due to an error such as under-run (TXURF) or UDC 2.0 Fetal error in the endpoint. Read clear
16	R	0x0	TXVOID: The Transmit Void Condition which is associated with NAK/STALL (TXCON Register) handshake returned by the function after a valid IN token, due to the conditions that cause the data register to be un-enabled or not ready to transmit. Read clear
15:11	-	-	Reserved.
10:0	RW	0x0	TXLEN: Byte length of data in the data buffer to be transmitted.

(TXCON) For TX2, 5, 8, 11, 14
Address: Operational Base + offset (0x68, 0xA0, 0xD8, 0x110, 0x148)
Transmitter control Register

bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13	RW	0x0	TXSTALL_AUTOCLR: Here are the first priority compare with function stall. Set as one to response STALL handshake for current endpoint. Set by CPU, Auto Clear by UDC.
12	RW	0x0	TX_CF_INTE: Set as one to enable the INTERRUPT OUTPUT for TX_CF_INT (Clear Feature).
11:8	RW	0x0	TXENDP_NUM: Endpoint number setting.
7	RW	0x0	TXDMADN_EN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of DMA DONE.
6	RW	0x0	TXACKINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TXACK.
5	RW	0x0	TXERRINTEN: Set as one to enable the INTERRUPT OUTPUT

			generation initiated by the assertion of TXERR.
4	RW	0x0	TXVOIDINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TXVOID.
3	RW	0x1	TXEPEN: Transmit Endpoint Enable. The endpoint does not respond when cleared.
2	RW	0x0	TXNAK: Set as one to response NAK handshake about this endpoint. Set by CPU
1	RW	0x0	TXSTALL: Set as one to response STALL handshake about this endpoint. Set by CPU.
0	RW	0x0	TXCLR: Set this bit to flush the endpoint transmit memory.

(TXBUF) For TX2, 5, 8, 11, 14
Address: Operational Base + offset (0x6C, 0xA4, 0xDC, 0x114, 0x14C)
Transmitter buffer Status Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3	R	0x0	TXDS1: Indicate Data set1 is full, high active.
2	R	0x0	TXDS0: Indicate Data set0 is full, high active.
1	R	0x0	TXURF: Set to one when the transmit memory were under-run.
0	R	0x0	TXFULL: Transmit Buffer status Indicate one packet has been written and available to be transmitted in the transmit data registers, will be cleared by the UDC for a successful IN transaction when receive ACK from UDC.

(DMAINCTL) For TX2, 5, 8, 11, 14
Address: Operational Base + offset (0x70, 0xA4, 0xDC, 0x118, 0x150)
DMA Control Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	DMAINSTA: Set as ONE to start the DMA operation. For the data movement from the local memory to the endpoint transmit buffer, and cleared by the DMA controller whenever operation is done.

(DMALM_IADDR) For TX2, 5, 8, 11, 14
Address: Operational Base + offset (0x74, 0xAC, 0xE4, 0x11C, 0x154)
DMA Local memory setting Register

bit	Attr	Reset Value	Description
31:2	RW	0x0	LMINADDR: Set to indicate the starting address of the local memory block.
1:0	-	-	Reserved

(TXSTAT) For TX3, 6, 9, 12, 15
Address: Operational Base + offset (0x78, 0xB0, 0xE8, 0x120, 0x158)

Transmitter Status Register

bit	Attr	Reset Value	Description
31:20	-	-	Reserved.
19	R	0x0	TX_CF_INT: Set to 1 to indicate clear feature sent by UHC. Read clear
18	R	0x0	TXACK: Transmit Successful. Receive ACK handshake from host. Read clear
17	R	0x0	TXERR: Transmit Error when set to indicate: Due to an error such as under-run (TXURF) or UDC Fetal error in the endpoint. Read clear
16	R	0x0	TXVOID: The Transmit Void Condition which is associated with NAK/STALL (TXCON Register) handshake returned by the function after a valid IN token, due to the conditions that cause the data register to be un-enabled or not ready to transmit. Read clear
15:11	-	-	Reserved.
10:0	RW	0x0	TXLEN: Byte length of data in the data buffer to be transmitted.

(TXCON) For TX3, 6, 9, 12, 15

Address: Operational Base + offset (0x7C, 0xB4, 0xEC, 0x124, 0x158)

Transmitter control Register

bit	Attr	Reset Value	Description
31:14	-	-	Reserved.
13	RW	0x0	TXSTALL_AUTOCLR: Here are the first priority compare with function stall. Set as one to response STALL handshake for current endpoint. Set by CPU, Auto Clear by UDC.
12	RW	0x0	TX_CF_INTE: Set as one to enable the INTERRUPT OUTPUT for TX_CF_INT (Clear Feature).
11:8	RW	0x0	TXENDP_NUM: Endpoint number setting.
7	-	-	Reserved.
6	RW	0x0	TXACKINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TXACK.
5	RW	0x0	TXERRINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TXERR.
4	RW	0x0	TXVOIDINTEN: Set as one to enable the INTERRUPT OUTPUT generation initiated by the assertion of TXVOID.
3	RW	0x1	TXEPEN: Transmit Endpoint Enable. The endpoint does not respond when cleared.
2	RW	0x0	TXNAK: Set as one to response NAK handshake about this endpoint. Set by CPU

1	RW	0x0	TXSTALL: Set as one to response STALL handshake about this endpoint. Set by CPU.
0	RW	0x0	TXCLR: Set this bit to flush the endpoint transmit memory. Set by CPU (RW).

(TXBUF) For TX3, 6, 9, 12, 15
Address: Operational Base + offset (0x80, 0xB8, 0xF0, 0x128, 0x160)
Transmitter Buffer Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1	R	0x0	TXURF: Set to one when the transmit memory were under-run.
0	R	0x0	TXFULL: Transmit Buffer status Indicate one packet has been written and available to be transmitted in the data registers, will be cleared by the UDC for a successful IN transaction when receive ACK from UDC.

(DMAINCTL) For TX3, 6, 9, 12, 15
Address: Operational Base + offset (0x84, 0xBC, 0xF4, 0x12C, 0x164)
DMA Control Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	DMAINSTA: Set as ONE to start the DMA operation. For the data movement from the local memory to the endpoint transmit buffer, and cleared by the DMA controller whenever operation is done.

(DMALM_IADDR) For TX3, 6, 9, 12, 15
Address: Operational Base + offset (0x88, 0xC0, 0xF8, 0x130, 0x168)
DMA Local memory setting Register

bit	Attr	Reset Value	Description
31:2	RW	0x0	LMINADDR: Set to indicate the starting address of the local memory block.
1:0	-	-	Reserved

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

11.4 Functional Description

11.4.1 Operation

This document is intended to be a programming guide of UDC 2.0. The UDC 2.0 is a USB 2.0 device with AHB bus interface, and can be a DMA master with its efficient buffer management.
The functional description is composed of several sections:

- Initialize the H/W
- Control Write Steps
- Control Read Steps
- Bulk Out Steps
- Bulk In Steps

From the point view of a driver, these operations cover all that a driver should provide. More detail programming sequences are listed in the following sections.

DIAGRAMS

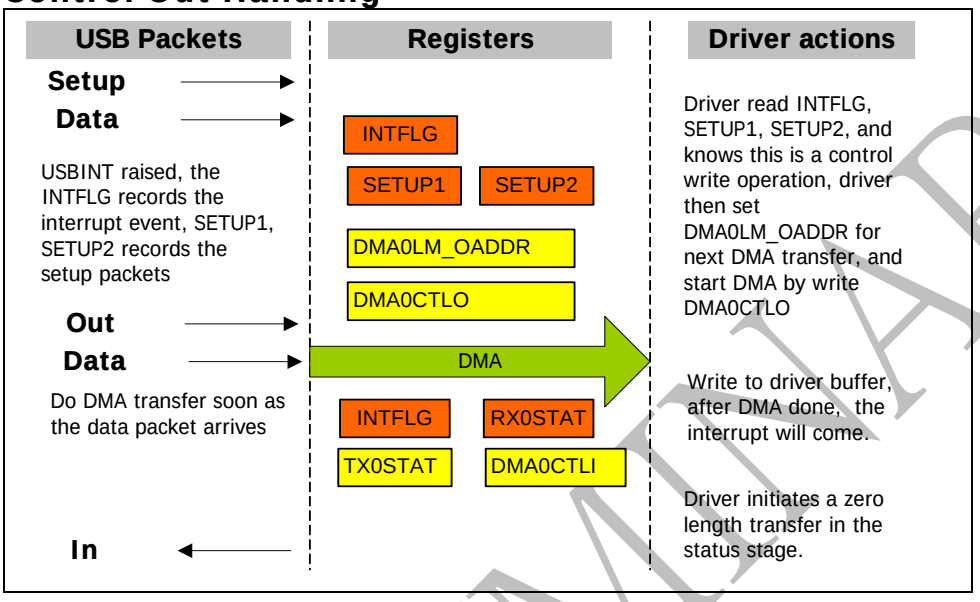
Register need to read

Register need to write

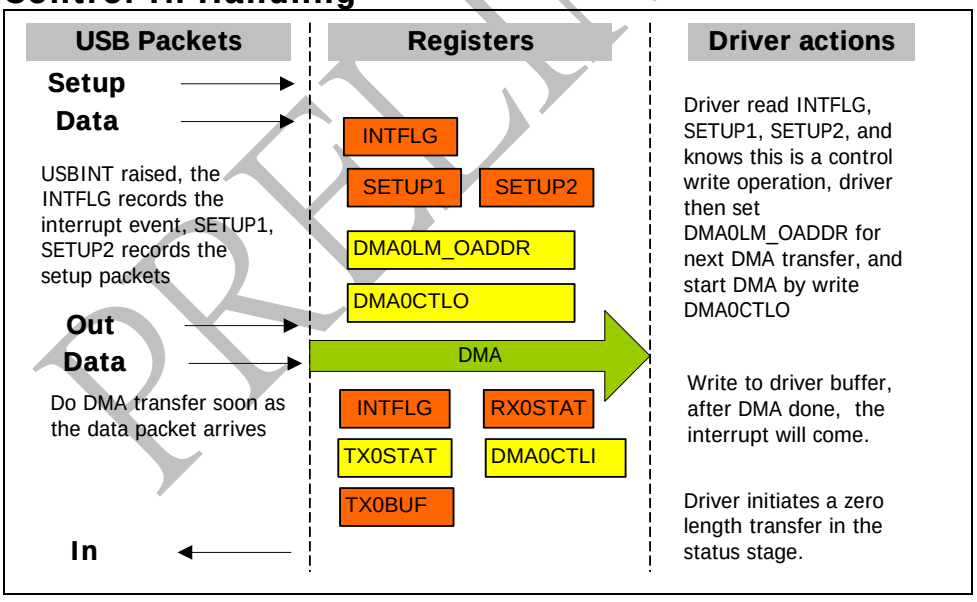
DMA data

Note: USB ACK is omitted for simplicity

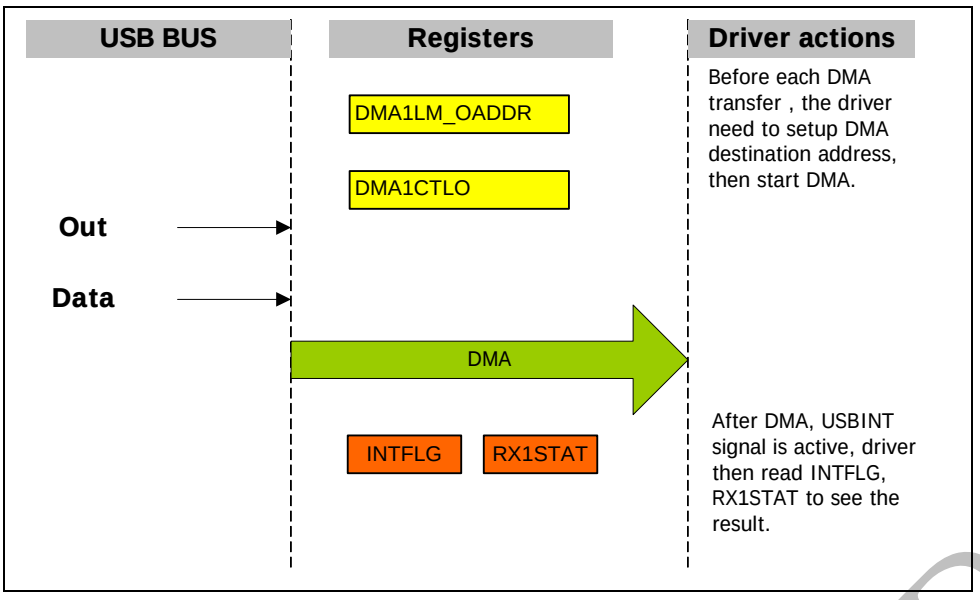
Control Out Handling



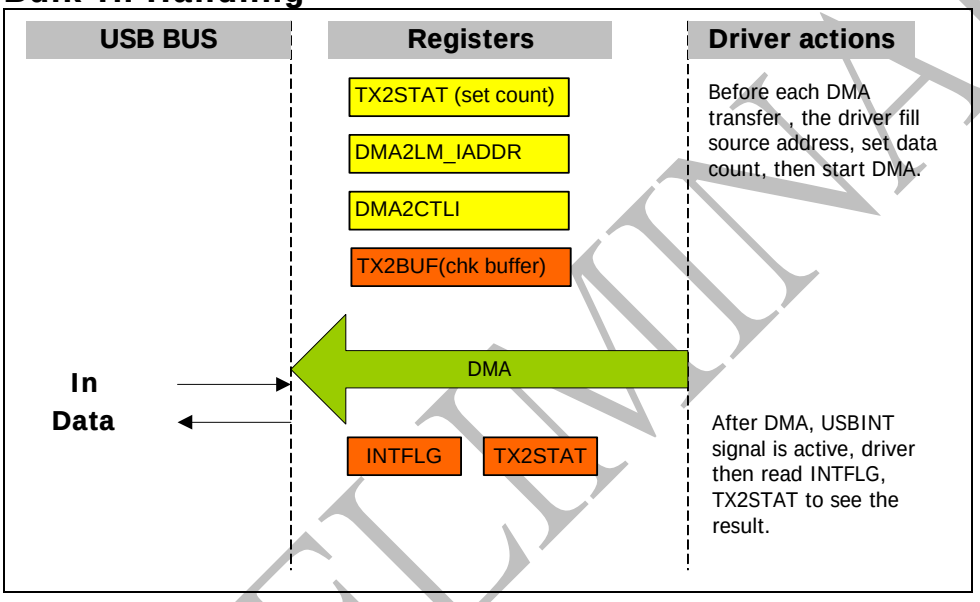
Control In Handling



Bulk Out Handling



Bulk In Handling



11.4.2 Programming sequence

1. INITIALIZE THE H/W

Before default control pipe activate, we do following steps (example):

- Pre allocate a buffer for DMA source or target (size > 512 bytes), get its begin address, here say lmbuffer.
- Do Soft disconnect, only set DEV_SELF_PWR bit in DEV_CTL register.
- If VBUS_STS bit in DEV_INFO equal 1, do power on reset to PHY --- set SOFT_POR in DEV_CTL, waiting more than 10ms and clear SOFT_POR.
- Configure EN_INT register:

Set EN_SETUP_INTR, EN_IN0_INTR, EN_OUT0_INTR in EN_INT register for endpoint 0.
Set EN_BOUT1_INTR in EN_INT register for endpoint 1.
Set EN_BIN2_INTR in EN_INT register for endpoint 2.
Set EN_IIN3_INTR in EN_INT register for endpoint 3.
Set EN_USBRST_INTR, EN_RESUME_INTR, EN_SUSP_INTR in EN_INT register to enable USB reset interrupt, resume interrupt and suspend interrupt.

- Set INTCON register, enable INTEN flag, configure the INTTYPE and INTMODE field.

• Configure endpoint 's control register:

Set TX0ACKINTEN, TX0NAK in TX0CON for endpoint 0.

Set RX0ACKINTEN, TX0NAK, EPOEN in RX0CON for endpoint 0.

Set RX1ACKINTEN, EP1EN in RX1CON for endpoint 1.

Set TX2DMADN_EN, TX2EPEN in TX2CON for endpoint 2.

Set TX3ACKINTEN, TX3VOIDINTEN, TX3ERRINTEN, TX3EPEN in TX3CON for endpoint 3.

• Do Soft disconnect, set DEV_SELF_PWR, DEV_SOFT_CN, CSR_DONE in DEV_CTL register.

CONTROL WRITE STEPS

Setup Stage:

SETUP transaction starts

UDC first acks host or indicates error for SETUP packet, then raise USBINT System enters the ISR for USBINT. In ISR, read the INTFLAG register, if SETUP_INTR event happened, it implies transaction has been acked:

• Read SETUP1, SETUP2 register for SETUP data, and parse the SETUP package

• From the SETUP data, if this is a 2 stage transaction, then go directly to the Status Stage.

• Program DMA destination address, DMA0LM_OADDR, to lmbuffer, write DMA0CTLO register to start DMA for the incoming OUT transactions.

Data Stage:

Out transaction starts

UDC moves DATA packet to lmbuffer, then raise USBINT.

In ISR, read the INTFLAG register, if OUT0_INTR event happened, then:

• Read RX0STAT register, check if RX0ACK flag is on, retrive RX0_CNT field, accumulate it to the total_data_length.

• Data length is RX0_CNT, if RX0_CNT $\neq$ 64 (Max Packet Size for Control), then mark this buffer as short_packet, then call upper_layer() to process lmbuffer, which knows how to process the short packet condition.

• Program DMA destination address, DMA0LM_OADDR, to lmbuffer, write DMA0CTLO register to start DMA for the incoming OUT transactions.

Status Stage:

In transactions starts

Since from Setup Stage we know the total_data_length, in Data Stage, we accumulate the data length received for each OUT transaction, when reached the total_data_length, we send a zero length IN transaction to the host:

• Set TX0STAT register, set TX0_CNT = 0

• Write DMA0CTLI to start DMA

• Wait for the TX0ACK in TX0STAT, if TX0ERR | TX0OVF | TX0URF happened, discard this transfer.

CONTROL READ STEPS

Setup Stage:

UDC first acks host or indicates error for SETUP packet, then raise USBINT System enters the ISR for USBINT. In ISR, read the INTFLAG register, if SETUP_INTR event happened, it implies transaction has been acked:

• Read SETUP1, SETUP2 register for SETUP data, and parse the SETUP packet

• From the SETUP data, if this is a 2 stage transaction, then directly go to the Status Stage.

- Wait for UDC dual port buffer available --- check TX0BUFFLG register for TX0_FULL status, then fill data buffer lmbuffer from upper_layer(), let DMA0LM_IADDR points to lmbuffer, then write DMA0CTLI to start DMA.

Data Stage:

In transaction starts.

From the last step, UDC will sent it to the host, then host will ACK, check TX0STAT for ack event, which cause the USBINT interrupt to occur In ISR, read the INTFLAG register, if IN0_INTR event happened, then:

- Read TX0STAT register, check if TX0ACK flag is on.
- Call upper layer() to put data in lmbuffer. Write TX0STAT register for data length TX0_CNT
- Wait for UDC dual port buffer available --- check TX0STAT register for TX0_BUF status, then fill data buffer lmbuffer from upper_layer(), let DMA0LM_IADDR points to lmbuffer, then write DMA0CTLI to start DMA

Status Stage:

OUT transaction start --- since from the Setup Stage we know the total_data_length --- it may be (i) the real data length that device can provide, or (ii) be specified on setup packet, choose either one which is smaller. In this stage, we accumulate the data length received on each IN transaction, when reached the total_data_length, host will send a zero length OUT packet.

Program DMA destination address, DMA0LM_OADDR, to dummy_buffer, write DMACTLO register to start next DMA for the incoming OUT transactions, though it has a zero length data packet.

USBINT will rise, and ISR entered:

- Read RX0STAT register, check if RX0ACK event is happened, if yes, the Control Read Transfer finished successfully.
- If RX0ERR | RX0OVF | RX0URF event happened, discard this transfer.

BULK OUT STEPS

Initialize for Endpoint:

- Program DMA destination address, DMA1LM_OADDR, to lmbuffer, write DMA1CTLO register to start DMA for the incoming OUT transactions.
- UDC moves DATA packet to lmbuffer, then raise USBINT.

OUT Transaction:

When USBINT rise, ISR will be entered:

- Read RX1STAT register, data length is RX1_CNT, if RX1_CNT $\neq$ 512 (Max Packet Size for Bulk), then mark this buffer as short_packet, then call upper_layer() to process lmbuffer, which knows how to process the short packet condition.
- If any of RX1ERR | RX1OVF | RX1URF happened, then discard this transaction.
- Prepare for next OUT transaction. Program next DMA destination address, DMA1LM_OADDR, and write DMA1CTLO register to start DMA.

BULK IN STEPS

Initialize for Endpoint:

Call upper_layer() to put data in lmbuffer, write TX2STAT register for data length TX2_CNT.

Let DMA2LM_IADDR points to lmbuffer, then write DMA2CTLI to start DMA.

In Transaction:

When USBINT rise, ISR will be entered, read TX2STAT register:

- If TX2ACK is set, then call upper_layer() to get next data in lmbuffer.
- If any of TX2ERR | TX2OVF | TX2URF happened, then discard this transaction.

- Prepare for next IN transaction --- wait for UDC dual port buffer available, check TX2 STAT register for TX2_FULL status, then fill data buffer lmbuffer from upper_layer(), let DMA2LM_IADDR points to lmbuffer, then write DMA2CTLI to start DMA.

Chapter 12 AHB-to-APB Bridge

12.1 Design Overview

12.1.1 Overview

The AHB-to-APB Bridge is designed for data transfer between AHB bus and APB bus. AHB-to-APB Bridge is the only bus master on the AMBA APB. In addition, the AHB-to-APB Bridge is also a slave on the AHB bus. The AHB-to-APB bridge unit converts system transfers into APB transfers and performs the following functions:

- Latches the address and holds it valid through the transfer.
- Decodes the address and generates a peripheral select. Only one select signal can be active a transfer.
- Drives the data onto the APB bus for a write transfer.
- Drives the data onto the AHB bus for a read transfer.
- Generate a timing strobe, PENABLE, for the transfer.

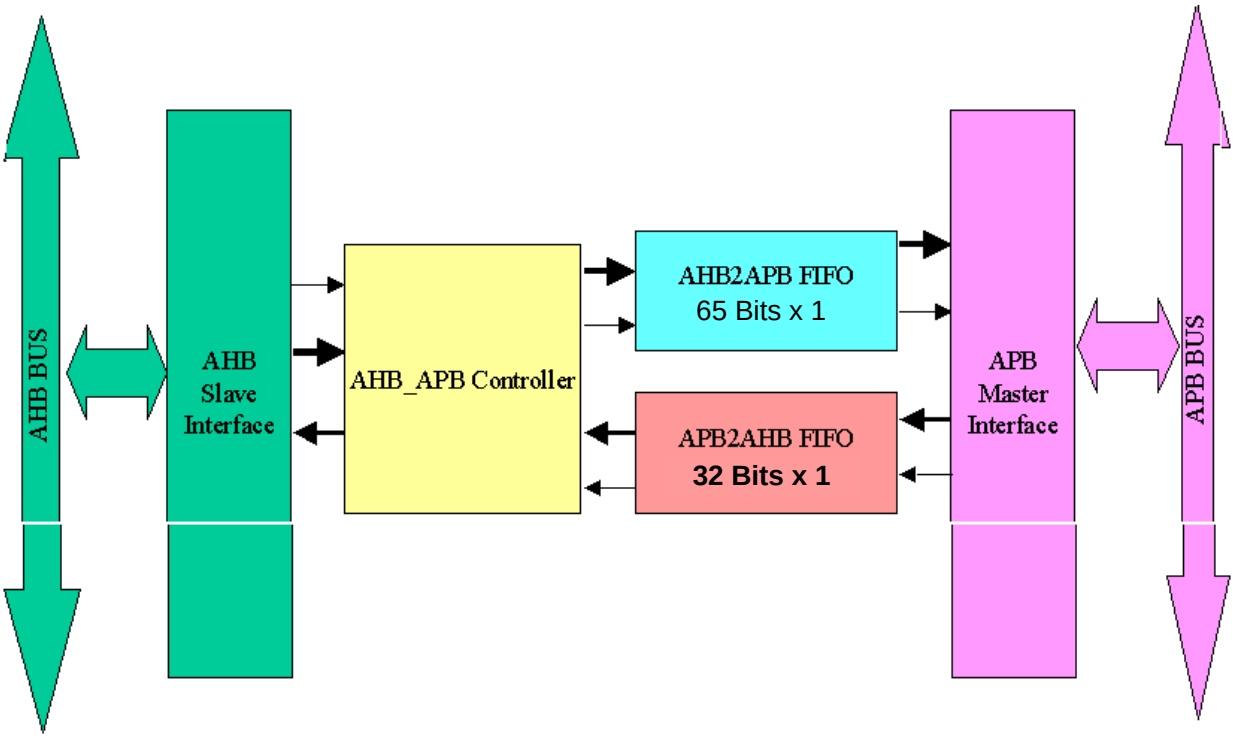
12.1.2 Features

- Supports AHB to APB bus protocol translation
- One AHB slave interface port for receiving and generating signals of AHB bus
- One APB master interface port for receiving and generating signals of APB system
- One one-depth FIFO between AHB bus and APB bus for queuing AHB inputs This FIFO will be clock-independent buffer interface between different clock domains. (The AHB bus clock domain and APB bus clock domain)
- Support byte, half word, and word data transfer size
- There is a waiting cycle for system write transfer
- There is a Retry response while the AHB-to-APB Bridge cannot complete the AHB master transfer immediately

12.2 Architecture

This section provides a description about the functions and behavior under various conditions.

12.2.1 Block Diagram



12.2.2 Block Descriptions

AHB-to-APB consist of an AHB slave interface to response the transfer from AHB bus and an APB master interface to issue transfer to the slave module on the APB bus. There are two FIFOs to buffer the data between the transfers of AHB bus and APB bus.

12.3 Functional Description

12.3.1 Operation

AHB Bus Interface

The AHB-to-APB Bridge AHB slave port only responds to transfer when it is selected by the decoder with hsel_brg input when the corresponding memory address belongs to peripherals is accessed. The AHB-to-APB Bridge first checks whether the transfer is accessible or not for it by referring to the status of inputs, hsel_brg and hreadyin. Then AHB-APB controller in AHB-to-APB Bridge will check whether the accessible transfer is valid or not by referring to the HTRANS, a NONSEQ or SEQ transfer is valid and the other two types of transfer type are not valid. AHB-to-APB Bridge will accept the transfer if the AHB transfer pass the checking. Whenever APB-bridge accepts a transfer, the AHB-APB controller will process the following procedure:

Classify the transfer-direction. If the transfer is a read transfer, then it will follow the READ-procedure and if the transfer is a write transfer, then it will follow the Write-procedure.

Write-procedure:

Check whether AHB-to-APB FIFO is full or not.

If AHB-to-APB FIFO is full, then assert AHB bus response, HRESP as RETRY case.

If AHB-to-APB FIFO is not full, then AHB-APB controller asserts AHB bus response HRESP as OKAY and set the waiting state for a cycle. At the next clock, AHB-APB controller fires a put-request to AHB-to-APB FIFO and push AHB address, AHB write control, and

AHB write data into AHB-to-APB FIFO.

Read procedure:

Check whether the AHB address makes an address-hit, that is, this read transfer has been fired before and the respective master has been re-granted and tries to get the required data again.

If current transfer does not make an address-hit, then AHB-APB controller will check whether the read-address-record register is empty or not. If the register is not empty, it means that there is read transfer has been fired before and the transfer has not been completed, then the current transfer will cause a retry response and waiting for processing next time.

If current transfer does not make an address-hit and the read-address-record register is empty, then AHB-APB controller will check whether the AHB-to-APB FIFO is full or not. If the AHB-to-APB FIFO is full, then it means that the APB master port is busy and there is another transfer has been queued in the FIFO, so the current transfer will be abandoned and cause a retry response.

If current transfer does not make an address-hit and read-address-record register is empty, and the AHB-to-APB FIFO is not full, AHB-APB controller will fire a put-request to AHB-to-APB FIFO and push the AHB address, AHB read control into FIFO.

If current transfer makes an address-hit and the APB-to-AHB FIFO is empty, then it means although this transfer has been fired to APB bus before, but the accessed data is not ready now. So the master who issues this transfer will need to wait for the data next time. The AHB-APB controller will assert a retry response to AHB bus.

If current transfer makes an address-hit and the APB-to-AHB FIFO is not empty, then it means that the read data is prepared for current transfer. AHB-APB controller will assert hready_brg and respond an OKAY reply to AHB bus. At the same time, AHB-APB controller will fire a get-request to APB-to-AHB FIFO for popping the required data.

APB Bus Interface

The APB master interface will get the transfer control and data form the AHB-to-APB FIFO and start the state machine to drive the relative control signals and data to APB bus. The APB master also get read data from peripherals on APB bus and push the read data into APB-to-AHB FIFO while a read transfer is processing. The procedure is illustrated below:

First, APB master will check whether there is a transfer to be processed by referring to the condition of AHB-to-APB FIFO. If the AHB-to-APB is not empty, then it means an accessible and valid transfer needs to be processed by APB master.

The APB master will start the state machine to enter the setup phase and drive the relative control signals and data signals to APB bus.

Following the setup phase, the state machine in APB master will go into the enable phase. In this phase, the APB master will assert the data strobe signal, PENABLE, and get the read data from APB bus if the transfer is a read-transfer.

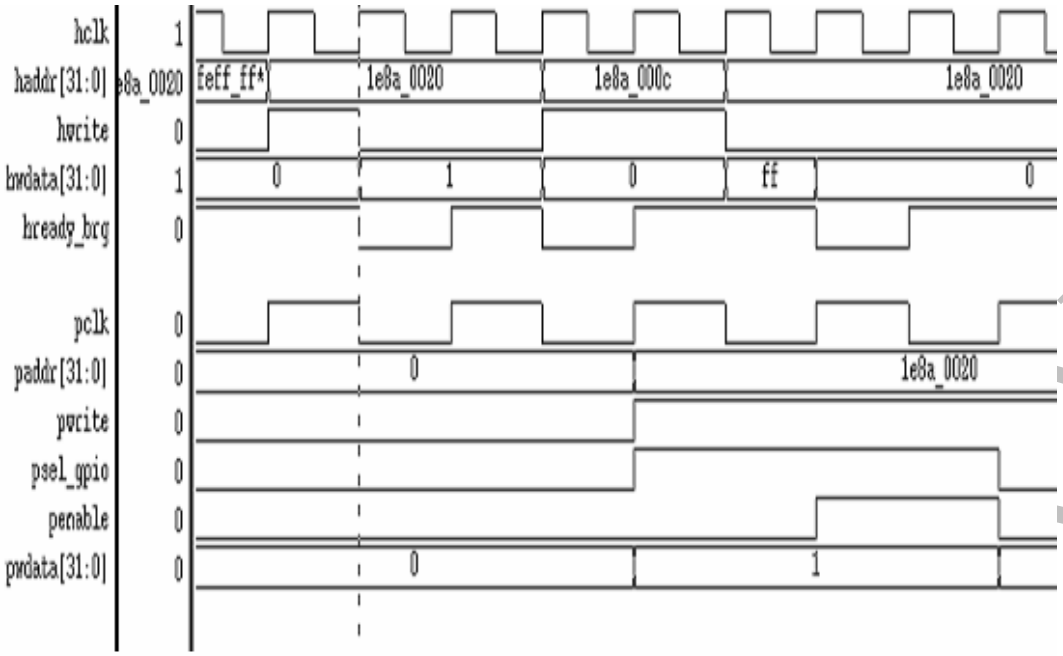
At the same time of getting the read data from APB bus, the APB master fires and puts-request to APB-to-AHB FIFO to push the read data into the FIFO.

After enable phase, the state machine in AHB-to-APB bridge will check whether there is any other transfer waiting for service or not and start the iteration illustrated before.

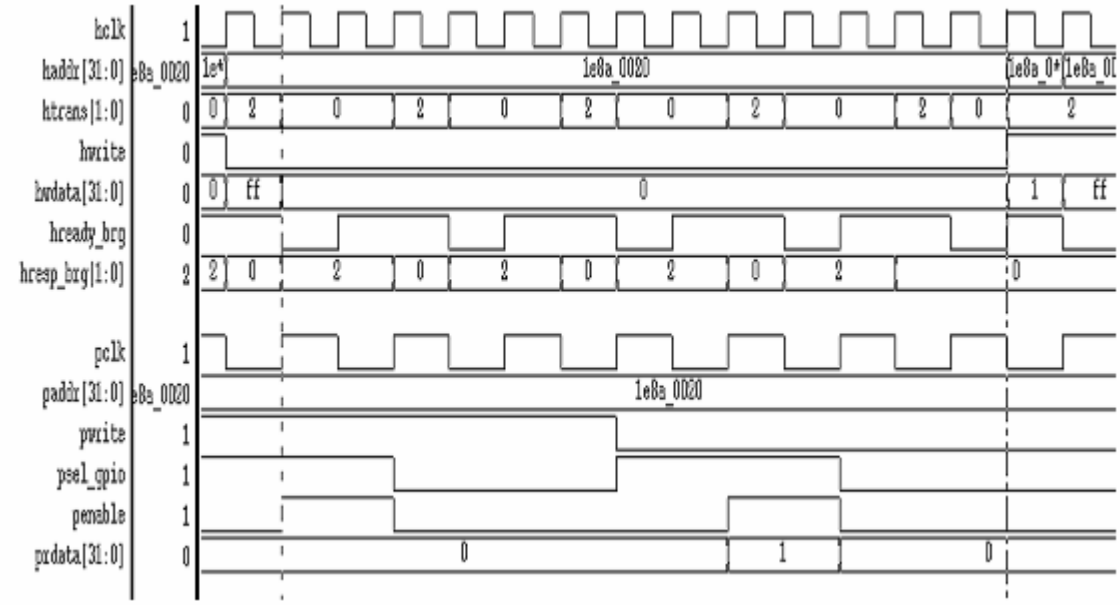
The bridge actives as AHB slave in AHB bus and APB master in APB bus. The slave interface in bridge handles APB data transfer. Because hclk, which is used in AHB components, and pclk, which is used in APB components, are different, so bridge provides write and read buffer to improve bus utilization. For example, if AHB master wants to write data to APB device, the bridge will capture write data that are from AHB master in write buffer and transfer the write data from write data buffer to APB device. It can avoid insetting wait state to drop off AHB bus utilization. When AHB master wants to read data from APB device, the bridge is designed to use RETRY response to AHB master and captures the read address to send to APB device to read the data that AHB master wants. To avoid insetting wait state except two cycle response to improve bus utilization between data transfer in different clock domain. There are two waveforms below to describe data

transfer between AHB bus and APB bus.

1. Write Transfer Waveform from AHB



2. Read Transfer Waveform from AHB



Chapter 13 UART (16550)

13.1 Design Overview

13.1.1 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.

13.1.2 Features

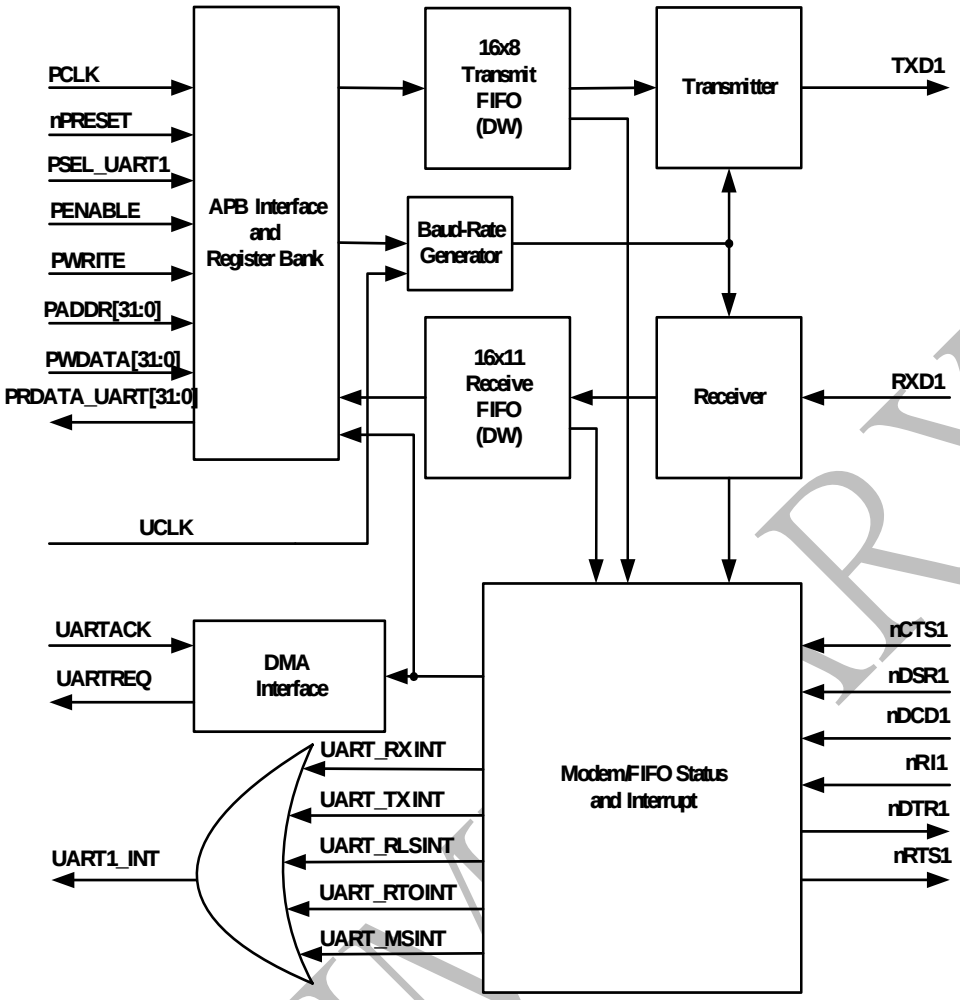
- Compliance to AMBA APB specification
- Supports up to 115.2Kbps baud-rate

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

13.2 Architecture

This section provides a description about the functions and behavior under various conditions.

13.2.1 Block Diagram



13.2.2 Block Descriptions

The UART consist 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.

13.3 Registers

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

13.3.1 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	0x00000007	Line Control register.

UART_MCR	0x0010	W	0x00000000	Modem Controls register.
UART_LSR	0x0014	W	0x00000006	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

13.3.2 Detail Register Description

UART_RBR
Address: Operational Base + offset(0x00)
Receive Buffer Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Any data words received by the UART form 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(0x00)
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(0x04)
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(0x08)
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**	**1st**	Receiver line status	Parity, Overrun or Framing errors or Break Interrupt.	Reading the Line Status Register.
0	**1**	**0**	**2nd**	Receive Data available	FIFO trigger level reached.	FIFO trigger level reached.
1	**1**	**0**	**3rd**	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**	**4th**	Transmitter holding register empty	Transmitter Holding Register Empty.	Writing to the Transmitter Holding Register or reading the IIR or LSR.
0	**0**	**0**	**5th**	Modem status	nCTS, nDSR, nRI or nDCD.	Reading the Modem status register.

UART_FCR

Address: Operational Base + offset(0x08)

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
5:3	-	-	Reserved
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	Ignored (Used to enable FIFOs in NS16550D). Since this UART only supports FIFO mode, this bit is ignored.

UART_LCR

Address: Operational Base + offset(0x0C)

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	0x1	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(0x10)

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(0x14)

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(0x18)

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(0x00)

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(0x04)
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

13.4 Functional Description

13.4.1 Clock Signals

The frequency selected for UCLK must accommodate the desired range of baud rates:
16 x 65535 x baud rate >= F(UCLK) >= 16 x 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.

13.4.2 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 is 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.

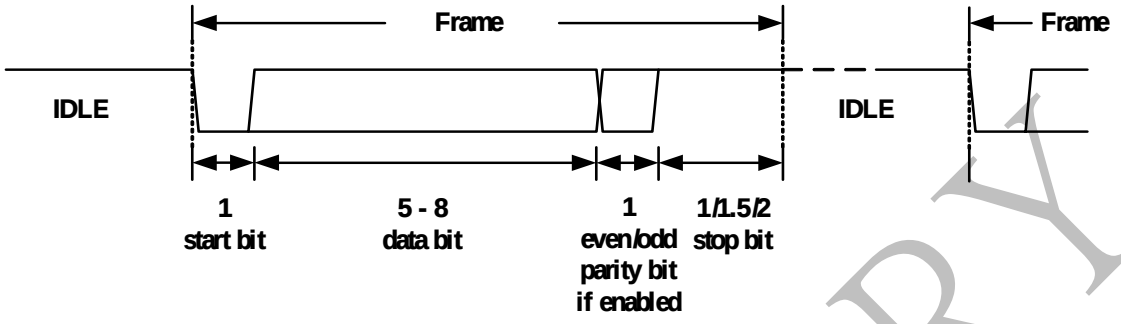
If the start bit was valid, successive data bits are sampled on every 16^{th} cycle of sample clock (one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.

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 14 General Purpose IO (GPIO)

14.1 Design Overview

14.1.1 Overview

The General Purpose IO (GPIO) is an APB slave module that connects to the APB bus. The GPIO has up to 32 bits programmable input/output organized as four ports, Port A, Port B, Port C and Port D. Pins of both ports can configured as either inputs or outputs. Interrupt signal can be generated depending on a level, or a transitional value of a pin.

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.

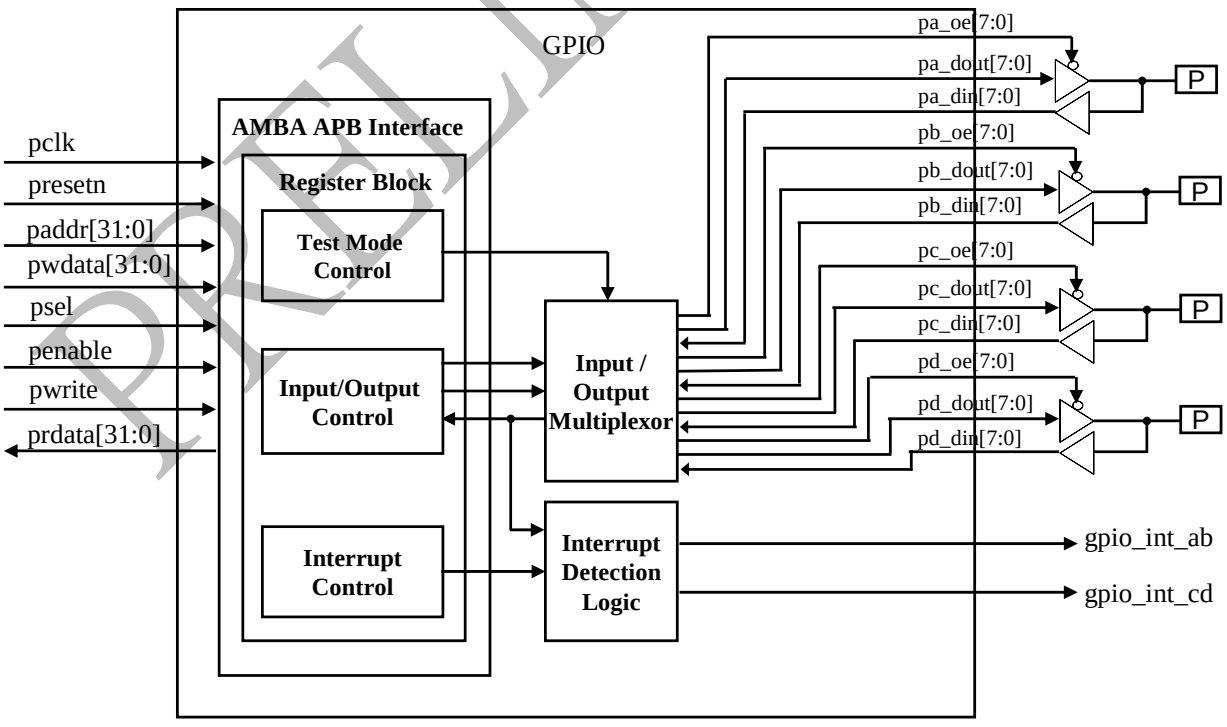
14.1.2 Features

- Compliance to the AMBA APB interface.
- Up to 32 individually programmable input/output pins.
- Control word read-back capability.
- All ports are inputs on system reset.
- The direction registers are programmable.
- Programmable interrupt generation capability, from a transition or a level condition, on any number of pins.

14.2 Architecture

This section provides a description about the functions and behavior under various conditions.

14.2.1 Block Diagram



14.2.2 Block Descriptions

The GPIO has up to 32 bits programmable input/output organized as four ports, Port

A, Port B, Port C, and Port D. The CPU reads and writes data and control/status information to and from GPIO via APB interface.

- AMBA APB Interface
  - The APB interface generates read and write decodes for accesses to control, interrupt, and data registers within the GPIO.
  - It implements the storage elements for the data, data direction, test mode control, and interrupt interface registers.
- Interrupt detection logic
  - The GPIO has the ability to generate mask-programmable interrupts based on the level, or the transitional value of any of its GPIO lines.
  - The *General Purpose Input Output Interrupt* (gpio_int_ab and gpio_int_cd) indicates to an interrupt controller that an interrupt has occurred in one or more of the GPIO lines. Bit reads HIGH in gpio_int_ab reflects that interrupt has occurred in Port A or Port B. Bit reads HIGH in gpio_int_cd reflects that interrupt has occurred in Port C or Port D.
  - Twenty registers in AMBA APB interface, each controlling a different feature or condition in the interrupt triggering chain.
  - Interrupt generation either on a change in the level, one edge, or both edges of the GPIO lines.
- Input/Output Control

  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.

14.3 Registers

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

14.3.1 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_PCDR	0x0010	W	0x00000000	Port C data register.
GPIO_PCCON	0x0014	W	0x00000000	Port C direction register.
GPIO_PDDR	0x0018	W	0x00000000	Port D data register.
GPIO_PDCON	0x001C	W	0x00000000	Port D direction register.
GPIO_TEST	0x0020	W	0x00000000	GPIO function test register.
GPIO_IEA	0x0024	W	0x00000000	Port A interrupt mask register.
GPIO_IEB	0x0028	W	0x00000000	Port B interrupt mask register.
GPIO_IEC	0x002C	W	0x00000000	Port C interrupt mask register.
GPIO_IED	0x0030	W	0x00000000	Port D interrupt mask register.
GPIO_ISA	0x0034	W	0x00000000	Port A interrupt sense register.
GPIO_ISB	0x0038	W	0x00000000	Port B interrupt sense register.
GPIO_ISC	0x003C	W	0x00000000	Port C interrupt sense register.
GPIO_ISD	0x0040	W	0x00000000	Port D 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_IBEC	0x004C	W	0x00000000	Port C interrupt both-edges register.
GPIO_IBED	0x0050	W	0x00000000	Port D 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_IEVC	0x005C	W	0x00000000	Port C interrupt event register.
GPIO_IEVD	0x0060	W	0x00000000	Port D interrupt event register.
GPIO_ICA	0x0064	W	0x00000000	Port A interrupt clear register.
GPIO_ICB	0x0068	W	0x00000000	Port B interrupt clear register.
GPIO_ICC	0x006C	W	0x00000000	Port C interrupt clear register.
GPIO_ICD	0x0070	W	0x00000000	Port D 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

14.3.2 Detail Register Description

GPIO_PxDR(x=A, B, C, D)
The GPIO_PxDR is the Port x data register.
Address: Operational Base + offset (0x00, 0x08, 0x10, 0x18)

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x data register.

GPIO_PxCON(x=A, B, C, D)
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 configured as input. At reset, all bits are configured as input.
Address: Operational Base + offset (0x04, 0x0C, 0x14, 0x1C)

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Port x direction register. 0: Input 1: Output

GPIO_TEST
GPIO function test register.
Address: Operational Base + offset (0x20)

bit	Attr	Reset Value	Description
31:3	-	-	Reserved.
2:1	RW	0x0	Test mode. 0x0 – PORTB -> PORTA 0x1 – PORTA -> PORTB 0x2 – PORTD -> PORTC 0x3 – PORTC -> PORTD
0	RW	0x0	Test mode enable indicator. '1' – The GPIO loop-back test mode enable. '0' – Normal mode.

GPIO_IEx (x=A, B, C, D)
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.
Address: Operational Base + offset (0x24, 0x28, 0x2C, 0x30)

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, C, D)
The GPIO_ ISx register is the Port x interrupt sense register. The default set is edge sensitive at reset.
Address: Operational Base + offset (0x34, 0x38, 0x3C, 0x40)

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, C, D)
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.
Address: Operational Base + offset (0x44, 0x48, 0x4C, 0x50)

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, C, D)
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.
Address: Operational Base + offset (0x54, 0x58, 0x5C, 0x60)

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, C, D)
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.
Address: Operational Base + offset (0x64, 0x68, 0x6C, 0x70)

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

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.

Address: Operational Base + offset (0x74)

bit	Attr	Reset Value	Description
31:24	R	0x0	Port D interrupt status register. The status of interrupts trigger conditions detection on pins. 0: GPIO interrupt not active. 1: GPIO asserting interrupt.
23:16	R	0x0	Port C interrupt status register. The status of interrupts trigger conditions detection on pins. 0: GPIO interrupt not active. 1: GPIO asserting interrupt.
15:8	R	0x0	Port B interrupt status register. The status of interrupts trigger conditions detection on pins. 0: GPIO interrupt not active. 1: GPIO asserting interrupt.
7:0	R	0x0	Port A interrupt status register. The status of interrupts trigger conditions detection on pins. 0: GPIO interrupt not active. 1: GPIO asserting interrupt.

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

14.4 Functional Description

14.4.1 Operation

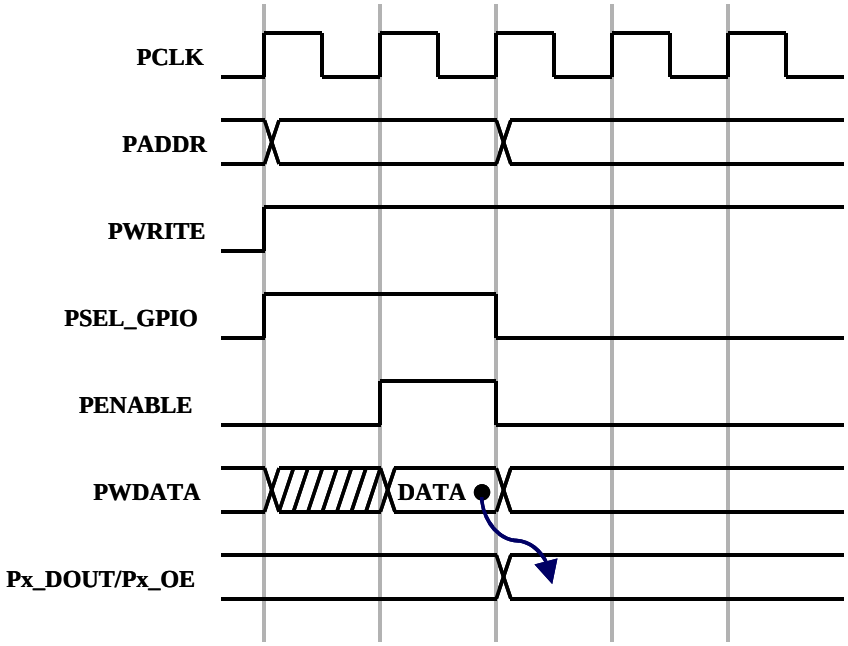
For each port, there is a data register and a mode register. On reads, the data register holds the current status of the corresponding port pins, whether they are configured as inputs or outputs. Writing to a data register only affects the pins that are configured as outputs. Those function pins shared 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, interrupt output **gpio_int_ab or gpio_int_cd** can be sent to the interrupt controller.

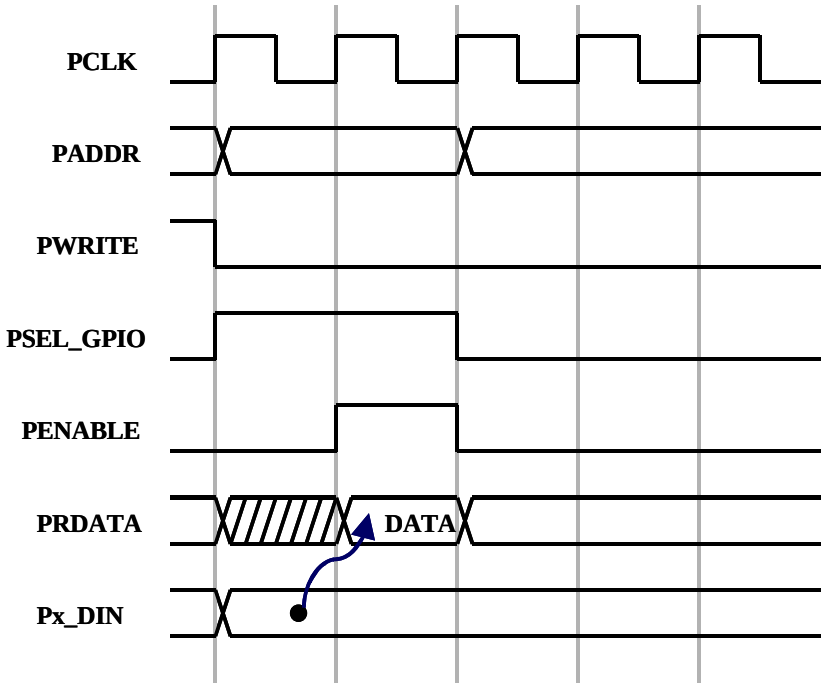
Port Mode registers:

The Port Mode registers operate in a different manner on each port. The input, output and function pin sharing are defined in the register section.

Write operation timing:



Read operation timing:



14.4.2 Programming sequence

- Rising edge detected

Bits set to "0" in GPIO_ISx (x=A, B, C and D), edge on corresponding pin is detected.

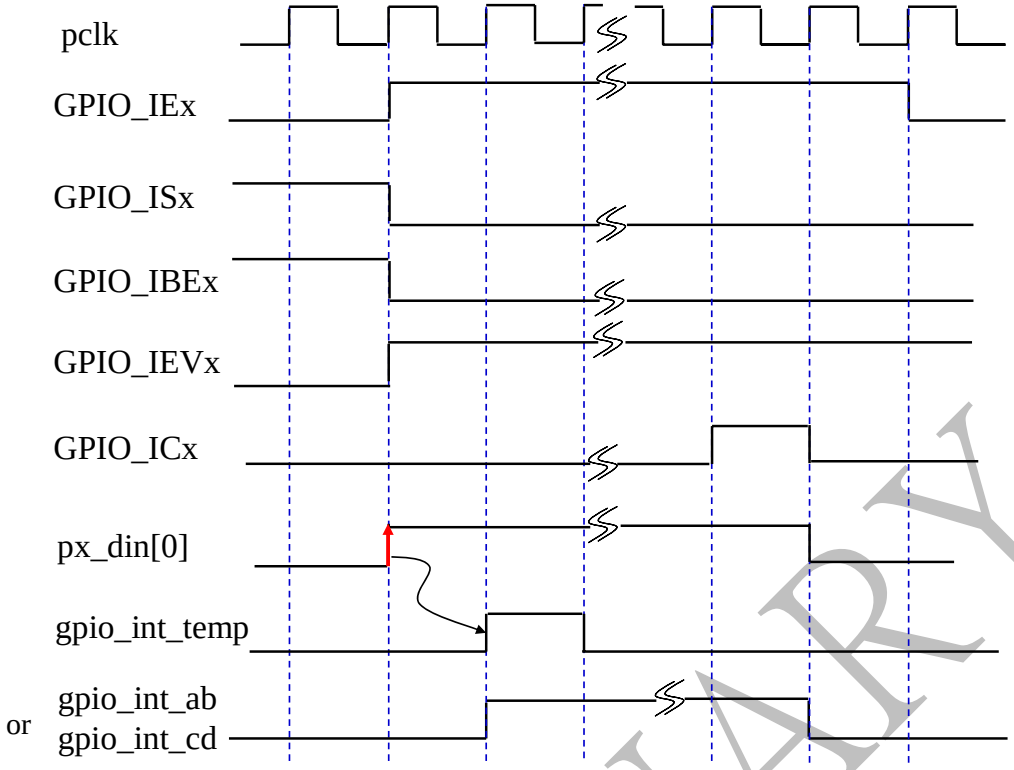
Bits set to "0" in GPIO_IBEx (x=A, B, C and D) configure the corresponding pin to detect single edge.

Bits set to "1" in GPIO_IEVx (x=A, B, C and D) configure the corresponding pin to detect rising edge.

Bits set to "1" in GPIO_IEx (x=A, B, C and D) allow the corresponding pins to trigger their individual interrupts.

When one or more input data pins cause a rising edge-triggered interrupt, interrupt output **gpio_int_ab or gpio_int_cd** can be set to "1".

Rising edge detected timing



- Falling edge detected

Bits set to "0" in GPIO_ISx (x=A, B, C and D), edge on corresponding pin is detected.

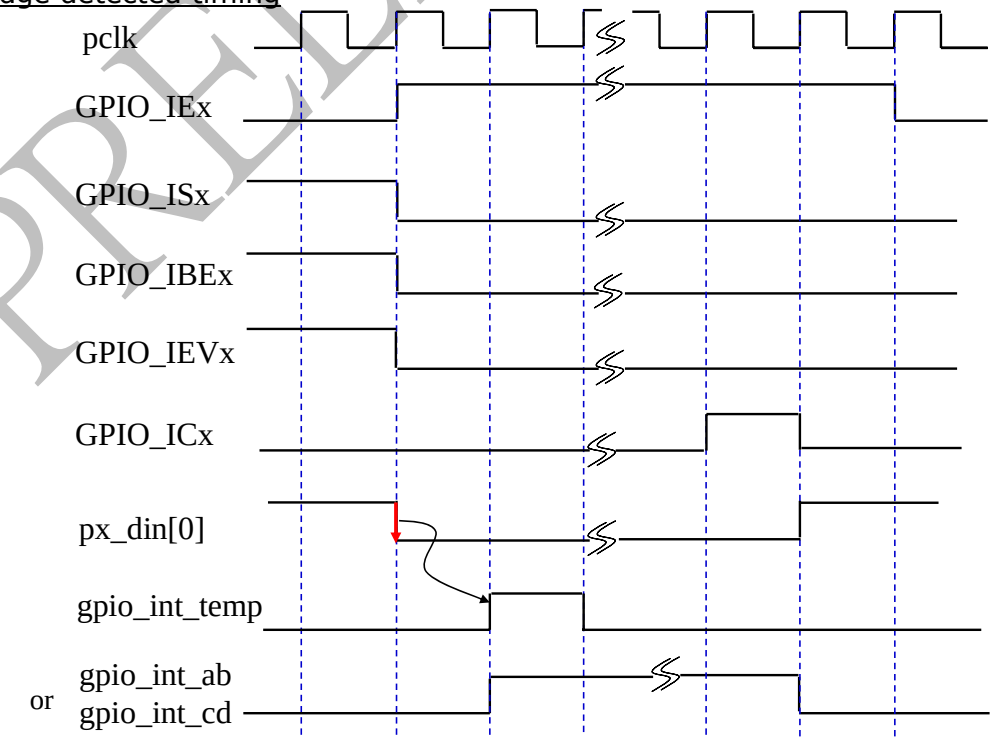
Bits set to "0" in GPIO_IBEx (x=A, B, C and D) configure the corresponding pin to detect single edge.

Bits set to "0" in GPIO_IEVx (x=A, B, C and D) configure the corresponding pin to detect rising edge.

Bits set to "1" in GPIO_IEx (x=A, B, C and D) allow the corresponding pins to trigger their individual interrupts.

When one or more input data pins cause a falling edge-triggered interrupt, interrupt output **gpio_int_ab or gpio_int_cd** can be set to "1".

Falling edge detected timing



- Both-edges detected

Bits set to "0" in GPIO_ISx (x=A, B, C and D), edge on corresponding pin is detected.

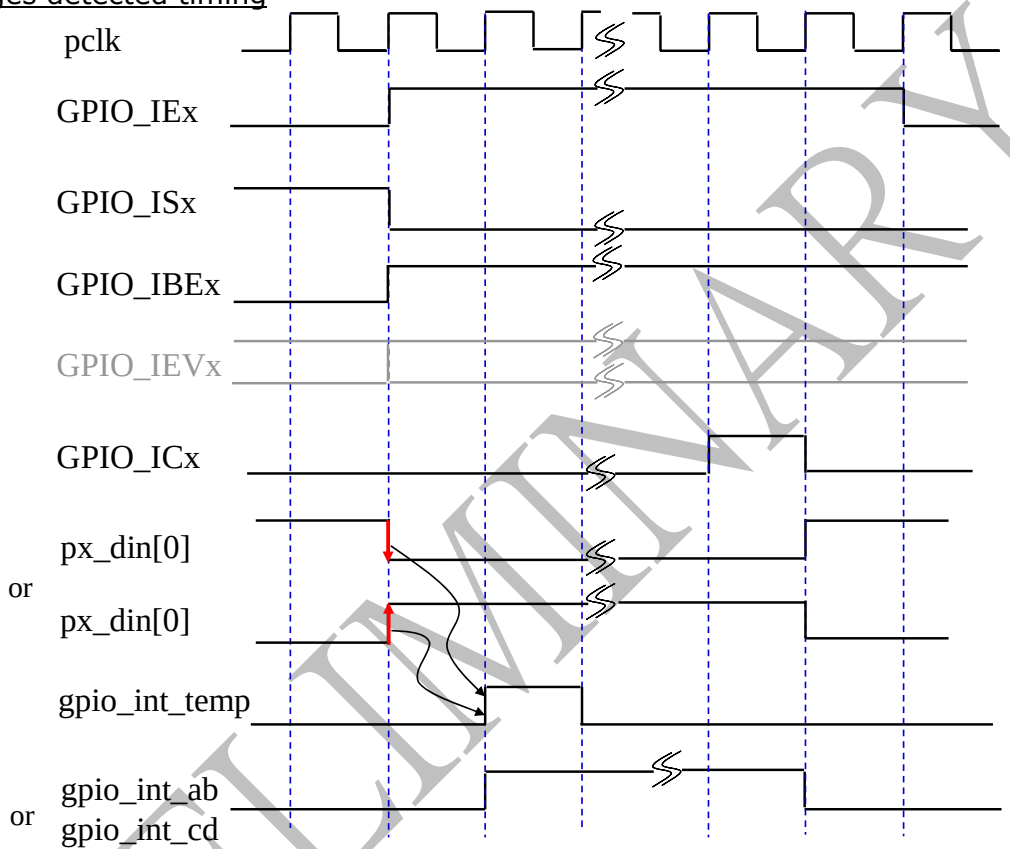
Bits set to "1" in GPIO_IBEx (x=A, B, C and D) configure the corresponding pin to detect both edges.

Bits set to "0" or "1" in GPIO_IEVx (x=A, B, C and D) are okay.

Bits set to "1" in GPIO_IEx (x=A, B, C and D) allow the corresponding pins to trigger their individual interrupts.

When one or more input data pins cause a falling edge-triggered or rising edge-triggered interrupt, interrupt output **gpio_int_ab or gpio_int_cd** can be set to "1".

Both-edges detected timing



- High level detected

Bits set to "1" in GPIO_ISx (x=A, B, C and D), level on corresponding pin is detected.

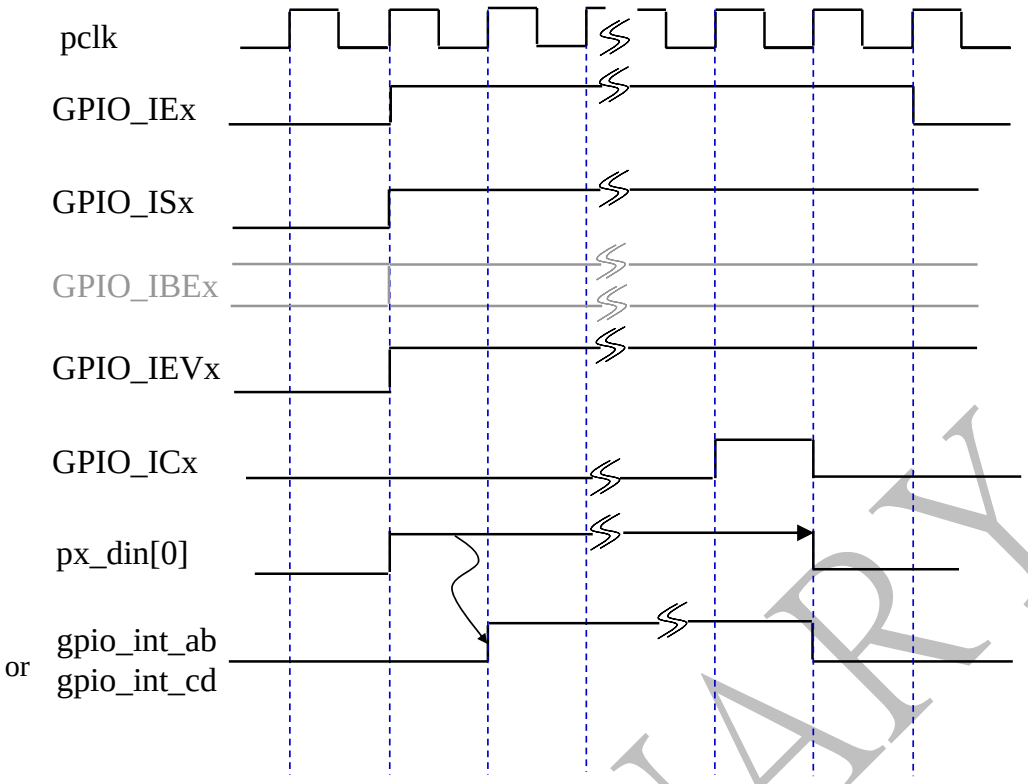
Bits set to "0" or "1" in GPIO_IBEx (x=A, B, C and D) are okay.

Bits set to "1" in GPIO_IEVx (x=A, B, C and D) configure the corresponding pin to detect high level.

Bits set to "1" in GPIO_IEx (x=A, B, C and D) allow the corresponding pins to trigger their individual interrupts.

When one or more input data pins causes a high level interrupt, interrupt output **gpio_int_ab or gpio_int_cd** can be set to "1".

High level detected timing



- Low level detected

Bits set to "1" in GPIO_ISx (x=A, B, C and D), level on corresponding pin is detected.

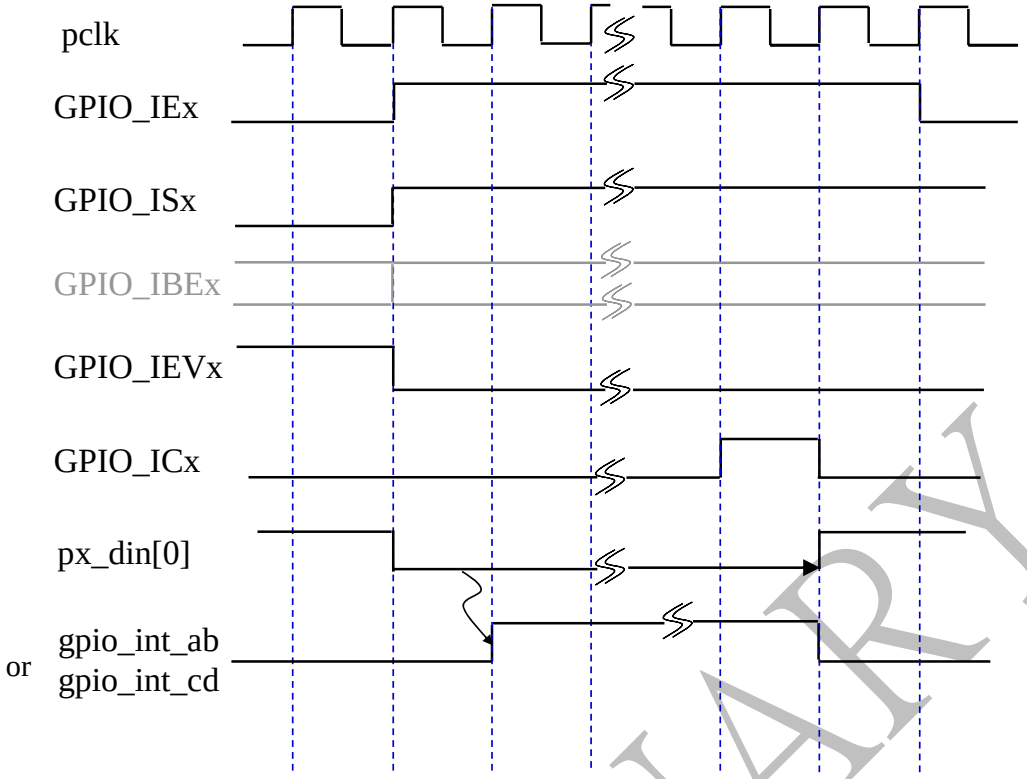
Bits set to "0" or "1" in GPIO_IBEx (x=A, B, C and D) are okay.

Bits set to "0" in GPIO_IEVx (x=A, B, C and D) configure the corresponding pin to detect low level.

Bits set to "1" in GPIO_IEx (x=A, B, C and D) allow the corresponding pins to trigger their individual interrupts.

When one or more input data pins causes a low level interrupt, interrupt output **gpio_int_ab or gpio_int_cd** can be set to "1".

Low level detected timing



14.5 GPIO MUX

MUX0	MUX1	MUX2		Config register	register bit
PORT-A	LCD_DATA16	RXD0	PULL-UP	SCU_IOMUXA_CON[1:0]	gpioa_lcd16_rxd0_sel
	LCD_DATA17	TXD0	PULL-UP	SCU_IOMUXA_CON[3:2]	gpioa_lcd17_txd0_sel
	FLASH CS1		PULL-UP	SCU_IOMUXA_CON[9]	
	LCD_HSYNC		PULL-UP	SCU_IOMUXA_CON[11]	
PORT-B[0:5]	SD-DATA0	SPI_MISO	PULL-UP	SCU_IOMUXA_CON[13:12]	gpiob_sd_spi_sel
	SD-CMD	SPI-MOSI	PULL-UP		
	SD-CLK	SPI_SCLK	PULL-UP		
SCL	FLASH CS2	PORT-B[6]	PULL-UP	SCU_IOMUXA_CON[15:14]	I2c_flashcs2_gpiob_sel
SDA	FLASH CS3	PORT-B[7]	PULL-UP	SCU_IOMUXA_CON[17:16]	I2c_flashcs3_gpiob_sel
PORT-C			PULL-DOWN		
	LCD DATAEN		PULL-DOWN	SCU_IOMUXA_CON[10]	
	SCLK		PULL-DOWN	SCU_IOMUXB_CON[0]	gpioc_i2ssclk_sel
	LRCK		PULL-DOWN	SCU_IOMUXB_CON[1]	gpioc_i2slrck_sel
	DATAIN		PULL-DOWN	SCU_IOMUXB_CON[2]	gpioc_i2ssdi_sel
	DATAOUT		PULL-DOWN	SCU_IOMUXB_CON[3]	gpioc_i2ssdo_sel
	MCLK		PULL-DOWN	SCU_IOMUXB_CON[4]	gpioc_i2scko_sel
	ST_CEN[1]		PULL-DOWN	SCU_IOMUXB_CON[5]	gpioc_stcs1_sel
PORT-D	SDMMC_PCA	TXD1	PULL-UP	SCU_IOMUXB_CON[7:6]	gpiod_sdpca_txd1_sel
	SDMMC_CDA	RXD1	PULL-UP	SCU_IOMUXB_CON[9:8]	gpiod_sdcda_rxd1_sel
	SDMMC_WPA		PULL-UP	SCU_IOMUXB_CON[10]	gpiod_sdwpa_sel
	SDRAM CKE		PULL-UP	SCU_IOMUXB_CON[21]	gpiod_sdcke_sel
	PWM0			SCU_IOMUXB_CON[11]	gpiod_pwm0_sel
	PWM1		PULL-DOWN	SCU_IOMUXB_CON[12]	gpiod_pwm1_sel
	PWM2		PULL-DOWN	SCU_IOMUXB_CON[13]	gpiod_pwm2_sel
PORT-E	LCD_DATA8		PULL-UP	SCU_IOMUXB_CON[15]	gpioe_lcd_se

	LCD_DATA9		PULL-UP		
	LCD_DATA10		PULL-UP		
	LCD_DATA11		PULL-UP		
	LCD_DATA12		PULL-UP		
	LCD_DATA13		PULL-UP		
	LCD_DATA14		PULL-UP		
	LCD_DATA15		PULL-UP		
PORT-F	VIP_CLKO		PULL-UP	SCU_IOMUXB_CON[16]	gpiof_vipclk_sel
	A11		PULL-UP	SCU_IOMUXB_CON[17]	sdtaddr11_gpiof_sel
	A12		PULL-UP	SCU_IOMUXB_CON[18]	sdtaddr12_gpiof_sel

Chapter 15 Timers

15.1 Design Overview

15.1.1 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.

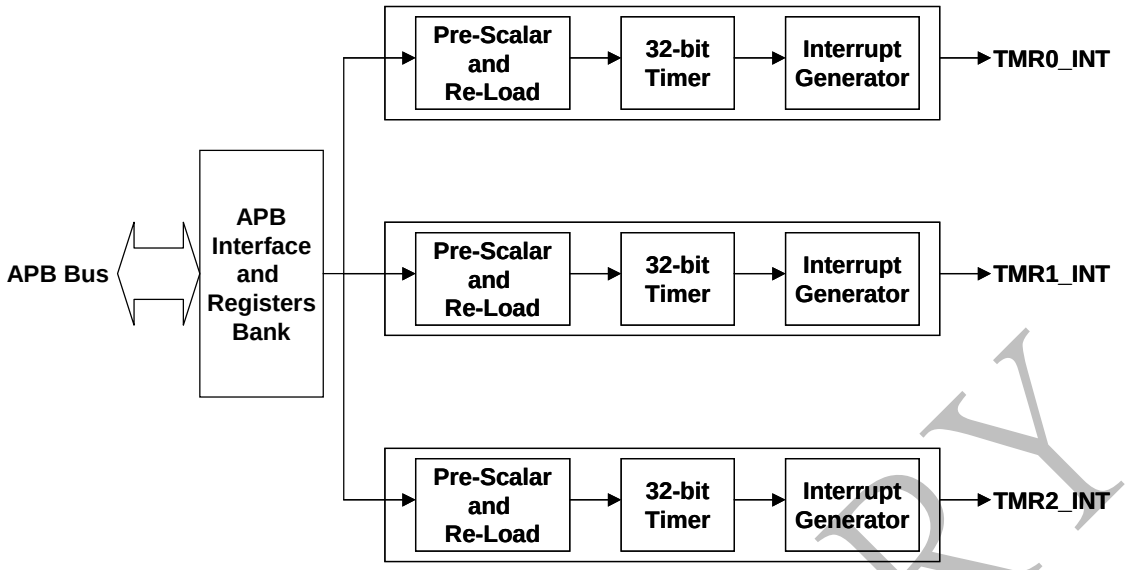
15.1.2 Features

- Built-in three 32 bits timer modules
- Provide independent free-running or periodical mode
- Mask-able interrupts
- Programmable counter

15.2 Architecture

This section provides a description about the functions and behavior under various conditions.

15.2.1 Block Diagram



15.2.2 Block Descriptions

The Timer in Leopard Generic Core consist 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.

15.3 Registers

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

15.3.1 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

15.3.2 Detail Register Description

TMRxLR(x=0,1,2)
Address: Operational Base + offset(0x00, 0x10, 0x20)
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(0x04, 0x14, 0x24)
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(0x08, 0x18, 0x28)
Control register

bit	Attr	Reset Value	Description
31:9	-	-	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 control bit. 0:Edge-triggered 1:Level-triggered
0	RW	-	Reserved.

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

15.4 Functional Description

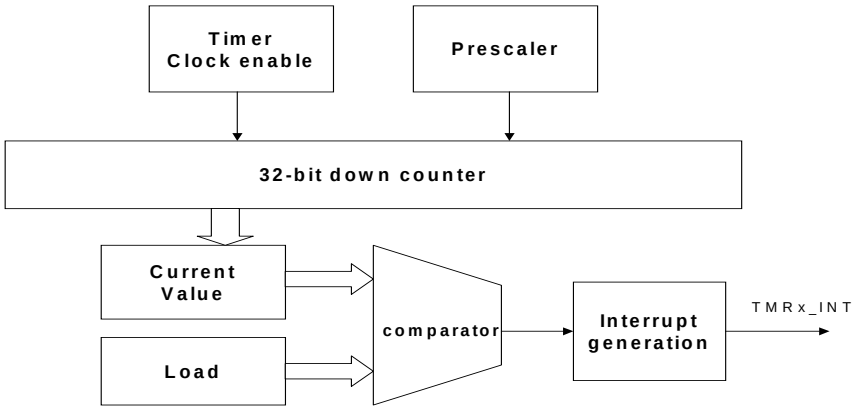
15.4.1 Operation

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:

$$F_{timer} = F_{(PCLK \text{ or } TMRCLKIN)} / (\textbf{Pre-scale} * \textbf{Counter Value})$$

Chapter 16 Watchdog Timer (WDT)

16.1 Design Overview

16.1.1 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.

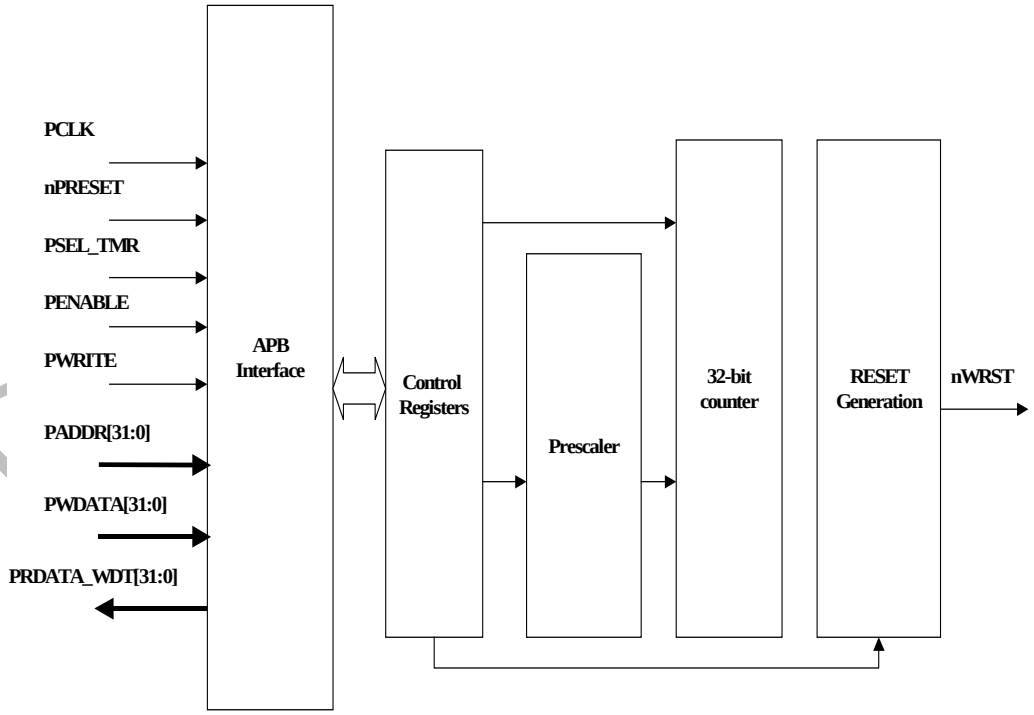
16.1.2 Features

- Watchdog function
- Built-in 32 bits programmable reset counter

16.2 Architecture

This section provides a description about the functions and behavior under various conditions.

16.2.1 Block Diagram



16.2.2 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.

16.3 Registers

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

16.3.1 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

16.3.2 Detail Register Description

WDTLR
Address: Operational Base + offset(0x00)
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(0x04)
Current value register.

Bit	Attr	Reset Value	Description
31:0	R	0xFFFF	The current counter value.

WDTCON
Address: Operational Base + offset(0x08)
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

16.4 Functional Description

16.4.1 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 17 Real Time Clock (RTC)

17.1 Design Overview

17.1.1 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.

17.1.2 Features

- 24 hour time mode with highest precision of one sixteenth of a second
- Calendar function with correction for leap year
- 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

RTC function is no included in RK2706/RK2708/Rk2710 chips. For detail descriptions, please refer **RK27xx Real Time Clock. PDF**.

Chapter 18 SPI Master Controller

18.1 Design Overview

18.1.1 Overview

The Serial Peripheral Interface (SPI) master 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 controller system is flexible enough to interface directly with numerous commercially available peripherals.

The SPI master controller is compatible with above-mentioned protocols as SPI bus master. At the host side, the core acts like an APB compliant slave device.

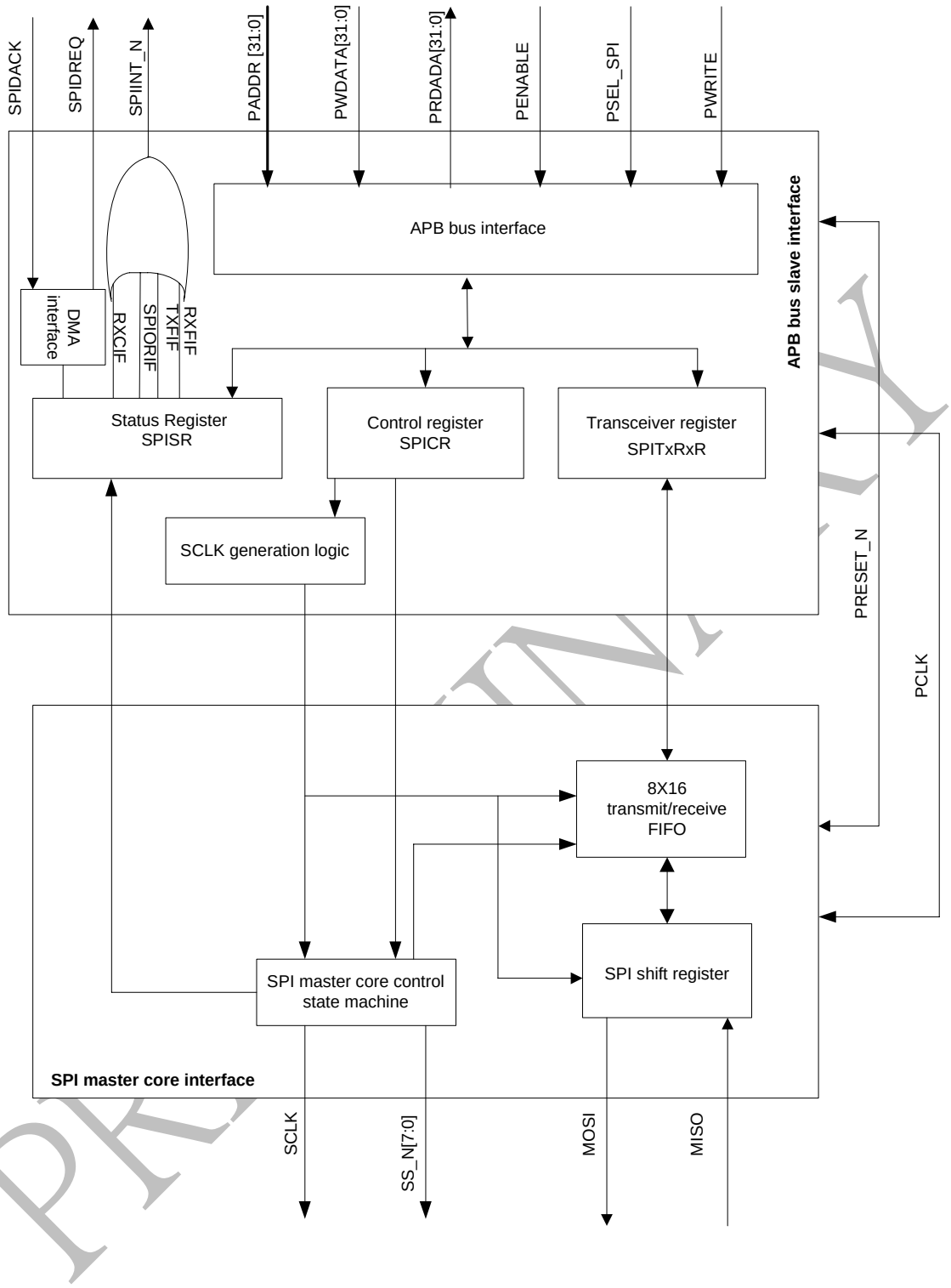
18.1.2 Features

- AMBA APB slave interface
- 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 up to 8 slave devices
- MSB or LSB first data transfer

18.2 Architecture

This section provides a description about the functions and behavior under various conditions.

18.2.1 Block Diagram



18.2.2 Block Descriptions

The SPI master controller consist of an APB Slave interface for control and transceiver register setting and a SPI master controller interface that generate SPI signals for peripherals. The transmit/receive FIFO used to buffer the data between APB slave interface and SPI master controller.

18.3 Registers

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

18.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SPI_TxR	0x0000	W	0x00000000	SPI master controller transmit FIFO input
SPI_RxR	0x0000	W	0x00000000	SPI master 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 controller FIFO control register
SPI_FWCR	0x000C	W	0x00000100	SPI master controller transaction flow control register
SPI_DLYCR	0x0010	W	0x00000000	SPI master controller delay control register
SPI_TxCR	0x0014	W	0x00000000	Transmit counter
SPI_RxCR	0x0018	W	0x00000000	Receive counter
SPI_SSCR	0x001C	W	0x00000000	SPI master controller slave select and characteristic register
SPI_ISR	0x0020	W	0x00000000	SPI master controller interrupt status register

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

18.3.2 Detail Register Description

SPI_TxR
Address: Operational Base + offset(0x00)
This register contains data to be transmitted on the SPI master 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(0x00)
This register contains the data received from the SPI master 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(0x04)
This register contains the bits to control the interrupt generation of this SPI master controller

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
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). This bit enables receive transfer complete interrupt each time a receive transaction is ended. "1"enable. "0"disable.

SPI_FCR

Address: Operational Base + offset(0x08)

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.
7:2	-	-	Reserved.
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(0x0C)
This register is used to control SPI master controller transaction flow.

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11	W	0x0	SPI master controller soft reset bit (SRST_N). Writing a '1' to this bit will reset the SPI master controller logic.
10	RW	0x0	SPI master enable (SPIEN). This bit enables the SPI master controller. "1" enables the SPI master "0" disables the SPI master
9	RW	0x0	SPI run bit (SPIRUN). When the CPU set this bit from "0" to "1", the SPI master controller begins to transfer the data stored in the transmit FIFO and/or receive the data into receive FIFO on the SPI master 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). 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.
7:6	-	-	Reserved.
5	RW	0x0	Transmit and receive simultaneously transfer enable (TxRxsten). This bit is 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 controller clock polarity bit (CPOL)(1). 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)(1). 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	RW	0x0	Bidirectional mode enable (BIDIREN). This bit is used to configure the SPI master controller operation is in bidirectional mode or unidirectional mode. "1" = Bidirectional mode. "0" = Unidirectional mode.
0	RW	0x0	Look back mode (LBKMD). This bit is to indicate the operation mode of the SPI master controller is in a normal operation mode or in a look back mode. '0' Normal operation mode. '1' Look back mode.

Notes:

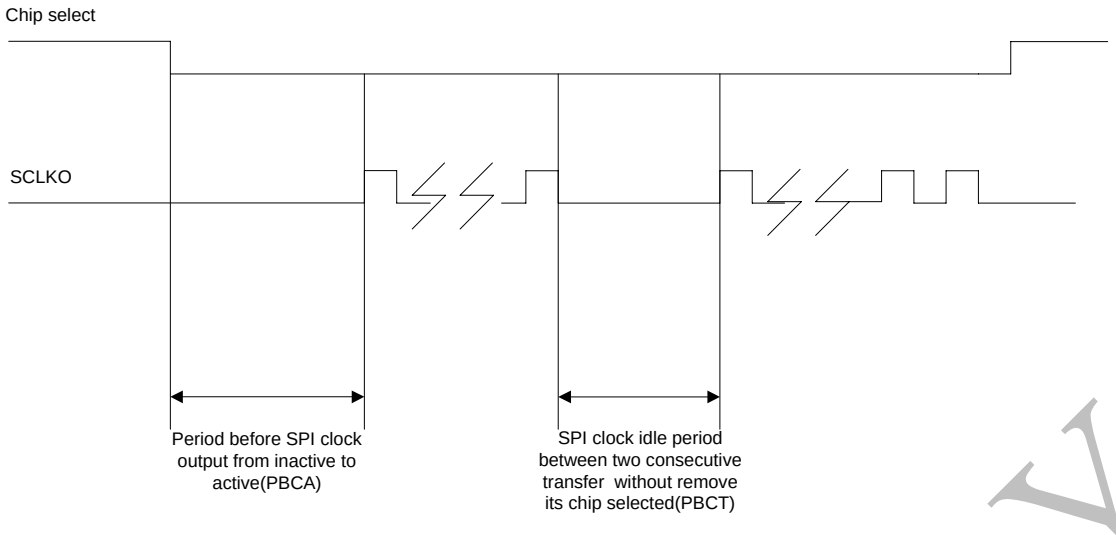
The soft reset signal would sustain **3 pclk cycles**, during the reset period any access to FIFO would be ignored.

SPI_DLYCR
Address: Operational Base + offset(0x10)

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



SPI_TxCR
Address: Operational Base + offset(0x14)
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(0x18)
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(0x1C)
SPI master 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 0x4: 5 bits 0x5: 6 bits 0x6: 7 bits 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). 0x0: Slave0 device (SS_N[0]) is set to an active state. 0x1: Slave0 device (SS_N[1]) is set to an active state. 0x2: Slave0 device (SS_N[2]) is set to an active state. 0x3: Slave0 device (SS_N[3]) is set to an active state. 0x4: Slave0 device (SS_N[4]) is set to an active state.

			0x5: Slave0 device (SS_N[5]) is set to an active state. 0x6: Slave0 device (SS_N[6]) is set to an active state. 0x7: Slave0 device (SS_N[7]) is set to an active state.
7:6	-	-	Reserved.
5:0	RW	0x0	SPI master controller clock divisor bits (SPIDIVR). 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(0x20)
SPI master controller interrupt status register.

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
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 TxFEIEN 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 RxFFIEN bit is enable. This bit will be cleared when receive FIFO pointer drops below trigger level
1	R	0x0	SPI master 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). 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.

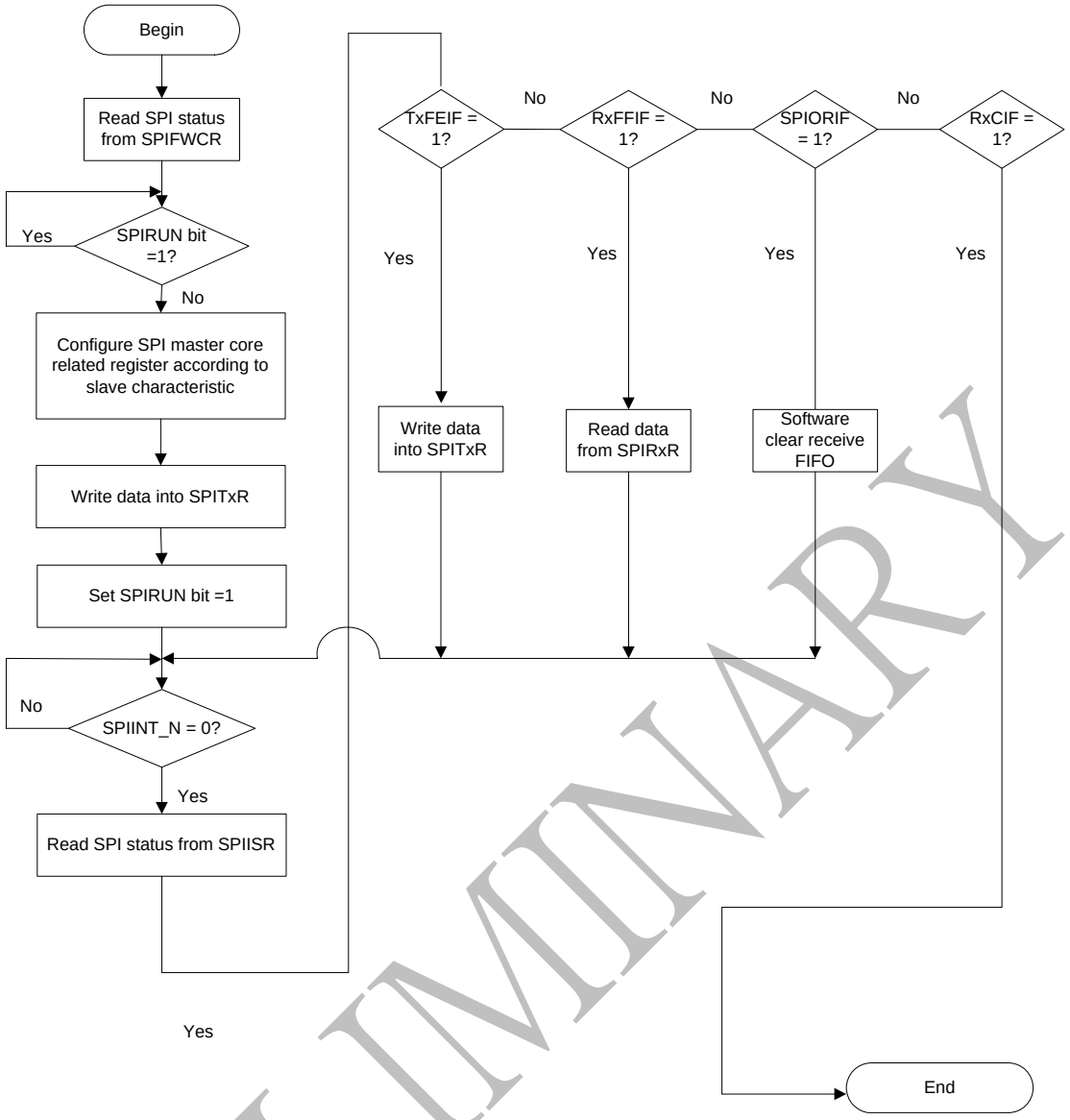
Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

18.4 Functional Description

18.4.1 Operation

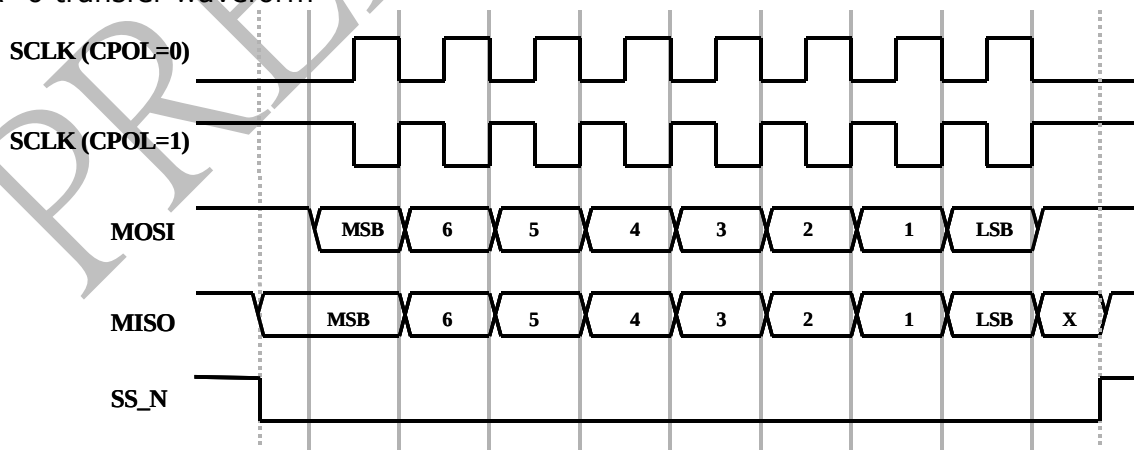
SPI master controller operation flow chart

The flow chart below is to describe how the software to configure and perform a SPI master controller transaction through this SPI master controller.

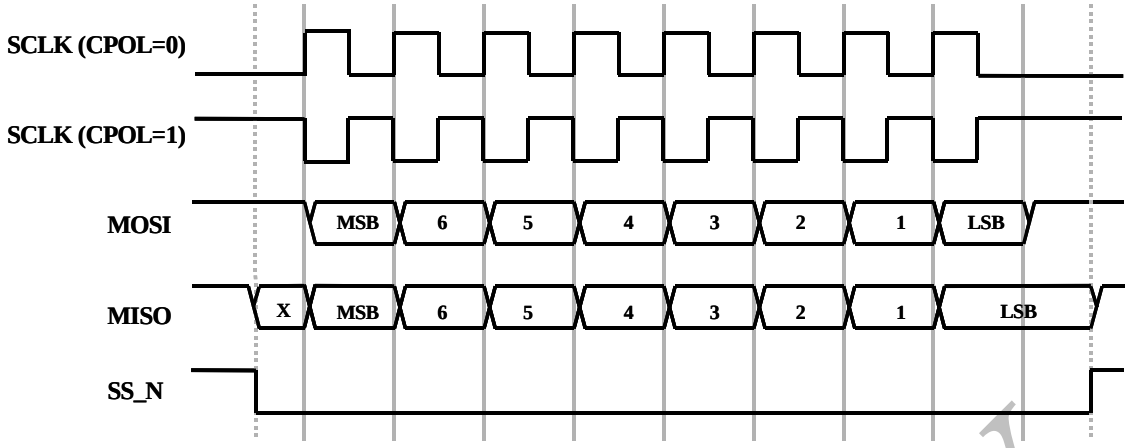


SPI master controller data transfer waveform

CPHA=0 transfer waveform



CPHA=1 transfer waveform



Chapter 19 I2C Controller

19.1 Design Overview

19.1.1 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.

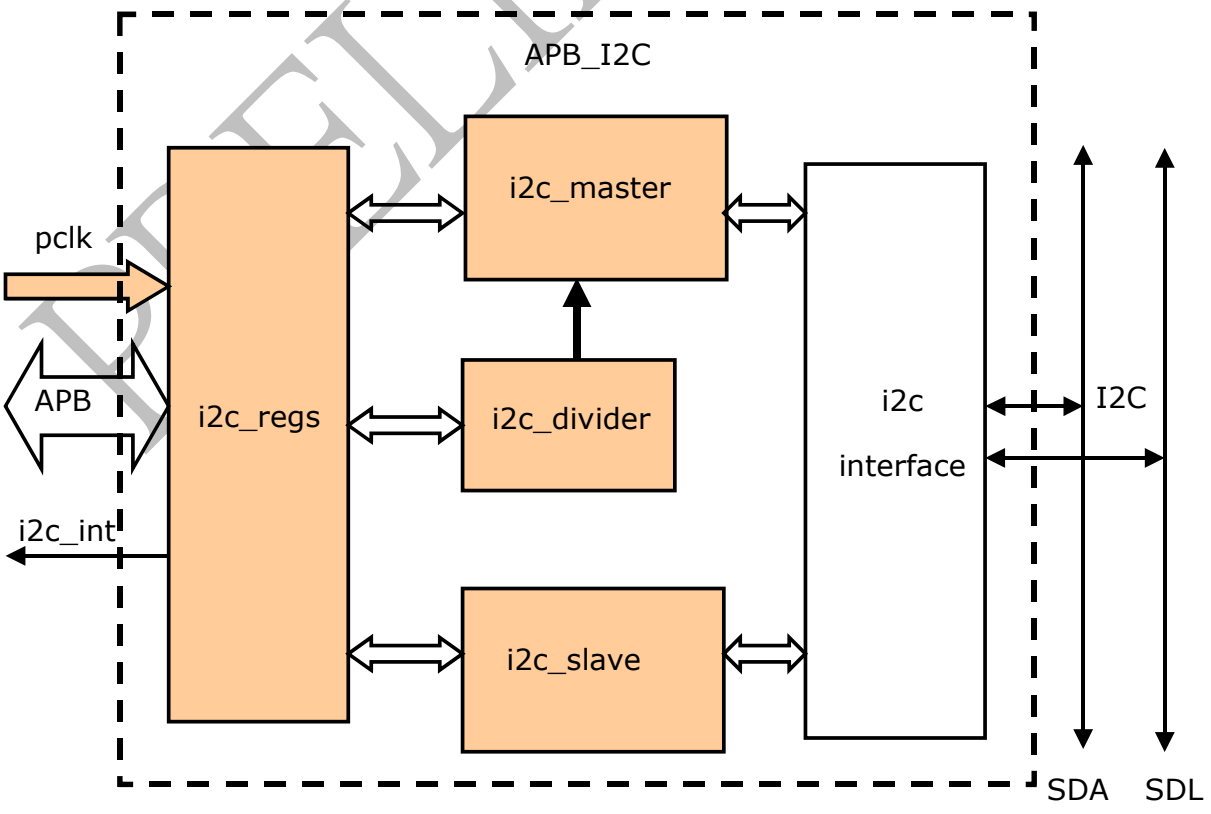
19.1.2 Features

- 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

19.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

19.2.1 Block Diagram



19.2.2 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.

19.3 Registers

This chapter describes the control/status registers of the design.
Software should read and write these registers using 32-bits accesses.

19.3.1 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

19.3.2 Detail Register Description

I2C_MTXR

Address: Operational Base + offset(0x00)
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(0x04)
This register contains data to be received on the I2C for master purpose.

bit	Attr	Reset Value	Description
31:8	-	-	Reserved

I2C_STXR
Address: Operational Base + offset(0x08)
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(0x0C)
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(0x10)
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(0x14)
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(0x18)
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(0x1C)
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(0x20)
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(0x24)
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(0x28)
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

19.4 Functional Description

19.4.1 Operation

I2C bus terminology

TERM	DESCRIPTION
Master	The device initiates/stops a transfer and generates SCL clock signals.
Slave	The device addressed by a master.
Transmitter	The device which sends data to SDA line.
Receiver	The device which receives data from SDA line.
Multi-master	More than one master can attempt to control the bus without corrupting the message.
Arbitration	If multi-master condition occurs, only one master is allowed to own the bus during this procedure.
Synchronization	Procedure to synchronize the clock signals of two or more devices.

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 determine by I2C_OPR register. The SCL frequency is calculated by the following formula

SCL Divisor = (I2CCDVR[5:3] + 1) x $2^{(I2CCDVR[2:0] + 1)}$

SCL = PCLK/ 5*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.

Salve 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 Salve Transmit Mode.

Salve 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.

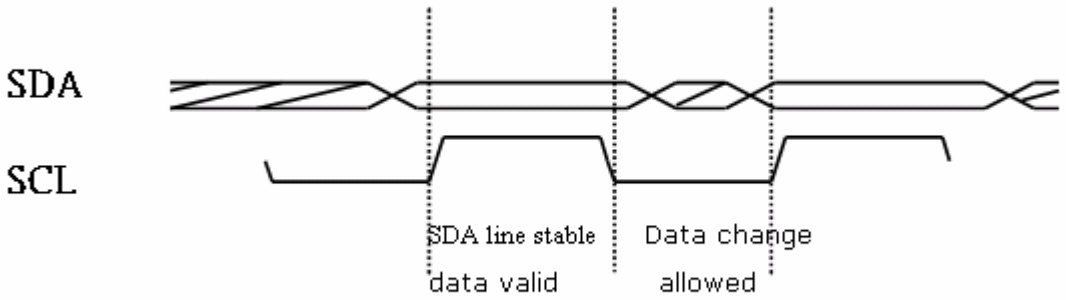
I2C controller data transfer waveform

- **Bit transferring**

(a) Data Validity

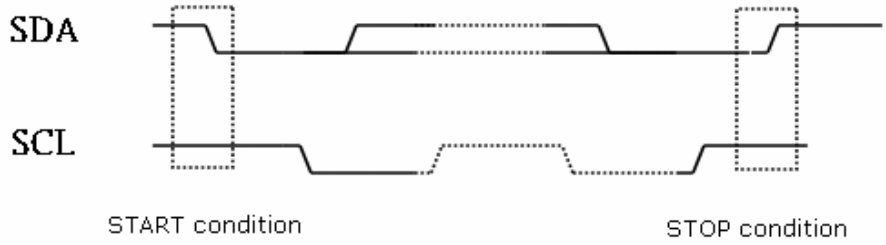
The SDA line must be stable during the high period of SCL, and the data on SDA line

can only be changed when SCL is in low state.



(b) START and STOP conditions

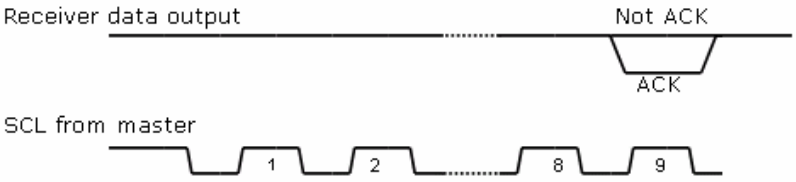
START condition occurs when SDA goes low while SCL is in high period. STOP condition is generated when SDA line goes high while SCL is in high state.



- **Data transfer**

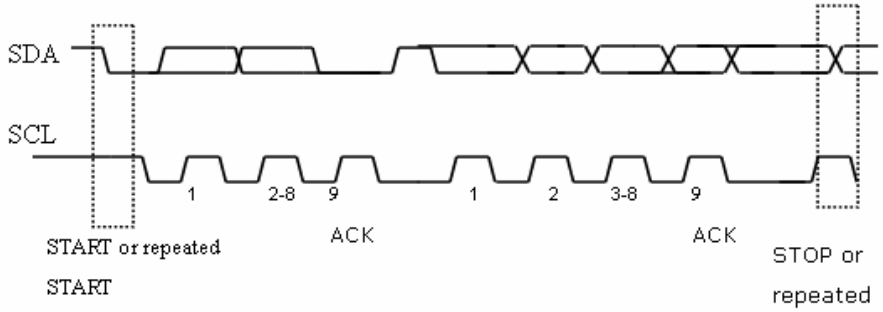
(a) Acknowledge

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".

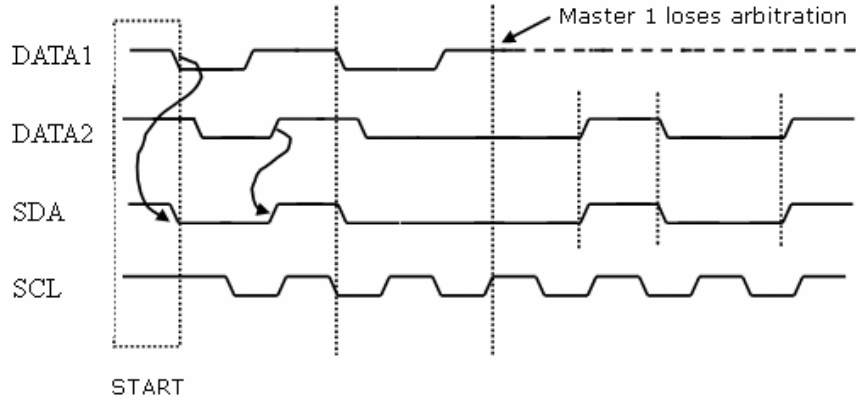


(b) Byte transfer

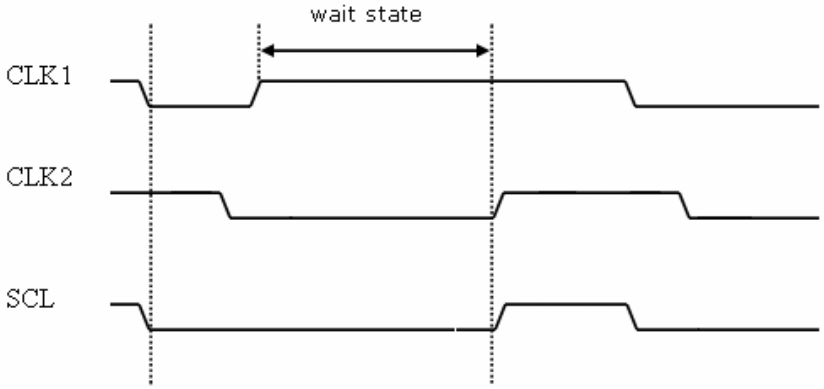
The master own I2C bus might initiate multi byte ot transfers to a slave, the transfers starts from a "START" command and ends in a "STOP"command. After every byte transfer, the receiver must reply an ACK to transmitter.



(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 connextion of I2C interface to the SCL line.

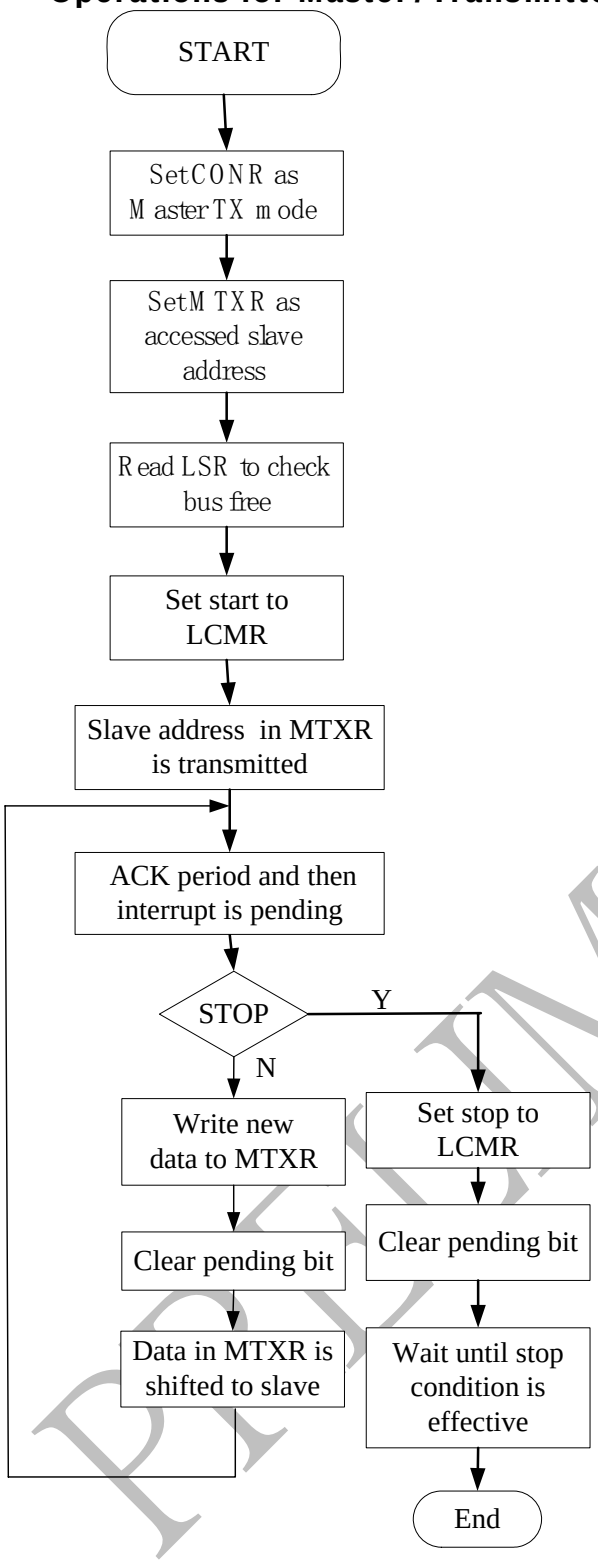


19.4.2 Programming sequence

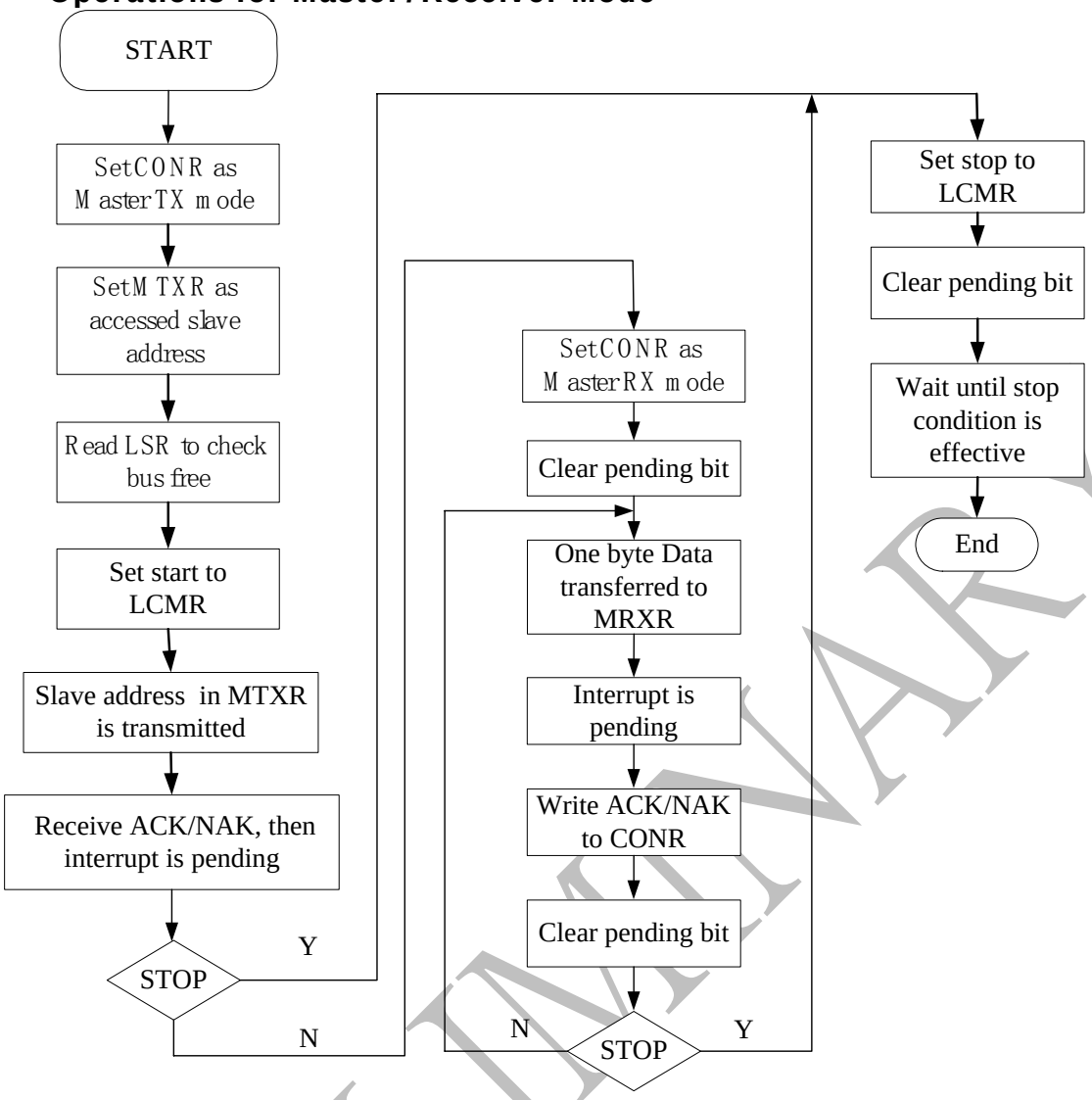
Control/Status Register programming sequence

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.

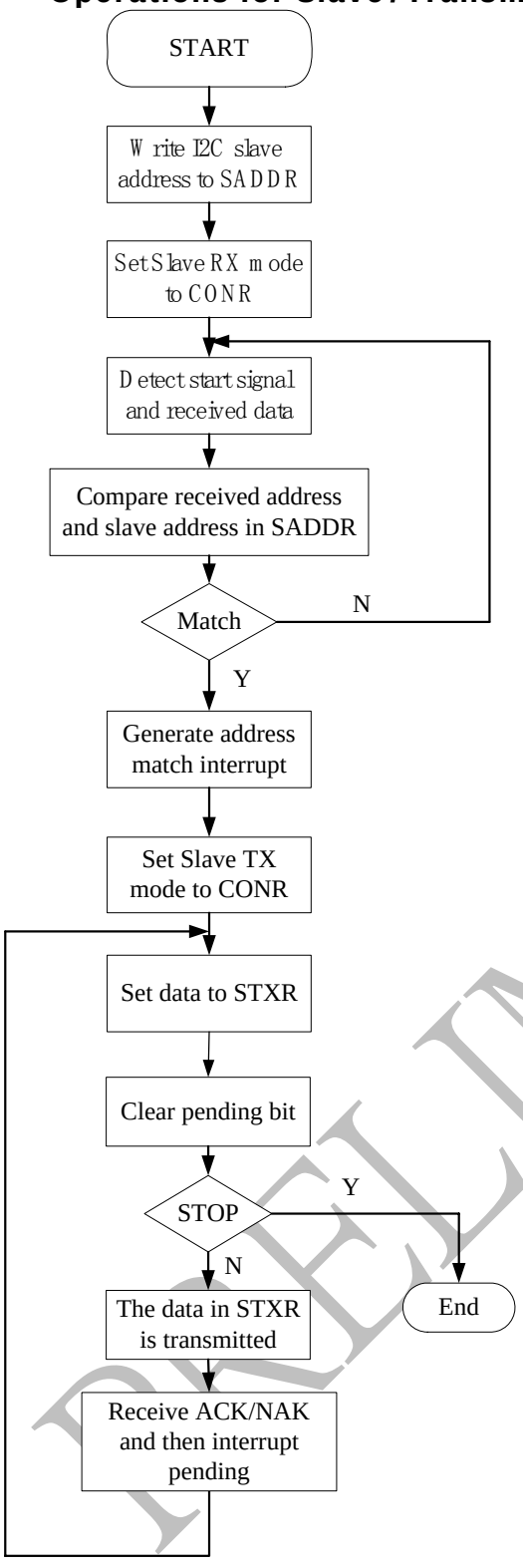
Operations for Master/Transmitter Mode



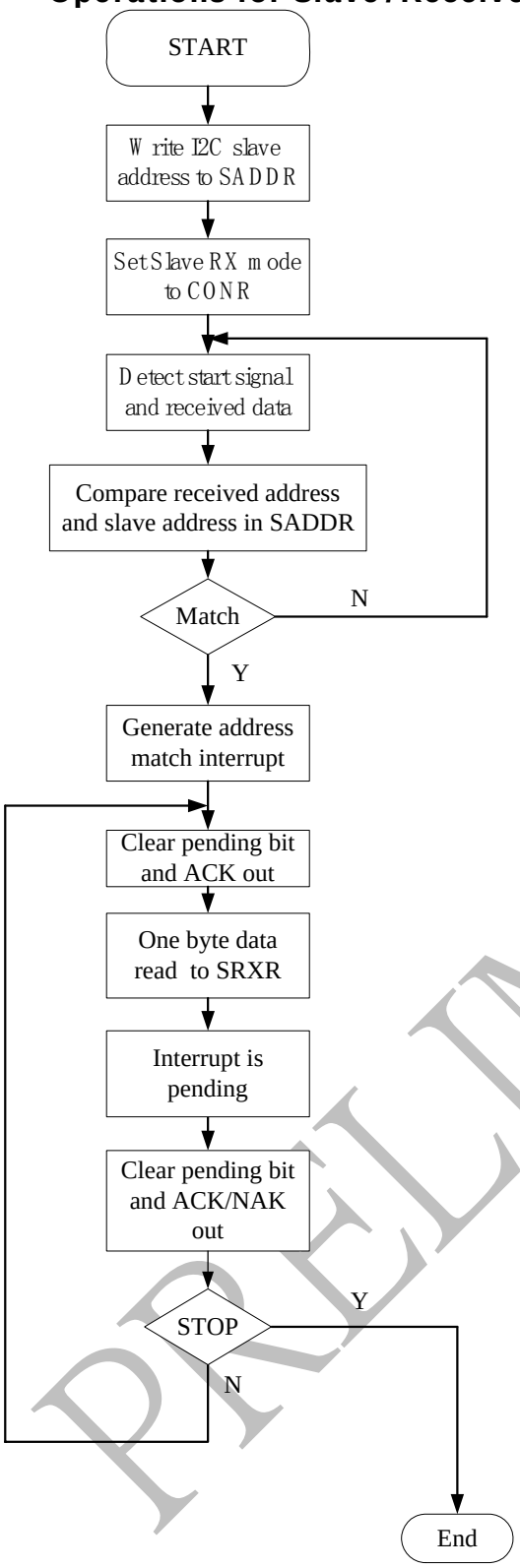
Operations for Master/Receiver Mode



Operations for Slave/Transmitter Mode



Operations for Slave/Receiver Mode



Chapter 20 I2S Controller

20.1 Design Overview

20.1.1 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 be 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.

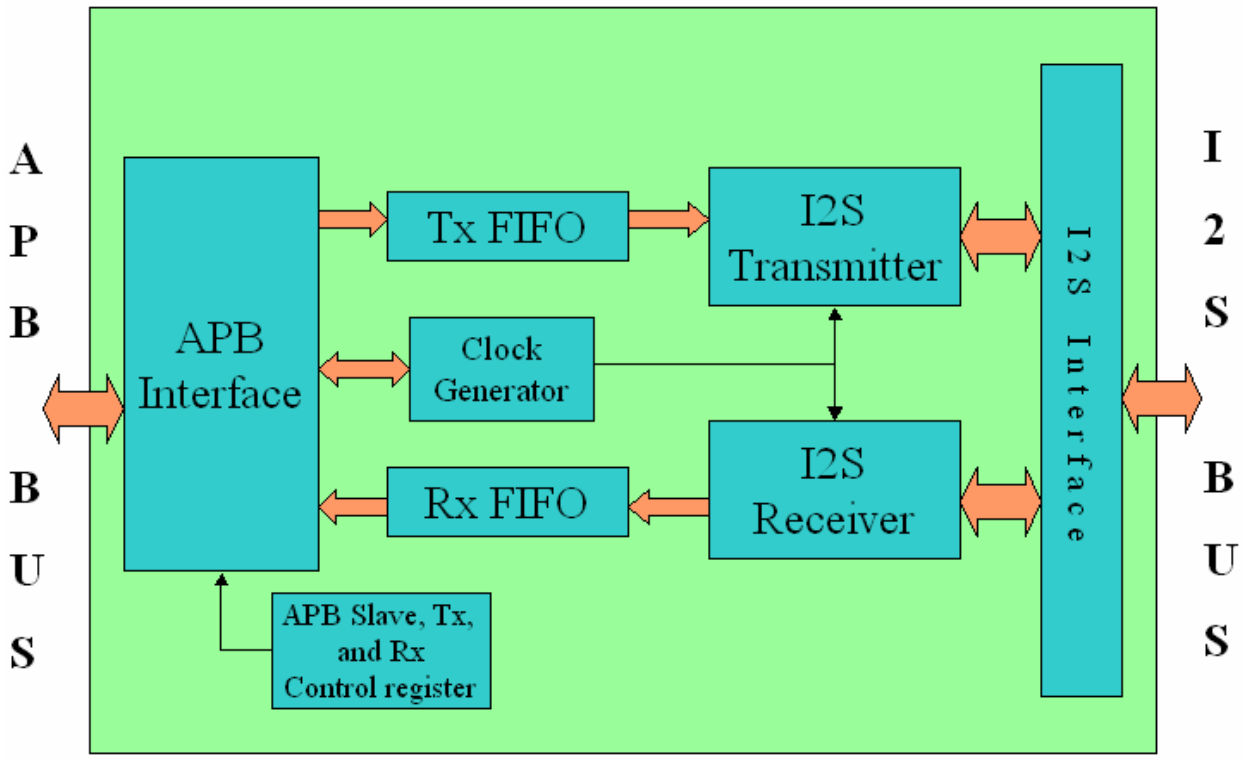
20.1.2 Features

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
Support Master and Slave mode function For Tx and Rx Transfer.

20.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

20.2.1 Block Diagram



20.2.2 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. The size of each FIFO can be programmable, which the default size is 32 bits x 32.

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.

20.3 Registers

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

20.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
I2S_OPR	0x0000	W	0x10000018	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.

2. Notes:

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

20.3.2 Detail Register Description

I2S_OPR

Address: Operational Base + offset(0x00)

This register contains I2S Controller's version and transmit/receive operation bit.

bit	Attr	Reset Value	Description
31:24	R	0x1	I2S version
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	W	0x0	HDMA REQ1 Disable 0 : Enable HDMA REQ1 1 : Disable (HDMA REQ1 Always 1)
5	W	0x0	HDMA REQ2 Disable 0 : Enable HDMA REQ2 1 : Disable (HDMA REQ2 Always 1)
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(0x04)

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(0x08)

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(0x0C)
This register controls the setting of the transmitter.

bit	Attr	Reset Value	Description
31:18	-	-	Reserved
17:16	RW	0x1	Oversampling rate select bits. 0x0 : 32fs 0x1 : 64fs 0x2 : 128fs 0x3 : reserved Oversampling rate = LRCK / SCLK
15:8	RW	0x8	Ratio bits. (MCLK / Ratio) = oversampling rate = SCLK frequency. This value is from 1 ~ 255. Default value is 8.
7:6	-	-	Reserved
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(0x10)
This register controls the setting of the receiver.

bit	Attr	Reset Value	Description
31:25	-	-	Reserved
24	W	0x0	Clear Rx FIFO bit. Writing a '1' to this bit clears the Receiver FIFO and resets its logic. But it doesn't clear the shift register, i.e. receiving of the current character continues.
23:18	-	-	Reserved
17:16	RW	0x1	Oversampling rate select bits. 0x0 : 32fs 0x1 : 64fs 0x2 : 128fs 0x3 : reserved Oversampling rate = LRCK / SCLK
15:8	RW	0x8	Ratio bits. (MCLK / Ratio) = oversampling rate = SCLK frequency. This value is from 1 ~ 255. Default value is 8.
7:6	-	-	Reserved
5:4	RW	0x1	Sample data resolution. Number of bits that are transmitted from each audio

			word. (20 and 24 bits is not available now) 0x0 : 8 bits 0x1 : 16 bits 0x2 : 20 bits 0x3 : 24 bits
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_FIFOSTS

Address: Operational Base + offset(0x14)

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 0x1 : half full 0x2 : almost full 0x3 : reserved
17:16	RW	0x1	Rx interrupt trigger level. 0x0 : almost empty 0x1 : half full 0x2 : almost 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(0x18)

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(0x1C)
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.

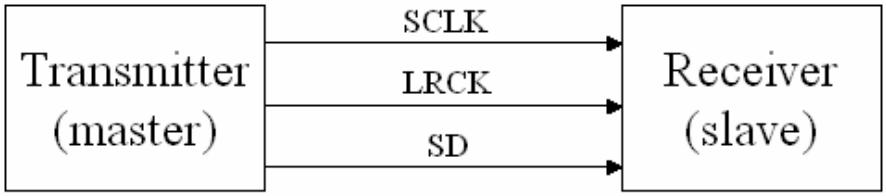
Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

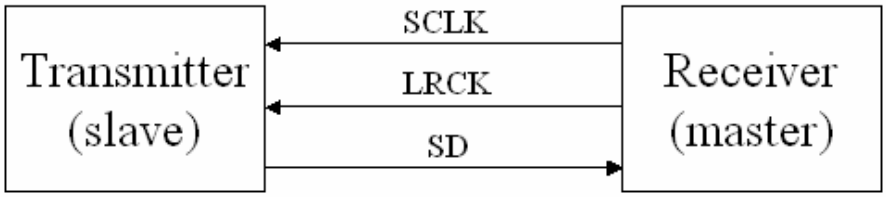
20.4 Functional Description

20.4.1 Operation

Digital audio serial data interface format

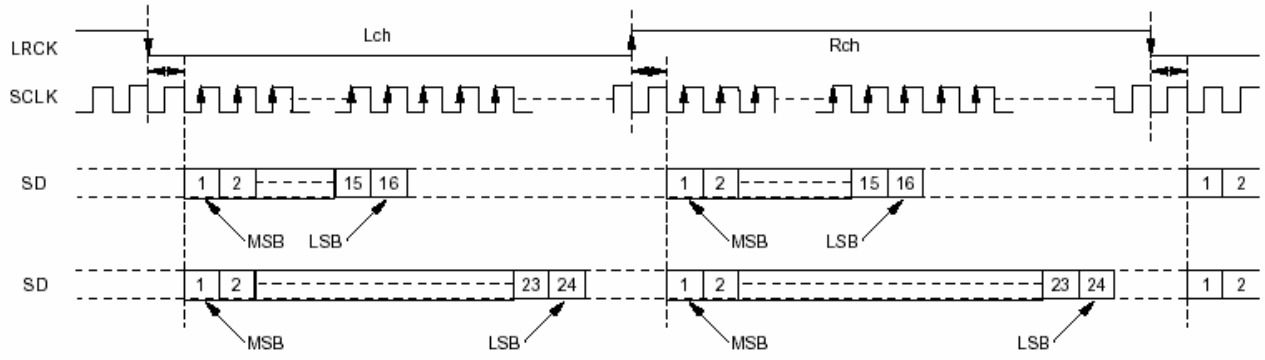
The I2S interface core supports three digital serial data interface formats for audio data transfer: I2S, Left-Justified, Right-justified. All these formats have SCLK, LRCK, and SD signals. The signal's direction is as below:





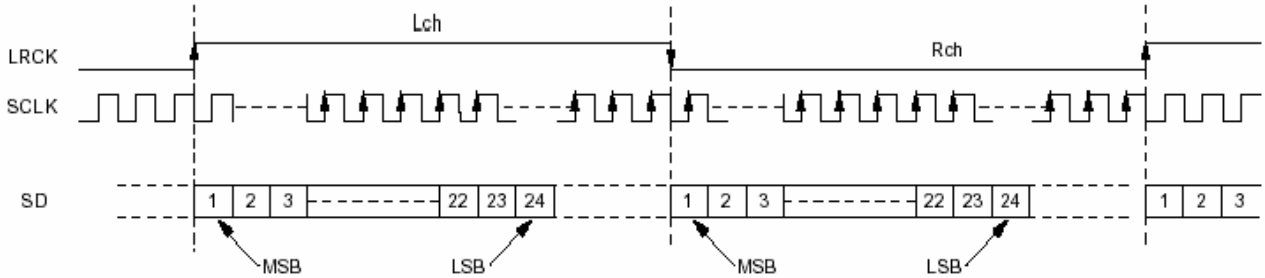
I2S Interface format

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.



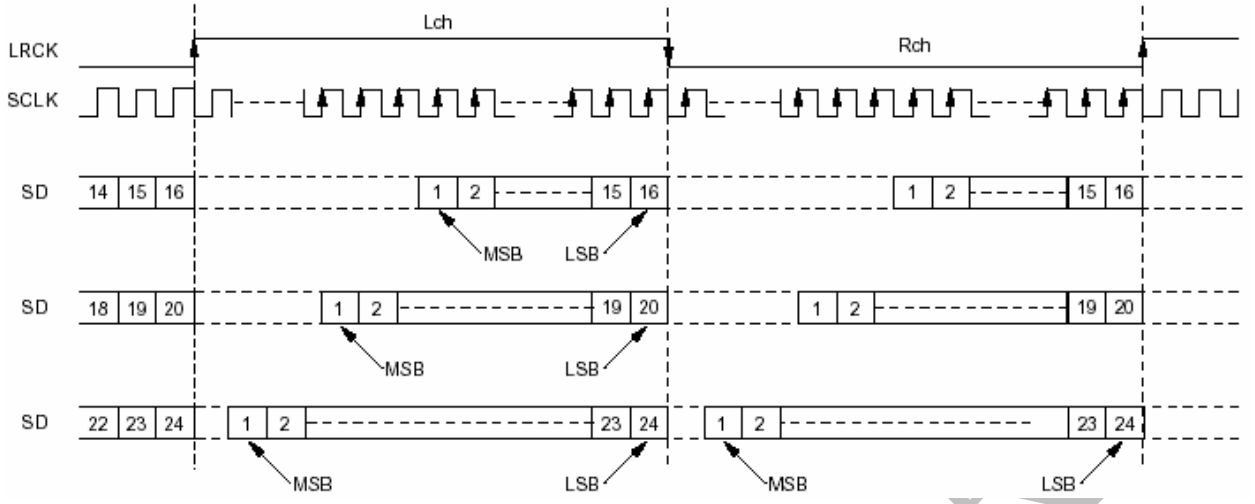
Left-Justified Interface format

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

Referring to Figure 6-1, in the I2S Controller, 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".

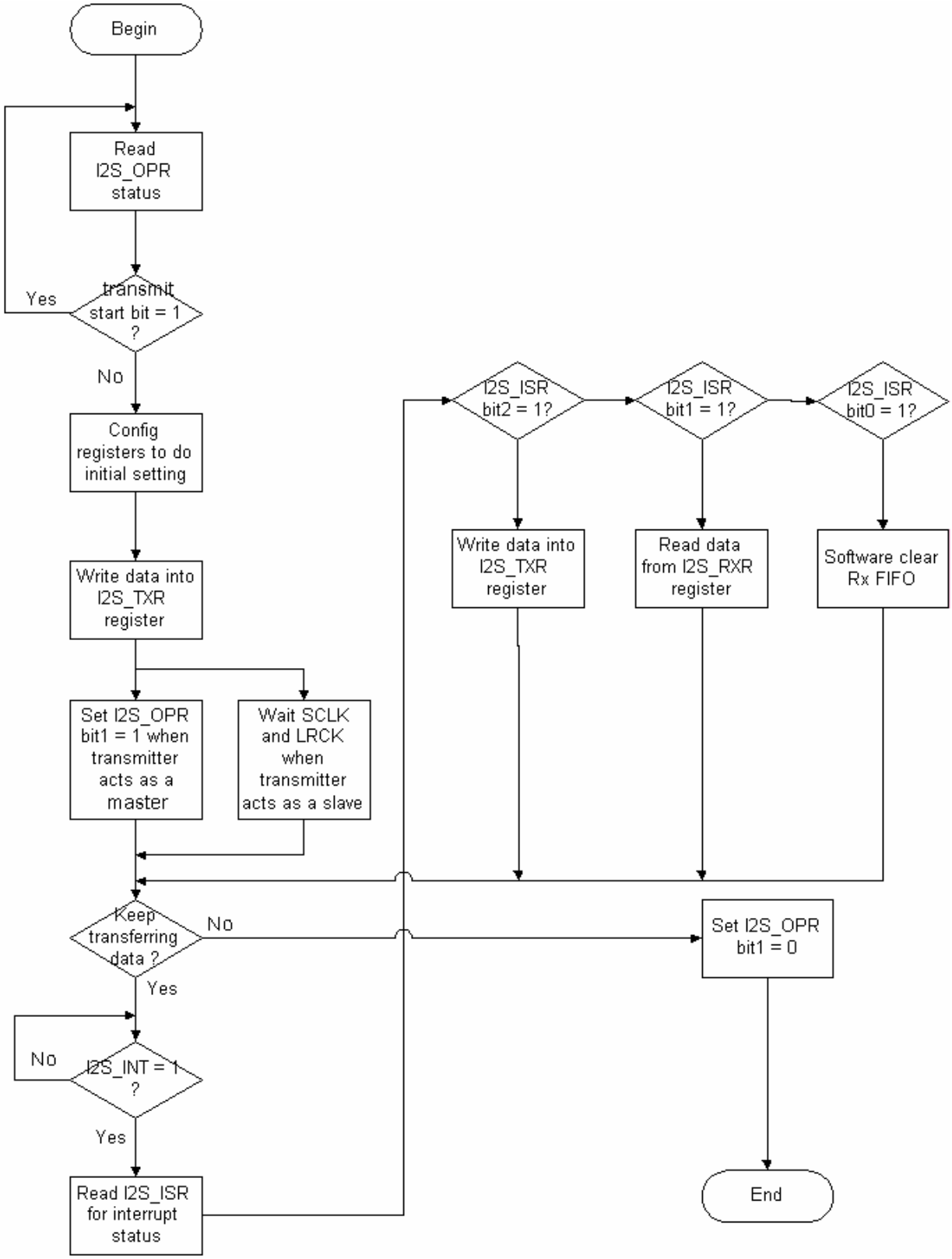
I2S Loop-back Test

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.

20.4.2 Programming sequence

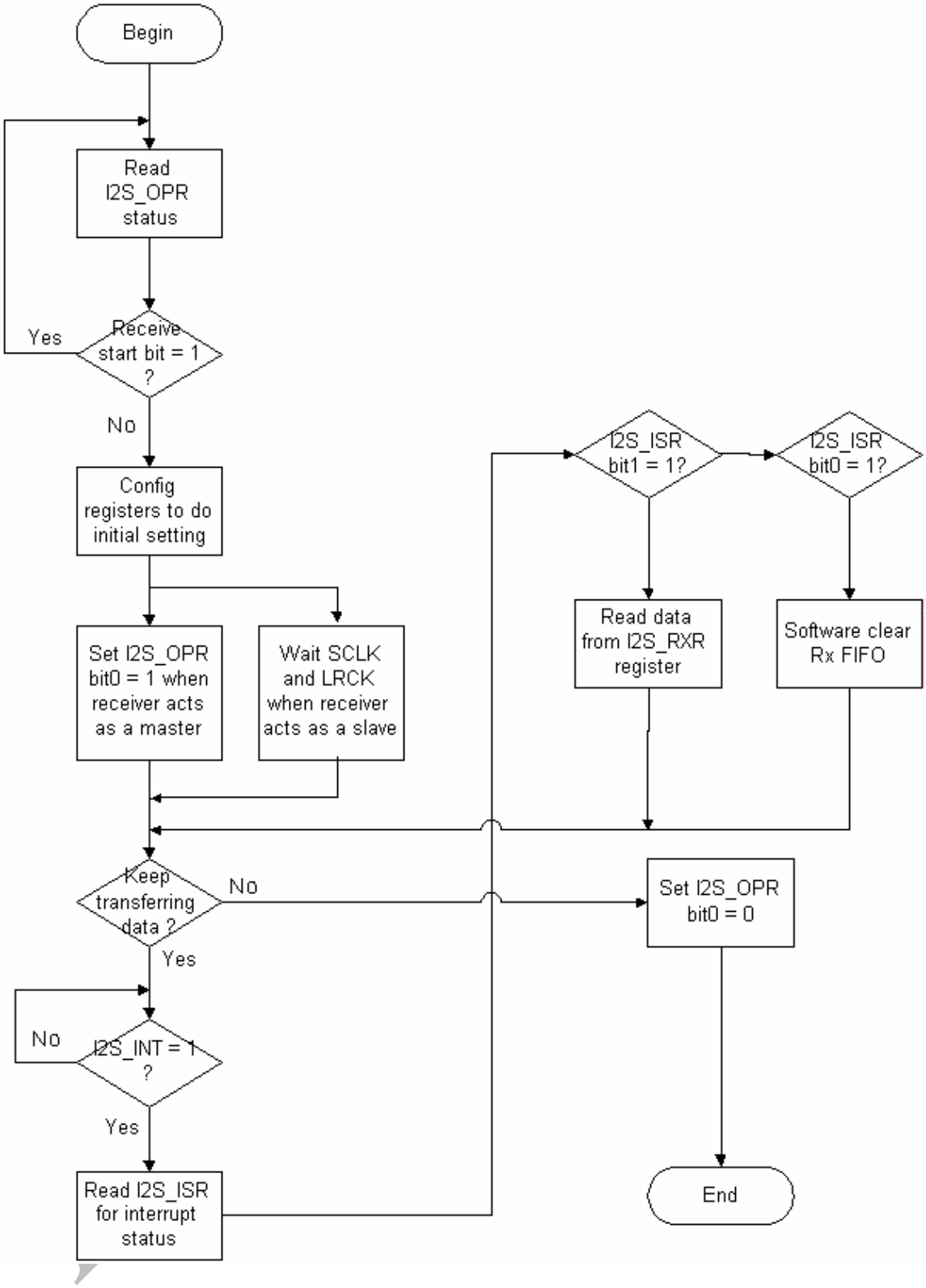
I2S Tx Operation Flow Chart

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.


Begin
Read I2S_OPR status
Receive start bit = 1 ?
Yes
No
Config registers to do initial setting
Set I2S_OPR bit0 = 1 when receiver acts as a master
Wait SCLK and LRCK when receiver acts as a slave
I2S_ISR bit1 = 1?
I2S_ISR bit0 = 1?
Read data from I2S_RXR register
Software clear Rx FIFO
Keep transferring data ?
No
Yes
Set I2S_OPR bit0 = 0
No
I2S_INT = 1 ?
Yes
Read I2S_ISR for interrupt status
End

Chapter 21 SD /MMC Host Controller

21.1 Design Overview

21.1.1 Overview

The SD 1.01/MMC 3.3 Host Controller(here after referred to as SD/MMC Host Controller) is an APB slave module that connects to the standard APB bus. The SD/MMC Host Controller is an interface bridge used between host microprocessor and SD/MMC memory card. There is a MMU structure with DMA interface for the SD/MMC Host Controller that can increase the data transfer rate between host microprocessor and SD/MMC memory card. It is designed to cover a wide area of applications such as PDA, DSC, Cellular phone, MP3 player, etc.

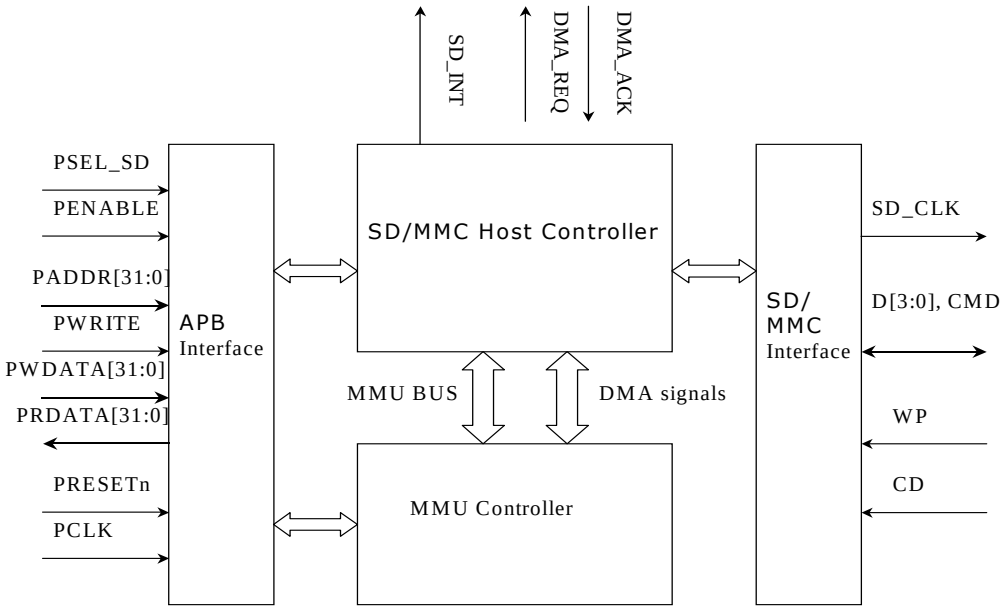
21.1.2 Features

- Compliant with the AMBA V2.0 APB interface
- Compliant with SD spec. Version 1.01 except SPI mode
- Compliant with MultiMediaCard system specification V3.3 except SPI mode, Stream R/W mode and I/O (Interrupt) mode
- Variable SD/MMC card clock rate 0 – 25 MHz which depends on APB clock frequency
- Controllable SD/MMC card clock to save power consumption
- Interrupt mechanism for card detection, command and response transfer and data transfer
- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Support MMU ping-pong structure to enhance the SD/MMC data transfer performance
- Support microprocessor and DMA data transfer with byte, half word and word access

21.2 Architecture

This chapter provides a description about the functions and behavior under various conditions.

21.2.1 Block Diagram



21.2.2 Block Descriptions

- APB Interface
  - The interface between AMBA APB and MMU, SD/MMC Host Controller
    1. SD/MMC Host Controller
  - Accept the command from host microprocessor and access to SD/MMC memory card
- MMU Controller
  - The MMU Controller is a swappable data buffer controlled by the host microprocessor and it can increase the data transfer rate between the host microprocessor and SD/MMC memory card
- SD/MMC Interface
  - Controlled by host microprocessor and generate signals to SD/MMC memory card

21.3 Registers

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

21.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
MMU_CTRL	0x0000	W	0x00000044	MMU control register
MMU_PNRI	0x0004	W	0x00000000	MMU pointer I setting register
CUR_PNRI	0x0008	W	0x00000000	MMU pointer I current register
MMU_PNRII	0x000C	W	0x00000000	MMU pointer II setting register
CUR_PNRII	0x0010	W	0x00000000	MMU pointer II current register

MMU_ADDR	0x0014	W	0x00000000	MMU address setting register
CUR_ADDR	0x0018	W	0x00000000	MMU address current register
MMU_DATA	0x001C	W	0x00000000	MMU data register
SD_CTRL	0x0020	W	0x00000FFF	SD/MMC Host control register
SD_INT	0x0024	W	0x00000000	SD/MMC Host interrupt register
SD_CARD	0x0028	W	0x00000007	SD/MMC card register
SD_CMDREST	0x0030	W	0x00000000	SD/MMC command and response transfer register
SD_CMDRES	0x0034	W	0x00000000	SD/MMC card command and response transfer status register
SD_DATAT	0x003C	W	0x00000000	SD/MMC data transfer register
SD_CMD	0x0040	W	0x00000000	SD/MMC command argument register
SD_RES3	0x0044	W	0x00000000	SD/MMC card response argument 3 register
SD_RES2	0x0048	W	0x00000000	SD/MMC card response argument 2 register
SD_RES1	0x004C	W	0x00000000	SD/MMC card response argument 1 register
SD_RES0	0x0050	W	0x00000000	SD/MMC card response argument 0 register

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

21.3.2 Detail Register Description

MMU_CTRL
Address: Operational Base + offset(0x00)
MMU control register

bit	Attr	Reset Value	Description
31:13	-	-	Reserved
12	RW	0x0	Endian control when CPU access to data buffer 1: Big endian access 0: Little endian access
11	RW	0x0	MMU DMA transfer signal 0: MMU DMA transfer end 1: MMU DMA transfer begin
10	RW	0x0	MMU DMA transfer direction 0: Read 1: Write
9	RW	0x0	MMU bus control swap data buffer 0: The MMU0 bus control data buffer I 1: The MMU0 bus control data buffer II
8	RW	0x0	CPU control swap data buffer 0: The host microprocessor control data buffer I 1: The host microprocessor control data buffer II
7	RW	0x0	Set 1 to reset data buffer II pointer and return to 0 automatically
6	R	0x1	Data buffer II pointer end signal

5:4	RW	0x0	Indicate data buffer II transfer width 0x0: Byte 0x 1: Half word 0x3: Word Note: 0x2 is not supported
3	RW	0x0	Set 1 to reset data buffer I pointer and return to 0 automatically
2	R	0x1	Data buffer I pointer end signal
1:0	RW	0x0	Indicate data buffer I transfer width 0x0: Byte 0x1: Half word 0x3: Word Note: 0x2 is not supported

MMU_PNRI
Address: Operational Base + offset(0x04)
MMU pointer I setting register

bit	Attr	Reset Value	Description
31:11	-	-	Reserved
10:0	RW	0x0	Set begin pointer for data buffer I transfer

CUR_PNRI
Address: Operational Base + offset(0x08)
MMU pointer I current register

bit	Attr	Reset Value	Description
31:11	-	-	Reserved
10:0	R	0x0	Read the current pointer for data buffer I transfer

MMU_PNRII
Address: Operational Base + offset(0x0C)
MMU pointer II setting register

bit	Attr	Reset Value	Description
31:11	-	-	Reserved
10:0	RW	0x0	Set begin pointer for data buffer II transfer

CUR_PNRII
Address: Operational Base + offset(0x10)
MMU pointer II current register

bit	Attr	Reset Value	Description
31:11	-	-	Reserved
10:0	R	0x0	Read the current pointer for data buffer II transfer

MMU_ADDR
Address: Operational Base + offset(0x14)
MMU address setting register

bit	Attr	Reset Value	Description
31:24	-	-	Reserved
23:0	RW	0x0	Set begin pointer for data address transfer

CUR_ADDR
Address: Operational Base + offset(0x18)
MMU address current register

bit	Attr	Reset Value	Description
31:24	-	-	Reserved
23:0	R	0x0	Read the current pointer for data address transfer

MMU_DATA
Address: Operational Base + offset(0x1C)
MMU data register

bit	Attr	Reset Value	Description
31:0	RW	0x0	MMU data register

SD_CTRL
Address: Operational Base + offset(0x20)
SD/MMC Host control register

bit	Attr	Reset Value	Description
31:14	-	-	Reserved
13	RW	0x0	Power control type for SD/MMC cards 0: The SD/MMC card power is controlled by CPU 1: The SD/MMC card power is controlled by CD
12	RW	0x0	Card detect type for SD cards 0: The card detect function is used by mechanism 1: The card detect function is used by CD/DAT3
11	RW	0x1	SD/MMC Card clock stop register 0: Run the SD/MMC Card clock 1: Stop the SD/MMC Card clock
10:0	RW	0x7FF	SD/MMC Card clock divider register

SD_INT
Address: Operational Base + offset(0x24)
SD/MMC Host interrupt register

bit	Attr	Reset Value	Description
31:7	-	-	Reserved
6	RW	0x0	Command and response transfer interrupt status 0: No 1: Yes
5	RW	0x0	Data transfer interrupt status 0: No 1: Yes
4	RW	0x0	Card detect interrupt status 0: No 1: Yes
3	-	-	Reserved
2	RW	0x0	Command and response transfer interrupt enable 0: disable 1: enable
1	RW	0x0	Data transfer interrupt enable 0: disable 1: enable
0	RW	0x0	Card detect interrupt enable 0: disable 1: enable

SD_CARD
Address: Operational Base + offset(0x28)
SD/MMC card register

bit	Attr	Reset Value	Description
31:7	-	-	Reserved
6	RW	0x0	Card select enable 0: No 1: Yes
5	RW	0x0	Card power control signal 0: disable 1: enable
4	RW	0x0	Card detect interrupt enable 0: disable 1: enable
3	-	-	Reserved

2	R	0x1	Card busy signal
1	R	0x1	Card write protect signal
0	R	0x1	Card detect signal

SD_CMDREST
Address: Operational Base + offset(0x30)
SD/MMC command and response transfer register

bit	Attr	Reset Value	Description
31:14	-	-	Reserved
13	RW	0x0	Command transfer signal 0: Command transfer end 1: Command transfer begin
12	RW	0x0	Response transfer signal 0: Response transfer end 1: Response transfer begin
11:9	RW	0x0	Response transfer type 0x0: R1 0x1: R1b 0x2:R2 0x3:R3 0x6:R6 Note: Other values are not supported
8	RW	0x0	Command and response transfer error status 0: No error 1: Error
7:6	-	-	Reserved
5:0	RW	0x0	SD/MMC Command Index

SD_CMDRES
Address: Operational Base + offset(0x34)
SD/MMC command and response transfer status register

bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	R	0x0	Card command transfer signal 0: Command transfer end 1: Command transfer begin
7	R	0x0	Card response transfer signal 0: Response transfer end 1: Response transfer begin
6	R	0x0	Card command and response error status 0: No error 1: Error
5	R	0x0	Card command and response bus conflict error 0: No error 1: Error
4	R	0x0	Card response timeout error 0: No error 1: Error
3	R	0x0	Card response transmission bit error 0: No error 1: Error
2	R	0x0	Card response index error 0: No error 1: Error
1	R	0x0	Card response CRC error 0: No error 1: Error
0	R	0x0	Card response end bit error 0: No error 1: Error

SD_DATAT
Address: Operational Base + offset(0x3C)

SD/MMC data transfer register

bit	Attr	Reset Value	Description
31:14	-	-	Reserved
13	RW	0x0	Data transfer signal 0: Data transfer end 1: Data transfer begin
12	RW	0x0	Data transfer direction 0: Read 1: Write
11	RW	0x0	Data transfer bus width 0: 1 line 1: 4 lines
10	RW	0x0	Data transfer with DMA interface 0: Disable 1: Enable
9	RW	0x0	Data transfer cycle 0: single or last 1: multiple
8	R	0x0	Data transfer error status 0: No error 1: Error
7	R	0x0	Data transfer bus conflict error 0: No error 1: Error
6	R	0x0	Data transfer timeout error 0: No error 1: Error
5	R	0x0	Data transfer CRC error 0: No error 1: Error
4	R	0x0	Read data transfer start bit error 0: No error 1: Error
3	R	0x0	Read data transfer end bit error 0: No error 1: Error
2:0	R	0x0	Write data transfer CRC status 0x2: No error 0x5: CRC error 0x7:No response Note: Other values are not supported

SD_CMD

Address: Operational Base + offset(0x40)

SD/MMC command argument register

bit	Attr	Reset Value	Description
31:0	RW	0x0	Command argument

SD_RES3

Address: Operational Base + offset(0x44)

SD/MMC card response argument-3 register

bit	Attr	Reset Value	Description
31:0	R	0x0	Card response argument-3

SD_RES2

Address: Operational Base + offset(0x48)

SD/MMC card response argument-2 register

bit	Attr	Reset Value	Description
31:0	R	0x0	Card response argument-2

SD_RES1

Address: Operational Base + offset(0x4C)

SD/MMC card response argument-1 register

bit	Attr	Reset Value	Description
31:0	R	0x0	Card response argument-1

SD_RES0
Address: Operational Base + offset(0x50)
SD/MMC card response argument-0 register

bit	Attr	Reset Value	Description
31:0	R	0x0	Card response argument-0

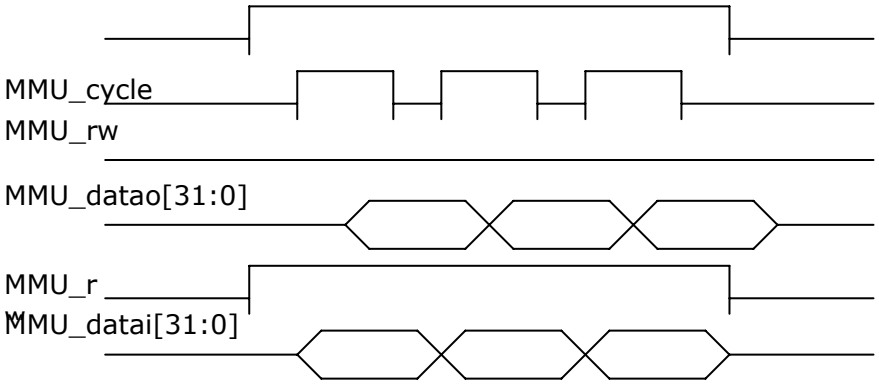
Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

21.4 Functional Description

21.4.1 Operation

MMU operation
The following sections will describe the operation of MMU:

- Interface reset
  All MMU registers are cleared during power on reset (PRESETn LOW) and then the MMU goes into normal operation.
- Normal operation
  After a power-on reset, the host microprocessor needs to set some registers in order to start MMU normal operation. The MMU operations can separate into two portions. One is the host microprocessor operation accessed from MMU_DATA register and the other is the application operation accessed from MMU bus. The operation steps are described as follows:
  1. Set MMU_PNRI register and MMU_PNRII register to indicate the access range for data buffer I and data buffer II.
  2. Set MMU_ADDR register to indicate the access range for DMA data transfer.
  3. Set MMU_swap bit to choose which data buffer the application wants to access.
  4. Set cpu_swap bit to choose which data buffer the host microprocessor wants to access.
  5. Set bufi_width bits and bufii_width bits to indicate the data buffer I and data buffer II access bus width.
  6. Set bufi_reset bit and bufii_reset bit to start the access data buffer operation.
  7. Set dma_dir bit to indicate the DMA data transfer is Read or Write.
  8. Set dma_enb bit to start the DMA data transfer.
  9. In the host microprocessor side, the host microprocessor accesses the data buffer from MMU_DATA register. In the application side, the application accesses the data buffer from MMU bus. The MMU bus operation timings are drawn as below:



SD/MMC Host Controller operation

The following sections will describe the operation of SD/MMC Host Controller:

- Interface reset

  All SD/MMC Host Controller registers are cleared during power on reset (PRESETn LOW) and then the SD/MMC Host Controller waits for SD/MMC memory card insertion, gives power to SD/MMC memory card and goes to normal operation.

- Normal operation

  After a power-on reset the SD/MMC Host Controller must initialize the SD/MMC memory cards by a special SD/MMC bus protocol. Each message is represented by one of the following transaction:

Command transaction:

A command is a token which starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transfer serially on the CMD line.

Response transaction:

A response is a token which is sent from an addressed card, or (synchronously) from all connected cards, to the host as an answer to a previously received command. A response is transfer serially on the CMD line.

Data transaction:

Data can be transferred from the card to the host or vice versa. Data is transferred via the data lines.

Data transfers to/from the SD/MMC Memory Card are done in blocks. Data blocks are always succeeded by CRC bits. Single and multiple block operations are defined. Note that the Multiple Block operation mode is better for faster write operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the host to use single or multiple data lines (as long as the card supports this feature).

Chapter 22 PWM Timer

22.1 Overview

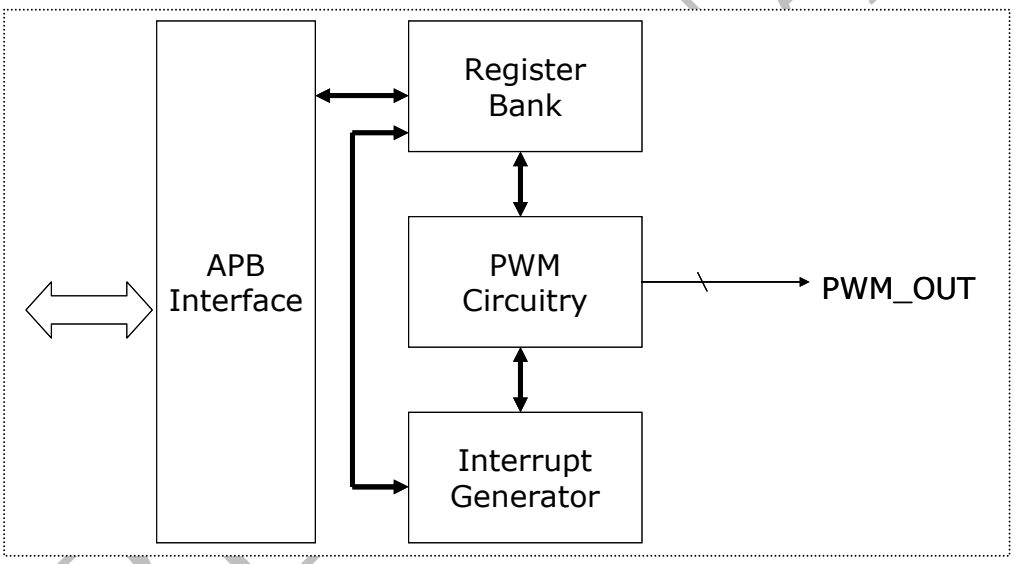
There are four PWM blocks in PWM Timer (PWM0, PWM1, PWM2 and PWM3). Each PWM block built-in 4-bit pre-scalar from PCLK. The PWM Timer supports both reference mode, which can output various duty-cycle waveforms, and capture, which can measure the duty-cycle of input waveform.

22.1.1 Key Features

- Programmable 4-bit pre-scalar
- 32-bit timer/counter facility
- Single-run or continues-run PWM mode
- Support maskable interrupt

22.2 Architecture

22.2.1 Block Diagram



22.2.2 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.

22.3 Registers

22.3.1 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	Current value 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	Current value 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	Current value 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	Current value register

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

22.3.2 Detail Register Description

PWMTn_CNTR (n=0~3)
Address: Operational Base + offset(0x00, 0x10, 0x20, 0x30)
PWM0 timer counter.

bit	Attr	Reset Value	Description
31:0	RW	0	Main PWM timer counter. Counting value ranges from 0 ~ (2^{32} –1).

PWMTn_HRC (n=0~3)
Address: Operational Base + offset(0x04, 0x14, 0x24, 0x34)
PWM0 HIGH reference or capture register

bit	Attr	Reset Value	Description
31:0	RW	0	PWM HIGH reference/capture registers

PWMTn_LRC (n=0~3)
Address: Operational Base + offset(0x08, 0x18, 0x28, 0x38)
PWM0 total reference or capture register

bit	Attr	Reset Value	Description
31:0	RW	0	PWM total reference/capture registers

PWMTn_CTRL (n=0~3)

Address: Operational Base + offset(0x0C, 0x1C, 0x2C, 0x3C)
This control register of PWM0 Timer.

bit	Attr	Reset Value	Description
31:13	-	-	Reserved.
12:9	R/W	0	Prescale factor. 0000: 1/2 0001: 1/4 0010: 1/8 0011: 1/16 0100: 1/32 0101: 1/64 0110: 1/128 0111: 1/256 1000: 1/512 1001: 1/1024 1010: 1/2048 1011: 1/4096 1100: 1/8192 1101: 1/16384 1110: 1/32768 1111: 1/65536
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 PWMTx_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

Chapter 23 ADC Controller

23.1 Overview

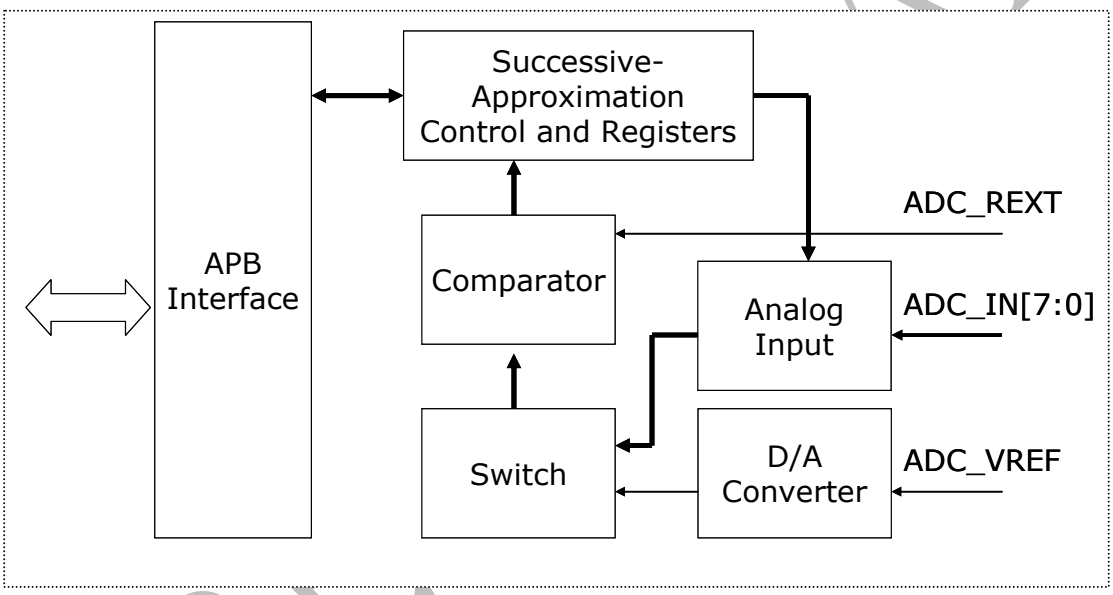
The ADC is an 4-channel signal-ended 10-bit Successive Approximation Register (SAR) A/D Converter. It converts the analog input signal into 10-bit binary digital codes at maximum conversion rate of 100KSPS with 1MHz A/D converter clock.

23.1.1 Key Features

- 4 input channels
- Maximum 10-bit resolution
- Maximum conversion rate of 100KSPS
- Channel 3 is tied to 1.2V

23.2 Architecture

23.2.1 Block Diagram



23.2.2 Block Descriptions

Successive-Approximate Register and Control Logic Block

This block is exploited to realize binary search algorithm, storing the intermediate result and generate control signal for analog block.

Comparator Block

This block compares the analog input ADC_CH[3:0] with the voltage generated from D/A Converter, and output the comparison result to SAR and Control Logic Block for binary search. Three level amplifiers are employed in this comparator to provide enough gain.

23.3 Registers

23.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
ADC_DATA	0x0000	W	0x00000000	ADC data registers
ADC_STAS	0x0004	W	0x00000000	ADC status register
ADC_CTRL	0x0008	W	0x00000000	ADC controller register

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

23.3.2 Detail Register Description

ADC_DATA
Address: Operational Base + offset(0x00)
This register contains the data after A/D Conversion.

bit	Attr	Reset Value	Description
31:10	-	-	Reserved.
9:0	R	0x00000000	A/D value of the last conversion.

ADC_STAS
Address: Operational Base + offset(0x04)
The status register of A/D Converter.

bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	R	0	ADC status. 0: ADC stop 1: Conversion in progress

ADC_CTRL
Address: Operational Base + offset(0x08)
The control register of A/D Converter.

bit	Attr	Reset Value	Description
31:7	-	-	Reserved.
6	RW	0	Interrupt status. This bit will be set to 1 when end-of-conversion. Set 0 to clear the interrupt.
5	RW	0	Interrupt enable. 0: Disable 1: Enable
4	RW	0	Start of Conversion(SOC) Set this bit to 1 to start an ADC conversion. This bit will reset to 0 by hardware when ADC conversion has started.
3	RW	0	ADC power down control bit 0: ADC power down 1: ADC power up and reset
2:0	RW	0	ADC input source selection. 000 : Input source 0 (ADC_CH[0]) 001 : Input source 1 (ADC_CH[1]) 010 : Input source 2 (ADC_CH[2]) 011 : Input source 3 (ADC_CH[3])

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

23.4 Function Description

A/D Conversion Sequence

The following is an example sequence of setting up A/D Converter, starting of conversion, and acquiring the result value.

- Power-up A/D Converter in ADC_CTRL[3]
- Select input channel of A/D Converter in ADC_CTRL[2:0] bit
- Set ADC start conversion in ADC_CTRL[4]
- Wait an A/D interrupt or poll the ADC_STAS register to determine when the conversion is completed
- Read the conversion result in the ADC_DATA register

Chapter 24 System Control Unit (SCU)

24.1 Design Overview

The SCU is an APB slave module that is designed for generating all of the internal and system clocks and resets of chip. Basically, SCU provides a memory remap function after boot for ARM7EJ, which makes the system memory space can be effectively utilized. SCU generates system clock from PLL output clock or external clock source , and generate system reset from external power-on-reset or watchdog timer reset. The SCU also provide power management mechanism for system power saving and general control bit.

Key Features
- Compliance to the AMBA APB interface
- Centralization of clock and reset sources control
- Memory space remap for ARM7EJ
- Five power management mode --- normal , slow , idle , stop, power off
- General peripheral control bit and chip status record

24.2 Registers

This section describes the control/status registers of the design.
- Registers Summary

Name	Offset	Size	Reset Value	Description
SCU_ID	0x0000	w	0xA1000604	Generic Core ID
SCU_REMAP	0x0004	w	0x0	Decoder remap control register
SCU_PLLCON1	0x0008	w	0x01850310	ARM PLL output freqrency control register (200MHz)
SCU_PLLCON2	0x000c	w	0x01850270	DSP PLL output freqrency control register (160MHz)
SCU_PLLCON3	0x0010	w	0x019807f8	CODEC PLL output freqrency control register (24.576MHz)
SCU_DIVCON1	0x0014	w	0x00700207	Clock divide frequency control register
SCU_CLKCFG	0x0018	w	0x0	Clock gating control register
SCU_RSTCFG	0x001c	w	0x00000010	Soft reset control register
SCU_PWM	0x0020	w	0x0bb80000	System Power Management control register
SCU_CPUPD	0x0024	w	0x0	ARM7 Power down control register
SCU_CHIPCFG	0x0028	w	0x00040005	System General Control register
SCU_STATUS	0x002c	w	0x0	SCU status register
SCU_IOMUXA_CON	0x0030	w	0x00040000	Chip I/O mux controller register A
SCU_IOMUXB_CON	0x0034	w	0x0	Chip I/O mux controller register B
SCU_GPIOUPCON	0x0038	w	0x0	Gpio PAD IO up or not controller register
SCU_DIVCON2	0x003c	w	0x0	Clock divide frequency control register

- **Detail Register Description**

SCU_ID
Address : Base Addr+0x00
Product ID register

bit	Attr	Reset Value	Description
31:0	R	0xA1000604	Chip Generic Core ID

SCU_REMAP
Address : Base Addr+0x04
Decoder remap control register

bit	Attr	Reset Value	Description
31:0	RW	0x0	Decoder remap function. Write value 0xdead_beef to this register to start remap function

SCU_PLLCON1
Address : Base Addr+0x08
ARM PLL configuration register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved
25	RW	0x0	ARM PLL test control 0 : normal 1 : test
24	RW	0x1	ARM PLL saturation behavior enable 0: disable 1 : enable
23	RW	0x1	ARM PLL Enables fast locking circuit 0 : disable 1 : enable
22	RW	0x0	ARM PLL Power down control 1: powerdown
21:16	RW	0x5	ARM PLL CLKR factor control
15:4	RW	0x31	ARM PLL CLKF factor control
3:1	RW	0x0	ARM PLL CLKOD factor control
0	RW	0x0	ARM PLL Bypass mode control 1: bypass

SCU_PLLCON2
Address : Base Addr+0x0c
DSP PLL configuration register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved
25	RW	0x0	DSP PLL test control 0 : normal 1 : test
24	RW	0x1	DSP PLL saturation behavior enable 0: disable 1 : enable
23	RW	0x1	DSP PLL Enables fast locking circuit 0 : disable 1 : enable
22	RW	0x0	DSP PLL Power down control
21:16	RW	0x5	DSP PLL CLKR factor control
15:4	RW	0x27	DSP PLL CLKF factor control
3:1	RW	0x0	DSP PLL CLKOD factor control
0	RW	0x0	DSP PLL Bypass mode control

SCU_PLLCON3
Address : Base Addr+0x10
CODEC PLL configuration register

bit	Attr	Reset Value	Description
31:26	-	-	Reserved
25	RW	0x0	CODEC PLL test control 0 : normal 1 : test
24	RW	0x1	CODEC PLL saturation behavior enable 0: disable 1 : enable
23	RW	0x1	CODEC PLL Enables fast locking circuit 0 : disable 1 : enable
22	RW	0x0	CODEC PLL Power down control
21:16	RW	0x18	CODEC PLL CLKR factor control
15:4	RW	0x7f	CODEC PLL CLKF factor control
3:1	RW	0x4	CODEC PLL CLKOD factor control
0	RW	0x0	CODEC PLL Bypass mode control

SCU_DIVCON1
Address : Base Addr+0x14
Internal clock configuration register

bit	Attr	Reset Value	Description
31	RW	0x0	Select USB PHY clk 0: 24MHz 1 : 12MHz
30:29	RW	0x0	Select Sensor CLK 00: select 24MHz 01 : select 48MHz 10 : select 27MHz
28	RW	0x0	Select LCDC CLK 0 : select divider output 1 : select 27MHz from external osc
27:20	RW	0x07	Control LCDC CLK divider frequency value lcdclk = pllclk/(lcdclk_div_sel+1)
19:18	RW	0x0	Control LCDC CLK divider frequency source 00 : select armpll_clk 01 : select DSPpll_clk 10 : select codecpll_clk
17:10	RW	0x0	Control LS_ADC CLK divider frequency ladcclk = pclk/(ladcclk_sel+1)
9	RW	0x1	Control CODECCLK work frequency 0 : select divider output from codec pll (default) 1 : select 12MHz from osc input
8:5	RW	0x0	Control CODECCLK divider frequency codecclk = codecpll_clk/(codecclk_sel+1)
4:3	RW	0x00	Control PCLK divider frequency 00 : HCLK: PCLK = 1:1 (default) 01 : HCLK: PCLK = 2:1 10 : HCLK: PCLK = 4:1
2	RW	0x1	clk ratio between armclk and hclk 1 : armclk: hclk=1:1 (default) 0 : armclk: hclk=2:1
1	RW	0x1	DSP subsys slow mode. When HIGH, enter into slow

			mode
0	RW	0x1	ARM subsys slow mode. When HIGH, enter into slow mode

SCU_CLKCFG

Address : Base Addr+0x18

Internal clock gating control register

bit	Attr	Reset Value	Description
31	RW	0x0	WDT pclk disable. When HIGH, disable clock
30	RW	0x0	Rtc pclk disable. When HIGH, disable clock
29	RW	0x0	PWM pclk disable. When HIGH, disable clock
28	RW	0x0	Timer pclk disable. When HIGH, disable clock
27	RW	0x0	Gpio pclk disable. When HIGH, disable clock
26	RW	0x0	HS_ADC clock disable. When HIGH, disable clock
25	RW	0x0	HS_ADC hclk disable. When HIGH, disable clock
24	RW	0x0	LS_ADC clock disable. When HIGH, disable clock
23	RW	0x0	LS_ADC pclk disable. When HIGH, disable clock
22	RW	0x0	SD/MMC clock disable. When HIGH, disable clock
21	RW	0x0	spi clock disable. When HIGH, disable clock
20	RW	0x0	i2c clock disable. When HIGH, disable clock
19	RW	0x0	uart1 clock disable. When HIGH, disable clock
18	RW	0x0	uart0 clock disable. When HIGH, disable clock
17	RW	0x0	i2s pclk clock disable. When HIGH, disable clock
16	RW	0x0	i2s clock disable. When HIGH, disable clock
15	RW	0x0	VIP clock disable. When HIGH, disable clock
14	RW	0x0	VIP hclk clock disable. When HIGH, disable clock
13	RW	0x0	LCDC clock disable. When HIGH, disable clock
12	RW	0x0	LCDC HCLK clock disable. When HIGH, disable clock
11	RW	0x0	embedded SRAM HCLK clock disable. When HIGH, disable clock
10	RW	0x0	a2a HCLK clock disable. When HIGH, disable clock
9	RW	0x0	NANDC HCLK clock disable. When HIGH, disable clock
8	RW	0x0	Reserved
7	RW	0x0	Reserved
6	RW	0x0	USB dev HCLK clock disable. When HIGH, disable clock
5	RW	0x0	USB host HCLK clock disable. When HIGH, disable clock
4	RW	0x0	DW DMA clock disable. When HIGH, disable clock
3	RW	0x0	HDMA clock disable. When HIGH, disable clock
2	RW	0x0	SDRAM HCLK clock disable. When HIGH, disable clock
1	RW	0x0	DSP clock disable. When HIGH, disable clock
0	RW	0x0	OTP clock disable. When HIGH, disable clock

SCU_RSTCFG

Address : Base Addr+0x1c

Internal soft reset control register

bit	Attr	Reset Value	Description
31:13	-	-	Reserved
12	RW	0x0	ARM7EJC soft reset
11	RW	0x0	Dual Core ECT soft reset
10	RW	0x0	Dual Core Mailbox soft reset
9	RW	0x0	SD/MMC soft reset request. When HIGH, reset relative logic

8	RW	0x0	HS_ADC soft reset request. When HIGH, reset relative logic
7	RW	0x0	LS_ADC soft reset request. When HIGH, reset relative logic
6	RW	0x0	CODEC soft reset request. When HIGH, reset relative logic
5	RW	0x0	DSP peripheral module soft reset request. When HIGH, reset relative logic
4	RW	0x1	DSP CORE soft reset request. When HIGH, reset relative logic
3	RW	0x0	Sensor soft reset request. When HIGH, reset relative logic
2	RW	0x0	LCDC soft reset request. When HIGH, reset relative logic
1	RW	0x0	USB DEV soft reset request. When HIGH, reset relative logic
0	RW	0x0	USB HOST soft reset request. When HIGH, reset relative logic

SCU_PWM

Address : Base Addr+0x20

System Power management register

bit	Attr	Reset Value	Description
31:16	RW	0x0bb8	pll lock period control
15:7	-	-	Reserved
6	RW	0x0	External wakeup pin type selection 0 positive polarity 1 negative polarity
5	RW	0x0	Disable RTC alarm wakeup stop mode 0 Enable RTC alarm wakeup 1 Disable RTC alarm wakeup
4	RW	0x0	Disable external wakeup stop mode 0 Enable external wakeup pin 1 Disable external wakeup pin
3	RW	0x0	SCU interrupt clear bit 0 pending 1 clear
2:0	RW	0x0	Power Management mode 000 : Normal mode 100 : stop mode Others : Reserved

SCU_CPUPD

Address : Base Addr+0x24

ARM power down control register

bit	Attr	Reset Value	Description
31:0	RW	0x0	If write " 0xdeed_babe" will power down ARM7 clock

SCU_CHIPCFG

Address : Base Addr+0x28

System configuration control register

bit	Attr	Reset Value	Description
31:20	RW	0x0	Reserved

19	RW	0x0	Nor flash size 0 : 16 bit 1 : 8bit
18	RW	0x1	Reserved
17	RW	0x0	DSP to ARM interrupt
16	RW	0x0	ARM to DSP interrupt
15:4	-	-	Reserved
3	RW	0x0	ARM7EJC high vector selection
2	RW	0x1	UHC databus16_8 0 : 8bit 1 : 16bit
1	RW	0x0	SEL_DOWNSTREAM 0 USB PHY for UDC 1 USB PHY for UHC
0	RW	0x1	Reserved

SCU_STATUS

Address : Base Addr+0x2c

Chip status register

bit	Attr	Reset Value	Description
31:5	R	0x0	Reserved
4	R	0x0	DSPSYSCLKVALID 0 system clock unstable 1 system clock stable
3	R	0x0	ARMSYSCLKVALID 0 system clock unstable 1 system clock stable
2	R	0x0	CODEC PLL clock locked indicator 0 unstable 1 stable
1	R	0x0	DSP PLL clock locked indicator 0 unstable 1 stable
0	R	0x0	ARM PLL clock locked indicator 0 unstable 1 stable

SCU_IOMUXA_CON

Address : Base Addr+0x30

Chip IO mux control register

bit	Attr	Reset Value	Description
31:20	-	-	Reserved
19	RW	0x0	i2s_codec_ext_sel 0 : i2s connected to internal codec 1 : i2s connected to pad
18	RW	0x1	i2c_codec_ext_sel 0 : i2c connected to internal codec 1 : i2c connected to pad
17:16	RW	0x0	I2c_flashcs3_gpiob_sel (I2C /FLASH/GPIO IO MUX control) 00: i2c sda 01: nand flash cs3 10: gpiob7
15:14	RW	0x0	I2c_flashcs2_gpiob_sel (I2C /FLASH/GPIO IO MUX control) 00: i2c scl 01: nand flash cs2 10: gpiob6

13:12	RW	0x0	gpiob_sd_spi_sel (GPIO /SD/SPI IO MUX control) 00: gpiob0~gpiob5 01: sd 10: spi
11	RW	0x0	gpio_lcdvsyn_sel (GPIO / LCD IO MUX control) 0: gpioa7 1: lcd vsync
10	RW	0x0	gpio_lcden_sel (GPIO / LCD IO MUX control) 0: gpioa6 1: lcd data enable
9	RW	0x0	gpio_flashcs1_sel (GPIO / FLASH IO MUX control) 0: gpioa5 1: nand flash cs1
8	RW	0x0	gpio_lcd22_sel (GPIO / LCD IO MUX control) 0: gpioa4 1: lcd data22
7:6	RW	0x0	gpioa_lcd20_nrts0_sel (GPIO /LCD/UART0 IO MUX control) 00: gpioa3 01: lcd data20 10: uart0 nrts
5:4	RW	0x0	gpioa_lcd18_ncts0_sel (GPIO /LCD/UART0 IO MUX control) 00: gpioa2 01: lcd data18 10: uart0 ncts
3:2	RW	0x0	gpioa_lcd17_txd0_sel (GPIO /LCD/UART0 IO MUX control) 00: gpioa1 01: lcd data17 10: uart0 txd
1:0	RW	0x0	gpioa_lcd16_rxd0_sel (GPIO /LCD/UART0 IO MUX control) 00: gpioa0 01: lcd data16 10: uart0 rxd

SCU_IOMUXB_CON
Address : Base Addr+0x34
Chip IO mux control register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved
22	RW	0x0	vip_hadc_sel (VIP / HADC mux control 0: vip 1: hadc
21	RW	0x0	gpiod_sdcke_sel (GPIO IO / SDRAM CKE MUX control) 0:gpiod3 1: SDRAM CKE
20	RW	0x0	gpiof_uhc_vbus_sel (GPIO IO / UHC VBUS MUX control) 0:gpiof4 1: UHC vbus
19	RW	0x0	gpiof_uhc_ocur_sel (GPIO IO / UHC OCUR MUX control) 0:gpiof3 1: UHC over current
18	RW	0x0	sdtaddr12_gpiof_sel (SDT/GPIO IO MUX control) 0: sdt addr12 1: gpiof2
17	RW	0x0	sdtaddr11_gpiof_sel (SDT/GPIO IO MUX control) 0: sdt addr11 1: gpiof1
16	RW	0x0	gpiof_vipclk_sel (GPIO /VIP IO MUX control) 0: gpiof0 1: VIP clk out
15	RW	0x0	gpioe_lcd_sel (GPIO /LCD IO MUX control) 0: gpioe0~7 1: lcd data8~15
14	RW	0x0	gpiod_pwm3_sel (GPIO /PWM IO MUX control) 0: gpiod7 1: pwm3
13	RW	0x0	gpiod_pwm2_sel (GPIO /PWM IO MUX control) 0: gpiod6 1: pwm2
12	RW	0x0	gpiod_pwm1_sel (GPIO /PWM IO MUX control) 0: gpiod5 1: pwm1
11	RW	0x0	gpiod_pwm0_sel (GPIO /PWM IO MUX control)

			0: gpiod4 1: pwm0
10	RW	0x0	gpiod_sdwpa_sel (GPIO/SD IO MUX control) 0: gpiod2 1:SD wpa
9:8	RW	0x0	gpiod_sdcda_rxd1_sel (GPIO/SD/UART1 IO MUX control) 00: gpiod1 01: SD cda 10: UART1 rxd
7:6	RW	0x0	gpiod_sdpca_txd1_sel (GPIO/SD/UART1 IO MUX control) 00: gpiod0 01: SD pca 10: UART1 txd
5	RW	0x0	gpioc_stcs1_sel (GPIO/SDT IO MUX control) 0: gpioc7 1: Static Mem cs1
4	RW	0x0	gpioc_i2scko_sel (GPIO/I2S IO MUX control) 0: gpioc6 1: i2s clk out
3	RW	0x0	gpioc_i2ssdo_sel (GPIO/I2S IO MUX control) 0: gpioc5 1: i2s sdo
2	RW	0x0	gpioc_i2ssdi_sel (GPIO/I2S IO MUX control) 0: gpioc4 1: i2s sdi
1	RW	0x0	gpioc_i2slrck_sel (GPIO/I2S IO MUX control) 0: gpioc3 1: i2s lrck
0	RW	0x0	gpioc_i2ssclk_sel (GPIO/I2S IO MUX control) 0: gpioc2 1: i2s sclk

SCU_GPIOUPCON
Address : Base Addr+0x38
Chip general IO pullup or pulldown control register

bit	Attr	Reset Value	Description
31:0	RW	0x0	GPIO A/B/C/D PAD IO up or not controller 0: UP 1 : NO

SCU_DIVCON2
Address : Base Addr+0x3c

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2:1	RW	0x0	Control HSADC CLK divider frequency 00 : osc : hsadcclk = 1:1 (default) 01 : osc : hsadcclk = 2:1 10 : osc : hsadcclk = 4:1
0	RW	0x0	HSADC select control 0 : select internal clock 1: select external clock

24.3 Application Notes

1. System remap

The system memory map will be remapped by programming SCU_REMAP register. Writing value with 0xdead_beef to this register cause the decoder change to new memory map from original one. Writing other value except 0xdeed_beef will back to original memory map.

2. PLL usage

- PLL output frequency configuration

The output frequency Fout is related to the reference frequency Fref by:

Fout = Fref * NF / NR / OD

Fout is clk output of PLL , and Fref is clk input of PLL from external oscillator (24MHz). Another , other factors such as NF, NR, OD can be configured by programming SCU_PLLCONx (x = 1~3) register , and their value will affect Fout as follows .

(1) CLKR: A 6-bit bus that selects the values 1-64 for the reference divider

NR = CLKR[5:0] + 1
Example:
/1 pgm 000000
/4 pgm 000011
/8 pgm 000111

(2) CLKF: A 12-bit bus that selects the values 1-4096 for the PLL multiplication factor
NF = CLKF[11:0] + 1
Example:
X1 pgm 000000000000
X2 pgm 000000000001
X4096 pgm 111111111111

(3) CLKOD: A 3-bit bus that selects the values 1-8 for the PLL post VCO divider
OD = CLKOD[2:0] + 1
Example:
/1 pgm 000
/4 pgm 011
/8 pgm 111

- PLL frequency range requirement

Fref/NR value range requirement : 300KHz - 600MHz
Fref/NR * NF value range requirement: 120MHz - 600MHz
If different CLKR and CLKF configuration value cause internal out of range, unpredicted result will be caused.

- PLL frequency change method

Before set some factors such NR/NF/OD to change PLL output frequency, you must change PLL from normal to bypass mode by programming SCU_PLLCONx register or change chip from normal to slow mode by programming SCU_DIVCON register. The later method is recommended. Then until PLL is in lock state by check SCU_STATUS register you can change PLL into normal mode, or after delay about 0.3ms.

- PLL powerdown

You can make PLL into or out of powerdown mode by programming SCU_PLLCONx register . After PLL will be out of powerdown mode , you can check SCU_STATUS register to confirm PLL in lock state.

3. Power management

The SCU provide five power management mode for system power saving and system can enter each power saving mode by setting appropriate control registers and programming sequence.

Mode	CPU	System IP	Peripheral IP	Power	Frequency
Normal	Run	Stop unused IP clock by software setting	Stop unused IP clock by software setting	On	200MHz/ 133MHz (ARM) 160MHz (DSP)
Slow	Run	Stop unused IP clock by software setting	Stop unused IP clock by software setting	On	24MHz Low speed
IDLE	Halt	Stop unused IP clock by software	Stop unused IP clock by software	On	Normal frequency or slow

		setting	setting		
Stop	Halt	Off	Off	On	Off
Power off	Off	Off	Off	RTC battery	Off

Normal mode :

In normal mode , CPU , system IPs and all peripheral IPs should works normally. The power consumption will be maximum when all IPs are turn on. Software allow to stop unused IP clock by programming SCU_CLKCFG register to reduce the power consumption.

Slow mode:

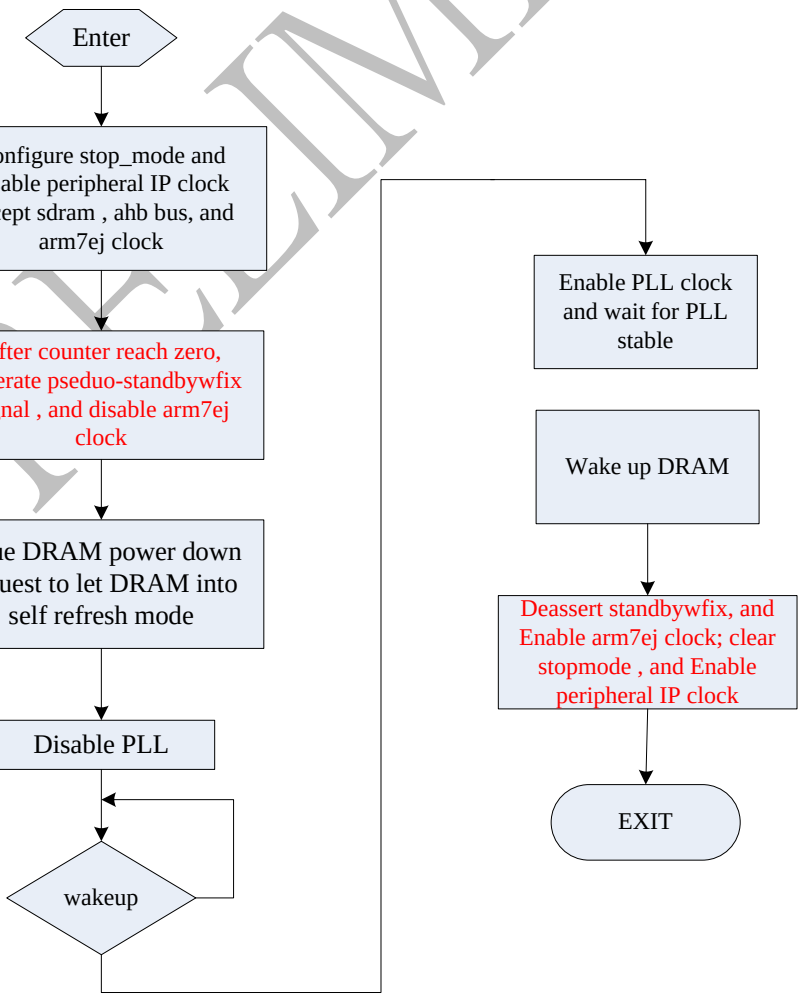
In SLOW mode, the system clock source is switching from high speed clock (PLL) to external lower speed clock source, and then power down PLL for further power saving. Enter by setting SCU_DIVCON register to select system clock source from PLL to the external OSC and set SCU_PLLCONx register to turn off PLL. Exit by turning on PLL and wait for PLL locked , switch system clock source back to PLL clock.

IDLE mode:

In IDLE mode , the CPU is expected to be idle and just wait for interrupts. In this case, software will make CPU to power down state and stop CPU clock by setting SCU_CPUPD register. The peripheral IP will keep running and wake up the CPU by external interrupt or external wakeup.

Stop mode:

In STOP mode, the operation of CPU and all IP should be halted. The clock of all IP is stop since PLL is power down. The system can release the stop mode from external wakeup pin or RTL alarm interrupt.



Power Off mode:

In power off mode , the system power is shut down. And the RTC is switch to battery power and keep running. The system can be power on again from RTC alarm or manually. For detail programming sequence please refer to RTC specification.

Programming Sequence:

- Normal mode:
  - Disable unused IP clock by setting SCU_CLKCFG register.
- Slow mode:
  - SCU_DIVCON[1:0] register to "11" to select clock source to low speed clock.
  - Turn off the PLL by setting SCU_PLLCONx[22]
  - Before switch to PLL clock , turn on PLL by setting SCU_PLLCONx[22] and read SCU_STATUS[0] to check PLL lock status.
- IDLE mode:
  - Set SCU_CPUPD to 0xdeed_babe
  - Idle mode will exit by external interrupt or wakeup pin
- STOP mode
  - Set SCU_PWM[2:0] to "100"
  - Set SCU_PWM[5:4] to select wakeup stop mode method
  - Set RTC alarm time in RTC control register if use RTC alarm to wakeup
  - Set SCU_PWM[6] to select ewakeup signal polarity if use external wakeup pin to wakeup
  - Disable all IP clock by setting SCU_CLKCFG except SDRAM clock
  - Set SCU_CPUPD to 0xdeed_babe to disable CPU clock and Power down PLL
  - Check the interrupt status when wakeup from stop mode and use SCU_PWM[3] to clear SCU_INT if system wakeup by external pin.

Chapter 25 LCD Controller

25.1 Design Overview

LCDC Unit is a controller interface to LCD panel, including a 2048X32bit SRAM.

Key features

- Compatible MCU and RGB interface ◦
- Compatible 8bits series and 24bits parallel RGB interface.
- Support delta RGB interface panel.
- Support YUV or RGB image source data input relatively in line.
- Support 8 grade alpha operations.
- Support even/odd field mode.
- Support 4:2:2 or 4:2:0 YUV source data input.
- Support 16bits or 24bits RGB source data input.
- Support scale base on YUV source.
- Support MCU buffer write and bypass mode.

25.2 Register

25.2.1 Register summary

symbol	description	Reset	R/W	offset	width
lcdc_ctrl	LCDC control register	0x00	RW	0x00	14
mcu_ctrl	Alpha buffer base address & Mcu mode control register	0x6600	RW	0x04	14
hor_period	define output horizontal total dots	0x	RW	0x08	11
vert_period/wait	define output vertical total lines; reuse in mcu mode define CSn/WEn/RDn signal timing		RW	0x0c	10
hor_pw	define output horizontal sync plus width		RW	0x10	5
vert_pw	define output vertical sync plus width		RW	0x14	5
hor_act	define output horizontal active region		RW	0x18	11
vert_act	define output vertical active region		RW	0x1c	10
hor_bp	define output horizontal back porch (dot)		RW	0x20	9
vert_bp	define output vertical back porch		RW	0x24	6
line0_yaddr	define line0 Y base address and line0 attribute	0x140	RW	0x28	15
line0_uvaddr	define line0 UV base address	0x141	RW	0x2c	11
line1_yaddr	define line1 Y base address and line0 attribute	0x280	RW	0x30	15
line1_uvaddr	define line1 UV base address	0x280	RW	0x34	11
line2_yaddr	define line2 Y base address and line0 attribute	0x3c0	RW	0x38	15
line2_uvaddr	define line2 UV base address	0x3c1	RW	0x3c	11
line3_yaddr	define line3 Y base address and line0 attribute	0x500	RW	0x40	15
line3_uvaddr	define line3 UV base address	0x501	RW	0x44	11
start_x	define scalar x start coordinate	0x00	RW	0x48	6
start_y	define scalar y start coordinate	0x00	RW	0x4c	10
delta_x	define scalar x step	0x200	RW	0x50	10
delta_y	define scalar y step	0x200	RW	0x54	10
intr_mask	Interrupt mask register	0x0	RW	0x58	3

alpha_alx	Alpha region A left x point	0x0	RW	0x5c	10
alpha_aty	Alpha region A top y point	0x0	RW	0x60	10
alpha_arx	Alpha region A right x point	0x13f	RW	0x64	10
alpha_aby	Alpha region A bottom y point	0xef	RW	0x68	10
alpha_blx	Alpha region B left x point	0x0	RW	0x6c	10
alpha_bty	Alpha region B top y point	0x0	RW	0x70	10
alpha_brx	Alpha region B right x point	0x13f	RW	0x74	10
alpha_bby	Alpha region B bottom y point	0xef	RW	0x78	10
lcdc_sta	Lcd controller status register	0x1000	R	0x7c	16
lcdc_buf	Lcd buffer adc	0x2000	R	0x7c	16

25.2.2 Detail Register Description

lcdc_ctrl	LCDC control register		0x00	RW	0x00	14

bits	**Name**	**direction**	**reset**	**description**
15-14	*reserved*			
13	*alpha _24b*	*r/w*	*0*	*Indicate rgb source in alpha buffer is 24bit or 16bit type*
12	*uvbufexch*	*r/w*	*0*	*Exchange the UV data buffer; this mode is useful when scalar enable and Y_mix mode not enable. This mode use in video decode optimize, otherwise keep it 0.*
11-9	*alpha*	*r/w*	*0*	*Select the alpha factor. Reference alpha description.*
8	*Y_mix*	*r/w*	*0*	*Set this bit when use the Y source data was mixed. This mode use in video decode optimize, otherwise keep it 0.*
7	*MCU*	*r/w*	*0*	*Select LCD interface as 8080 bus mode.*
6	*rgb24b*	*r/w*	*0*	*Indicate RGB source type. The RGB source data type is 24 bit,when this bit set 1, otherwise the RGB source data is 16 bit rgb565 format.*
5	*start_even*	*r/w*	*0*	*The first field is even field if this bit set 1, the first field is odd field in default.*
4	*even_en*	*r/w*	*0*	*Even/Odd field mode enable.*
3-2	*rgb_dummy*	*r/w*	*0*	*Indicate RGB interface as parallel or series mode. 0: if external LCD panel use a parallel interface. 1: reserve. 2: if external LCD panel use a series interface as UPS501. 3: if external LCD panel use a series interface as UPS502.*
1	*enable*	*r/w*	*0*	*0: disable LCDC RGB interface and reset all the state of LCDC. 1: enable LCDC when RGB interface mode select.*
0	*stop*	*r/w*	*0*	*Set 1 to stop LCDC at current frame complete when LCDC RGB interface mode is enabled.*

mcu_ctrl	Alpha buffer base address & Mcu mode control register	0x6600	RW	0x04	14

bits	**Name**	**direction**	**reset**	**description**
15	*reserve*			
14-8	*alpha_base*	*r/w*	*0x66*	*Set the alpha buffer start address in the 2048x32 ram. The address used as a word (32bit) address. The address was supposed align in 16word, and alpha buffer may*

				*only used the top 1024 word, so the bit14 was force as 1, the real address was calculated as alpha_base *16.*
7	*reserved*			
6	*alp_buf_en*	*r/w*	*0x0*	*The alpha buffer use as a circuit buffer. There was a pointer used to access the buffer when alpha process. 0: reset the buffer pointer to the alpha buffer start address. 1: enable the alpha buffer. This bit must set to 1 before enable alpha process. You can clear this bit at the end of very frame to sync the buffer pointer.*
5	*LCD_RS*	*r/w*	*0*	*Determine the pad LCD_RS state in MCU interface buffer write mode.*
4-2	*reserved*			
1	*buff_start*	*w*	*0*	*1: Start the write operation in MCU interface buffer write mode. This bit will clear automatic. 0: no effect.*
0	*bypass*	*r/w*	*1*	*1: MCU interface bypass mode, in this mode, a host write to the 2048x32 ram region will bypass to the external bus, usually the external bus connect a LCD panel with MCU interface mode. Usually use this mode to setup the LCD panel. 0: buffer write mode, in this mode, all host write to the 2048x32 ram region will buffered in the ram. You may set the buff_start bit to start the write operation. Usually use this mode to send out the image data.*

hor_period	define output horizontal total dclk number		0x	RW	0x08	11
bits	**Name**	**direction**	**reset**	**description**		
15-11	*reserve*					
10-0	*hor_period*	*r/w*	*0x666*	*Horizontal total dclk number*		

vert_period/wait	define output vertical total lines; reuse in mcu mode define CSn/WEn/RDn signal timing			RW	0x0c	10
bits	**Name**	**direction**	**reset**	**description**		
15-10	*reserve*					
9-0	*vert_period*	*r/w*	*0x666*	*output vertical total lines.*		
9-7	*csrw*	*r/w*		*This field specifies the number of processor clock cycles from the falling edge of CSN to the falling edge of RDN or WRN. If this bit is set to 0x0, the processor uses a csrw value of 0x1.*		
6-3	*rwpw*	*r/w*		*This field controls the width of RDN or WRN in processor clock cycles. This field must not set to 0.*		
2-0	*rwcs*	*r/w*		*This bit field specifies the number of processor clock cycles from the rising edge of RDN or WRN to the rising edge of CSN.*		

hor_pw	define output horizontal sync plus width			RW	0x10	5
bits	**Name**	**directio**	**res**	**description**		

		n	et	
15-5	reserve			
4-0	hor_pw	r/w	0x66	Horizontal sync plus width.

vert_pw	define output vertical sync plus width		RW	0x14	5
bits	**Name**	**direction**	**reset**	**description**	
15-5	reserve				
4-0	vert_pw	r/w	0x66	Vertical sync plus width.	

hor_act	define output horizontal active region		RW	0x18	11
bits	**Name**	**direction**	**reset**	**description**	
15-11	reserve				
10-0	hor_act	r/w	0x66	Horizontal active region.	

vert_act	define output vertical active region		RW	0x1c	10
bits	**Name**	**direction**	**reset**	**description**	
15-10	reserve				
9-0	vert_act	r/w	0x66	Vertical active region.	

hor_bp	define output horizontal back porch (dclk)		RW	0x20	9
bits	**Name**	**direction**	**reset**	**description**	
15-9	reserve				
8-0	hor_bp	r/w	0x66	Horizontal back porch.	

vert_bp	define output vertical back porch		RW	0x24	6
bits	**Name**	**direction**	**reset**	**description**	
15-9	reserve				
8-0	vert_bp	r/w	0x66	Vertical back porch.	

//

Those register define the output horizontal and vertical active region. The setting value will be the hor_bp or vert_bp setting value plus the real active value. Reference the example below.

Example:

horizontal sync width is 2 dot cycles, horizontal back porch needs 8 dot cycles, and horizontal active is 320 pixels.

Vertical sync width is 3 lines, vertical porch is 6 lines and vertical active is 240 lines.

The set the a hor_pw=2, vert_pw=3, hor_bp=10(2+8), vert_bp=9(3+6), hor_act=331(10+(320+1)*dot_per_pixel), vert_act=249(9+240).

dot_per_pixel =1 if external lcd panel uses a 24/18bits parallel RGB interfaces. In this case, lcdc_ctrl rgb_dummy will be set as 2'b00.

dot_per_pixel =3 if external lcd panel uses a 8bits series RGB interfaces without dummy cycles. In this case, lcdc_ctrl rgb_dummy will be set as 2'b10.

dot_per_pixel =3 if external lcd panel uses a 8bits series RGB interfaces with dummy cycles. In this case, lcdc_ctrl rgb_dummy will be set as 2'b11.

//

Detail refer the coding example.

line0_yaddr	define line0 Y base address and line0 attribute	0x140	RW	0x28	15
line1_yaddr	define line1 Y base address and line0 attribute	0x280	RW	0x30	15
line2_yaddr	define line2 Y base address and line0 attribute	0x3c0	RW	0x38	15
line3_yaddr	define line3 Y base address and line0 attribute	0x500	RW	0x40	15

bits	Name	direction	reset	description
15	*reserve*			
14	*Alpha_en*	*r/w*	*0*	*1: Alpha enable in current line.* *0: disable alpha.* *There are to method to condition alpha region. Reference alpha description.*
13	*Scale*	*r/w*	*0*	*1: enable scalar* *0: disable scalar.*
12	*gbr_order*	*r/w*	*0*	*0: rgb output as r8-g8-b8* *1: rgb output as g8-b8-r8* *This bit will use in delta LCD panel*
11	*YUV_scr*	*r/w*	*0*	*0: image source is RGB format.* *1: image source is YUV format.*
10:0	*Y_base*	*r/w*	*0*	*Y source data or RGB source data base address in the 2048x32 ram.*

line0_uvaddr	define line0 UV base address	0x141	RW	0x2c	11
line1_uvaddr	define line1 UV base address	0x280	RW	0x34	11
line2_uvaddr	define line2 UV base address	0x3c1	RW	0x3c	11
line3_uvaddr	define line3 UV base address	0x501	RW	0x44	11

bits	Name	direction	reset	description
15-11	*reserve*			
10-0	*UV_base*	*r/w*	*0*	*UV source data or RGB source data base address in the 2048x32 ram. If the source data is RGB format, UV_base address will set as Y_base+1.*

start_x	define scalar x start coordinate	0x00	RW	0x48	9

bits	Name	direction	reset	description
15-9	*reserve*			
8-0	*start_x*	*r/w*	*0*	*The 1/4 x offset of the first pixel in scale mode. This is means the first pixel after scale may between the 1fst and 2nd pixel of the raw. The low 4bits was forced to 0.*

start_y	define scalar y start coordinate	0x00	RW	0x4c	9

bits	Name	direction	reset	description
15-9	*reserve*			
8-0	*start_y*	*r/w*	*0*	*The y offset of the first pixel in scale mode. This is means the first pixel after scale may between the first line and 2nd line of the raw. 0*

				means the first line, 0x200 means the second line.

delta_x	define scalar x step			0x200	RW	0x50	10
bits	**Name**	**direction**	**reset**	**description**			
15-10	*reserve*						
9-0	*delta_x*	*r/w*	*0*	*The scale factor in x direction. 0x200 as factor equate 1.*			

delta_y	define scalar y step			0x200	RW	0x54	10
bits	**Name**	**direction**	**reset**	**description**			
15-10	*reserve*						
9-0	*delta_x*	*r/w*	*0*	*The scale factor in y direction. 0x200 as factor equate 1.*			

intr_mask	Interrupt mask register			0x0	RW	0x58	3
bits	**Name**	**direction**	**reset**	**description**			
15-4	*reserve*						
3	*req_mode*	*r/w*	*0*	*This bit set the DMA request mode. Lcd_Dma_req will active at the every line end if this bit set 1, otherwise the request signal only active at even line end.*			
2	*buff_intr*	*r/w*	*0*	*MCU mode buffer write interrupt mask bit. MCU mode buffer write interrupt signal active 1 at the end of buffer write process.*			
1	*vert_intr*	*r/w*	*0*	*Vertical interrupt mask bit. Vertical interrupt bit is the reverse of VSYNC signal.*			
0	*hor_intr*	*r/w*	*0*	*Horizontal interrupt mask bit. Horizontal interrupt line set 1 at the end of every valid display line active region, set 0 at the end of horizontal back porch.*			

alpha_alx	Alpha region A left x point			0x0	RW	0x5c	10
bits	**Name**	**direction**	**reset**	**description**			
15-10	*reserve*						
9-0	*alpha_alx*	*r/w*	*0x00*	*There are two rectangle specify the alpha region. Rectangle A and B.* *Set the alpha region A left x point.*			

alpha_aty	Alpha region A top y point			0x0	RW	0x60	10
bits	**Name**	**direction**	**reset**	**description**			
15-10	*reserve*						
9-0	*alpha_aty*	*r/w*	*0x00*	*There are two rectangle specify the alpha region. Rectangle A and B.* *Set the alpha region A top y point.*			

alpha_arx	Alpha region A right x point			0x0	RW	0x64	10
bits	**Name**	**direction**	**reset**	**description**			
15-10	*reserve*						
9-0	*alpha_arx*	*r/w*	*0x13f*	*There are two rectangle specify the alpha region. Rectangle A and B.* *Set the alpha region A right x point. 0x13f as 320.*			

alpha_aby	Alpha region A bottom y point			0x0	RW	0x68	10
bits	**Name**	**direction**	**reset**	**description**			

		n		
15-10	reserve			
9-0	alpha_aby	r/w	0xef	There are two rectangle specify the alpha region. Rectangle A and B. Set the alpha region A bottom y point. 0xef as 240.

alpha_blx	Alpha region B left x point		0x0	RW	0x6c	10
bits	**Name**	**direction**	**reset**	**description**		
15-10	reserve					
9-0	alpha_blx	r/w	0x00	There are two rectangle specify the alpha region. Rectangle A and B. Set the alpha region B left x point.		

alpha_aty	Alpha region B top y point		0x0	RW	0x70	10
bits	**Name**	**direction**	**reset**	**description**		
15-10	reserve					
9-0	alpha_bty	r/w	0x00	There are two rectangle specify the alpha region. Rectangle A and B. Set the alpha region B top y point.		

alpha_arx	Alpha region B right x point		0x0	RW	0x74	10
bits	**Name**	**direction**	**reset**	**description**		
15-10	reserve					
9-0	alpha_brx	r/w	0x13f	There are two rectangle specify the alpha region. Rectangle A and B. Set the alpha region B right x point. 0x13f as 320.		

alpha_aby	Alpha region B bottom y point		0x0	RW	0x78	10
bits	**Name**	**direction**	**reset**	**description**		
15-10	reserve					
9-0	alpha_bby	r/w	0xef	There are two rectangle specify the alpha region. Rectangle A and B. Set the alpha region B bottom y point. 0xef as 240.		

lcdc_sta	Lcd controller status register		0x1000	R	0x7c	16
bits	**Name**	**direction**	**reset**	**description**		
15-13	reserve					
12	mcu_idle	r/w	0x0	Mcu interface in idle state not in buffer write state.		
11	vactive	r/w	0x0	Set 1indicate the RGB interface in vertical active region.		
10	hactive	r/w	0x0	Set 1indicate the RGB interface in horizontal active region.		

25.3 LCD IO MUX

LCD PIN MUX				
PIN DEFINE	18bit MCU	18bit RGB	16bit MCU	8bit RGB
LCD_D0	DB1	B3	DB0	DB0
LCD_D1	DB2	B4	DB1	DB1
LCD_D2	DB3	B5	DB2	DB2
LCD_D3	DB4	B6	DB3	DB3
LCD_D4	DB5	B7	DB4	DB4
LCD_D5	DB6	G2	DB5	DB5
LCD_D6	DB7	G3	DB6	DB6
LCD_D7	DB8	G4	DB7	DB7
LCD_D8	DB9	G5	DB8	
LCD_D9	DB10	G6	DB9	
LCD_D10	DB11	G7	DB10	
LCD_D11	DB13	R3	DB11	
LCD_D12	DB14	R4	DB12	
LCD_D13	DB15	R5	DB13	
LCD_D14	DB16	R6	DB14	
LCD_D15	DB17	R7	DB15	
LCD_D16	DB0	B2		
LCD_D17	DB12	R2		
PA7/LCD_VSYNC/MCU_CS	CS	VSYNC	CS	VSYNC
LCD_HSYNC/ MCU_WR	WR	HSYNC	WR	HSYNC
LCD_CLK/MCU_RS	RS	DOT_CLK	RS	DOT_CLK

Chapter 26 VIP (Video Input Processor)

26.1 Design Overview

Key Features

- Support CMOS type image sensor interface
- Support CCIR656 interface
- Support CCIR-656 YCbCr 4:2:2 raster video input for 8bit mode in 525/60 NTSC and 625/50 PAL video system
- Data input clock is 27MHz for CCIR656 and 24MHz/48MHz for sensor
- Provide YUV 4:2:2/4:2:0 output
- Support up to (2048×1536) resolution, 3M pixel
- Support YUYV/UYVY format input

26.2 Registers

- 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	VIP Status register
VIP_CTRL	0x0010	w	0x00000000	VIP control register
VIP_CAPTURE_F1SA_Y	0x0014	w	0xFFFFFFFF	Capture data frame 1 start address for Y
VIP_CAPTURE_F1SA_UV	0x0018	w	0xFFFFFFFF	Capture data frame 1 start address for UV
VIP_CAPTURE_F1SA_Cr	0x001C	w	0xFFFFFFFF	Capture data frame 1 start address for Cr
VIP_CAPTURE_F2SA_Y	0x0020	w	0xFFFFFFFF	Capture data frame 2 start address for Y
VIP_CAPTURE_F2SA_UV	0x0024	w	0xFFFFFFFF	Capture data frame 2 start address for UV
VIP_CAPTURE_F2SA_Cr	0x0028	w	0xFFFFFFFF	Capture data frame 2 start address for Cr
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 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

- Detail Register Description

VIP_AHBR_CTRL

Address: Operational Base + offset (0x00)

AHB write control register

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2:0	RW	0x5	AHB Data Maximum Burst Length for Reading from H/W. 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. The following is the meaning. 001: INCR 101: INCR8 111: INCR16 Other: unused

VIP_AHBR_MASK
Address: Operational Base + offset (0x04)
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.(only for 656 mode) 0: Disable 1: Enable
0	RW	0x0	Capture complete interrupt enable 0: Disable 1: Enable

VIP_INT_STS
Address: Operational Base + offset (0x08)
Interrupt status register

Bit	Attr	Reset Value	Description
31:2	-	-	Reserved
2	R	0x0	Capture data line end happened interrupt (Read Clear, just a cycle pulse)(for debug)
1	R	0x0	Capture frame loss happened interrupt. (Read Clear, only for 656 mode) 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 (0x0C)
Status register

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

VIP_CTRL
Address: Operational Base + offset (0x10)
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	0x0	One frame stop enable 0: continuous mode or ping-pong mode 1: one frame complete stop
4	-	-	reserved
3	RW	0x0	Input data order, Y or UV first 0: UYVY 1: YUYV
2	RW	0x0	Sensor_or_656 0: 656 1: sensor
1	RW	0x0	Href_sentive 0: High active 1: Low active
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 (0x14)
Capture raw data frame 1 start address for Y

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Frame 1 Starting Address Register for Y

VIP_CAPTURE_F1SA_UV
Address: Operational Base + offset (0x18)
Capture raw data frame 1 start address for UV

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Frame 1 Starting Address Register for UV

VIP_CAPTURE_F1SA_Cr (No use)
Address: Operational Base + offset (0x1C)
Capture raw data frame 1 start address for Cr

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Frame 1 Starting Address Register for Cr

VIP_CAPTURE_F2SA_Y
Address: Operational Base + offset (0x20)
Capture raw data frame 1 start address for Y

Bit	Attr	Reset Value	Description

31:0	RW	0x0	Capture Frame 2 Starting Address Register for Y

VIP_CAPTURE_F2SA_UV
Address: Operational Base + offset (0x24)
Capture raw data frame 1 start address for UV

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Frame 2 Starting Address Register for UV

VIP_CAPTURE_F2SA_Cr (No use)
Address: Operational Base + offset (0x28)
Capture raw data frame 1 start address for Cr

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Capture Frame 2 Starting Address Register for Cr

VIP_FB_SR
Address: Operational Base + offset (0x2C)
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	0x0	Indicate the latest used Frame buffer number, for example, if Bit0 and Bit1 are both 1 and Bit3 is 1, means that Frame 2 is capture 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	1x0	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	1x0	Status bit to indicate current status of frame 1 (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.

VIP_FS
Address: Operational Base + offset (0x30)
Frame data size register

Bit	Attr	Reset Value	Description
31:16	RW	0x0	Pixel number per line width up to 2048
15: 0	RW	0x0	Line numbers per frame Height up to 1536

VIP_CROP

Address: Operational Base + offset (0x38)
Cropping start upper left point 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 (0x3C)
Y/CB/CR color modification

Bit	Attr	Reset Value	Description
31:27	-	-	Reserved.
26	RW	0x0	Y direction, 0-decrease 1-increase
25	RW	0x0	Cb direction, 0-decrease 1-increase.
24	RW	0x0	Cr direction, 0-decrease 1-increase
23:16	RW	0x0	Y value
15:8	RW	0x0	Cb value
7:0	RW	0x0	Cr value

VIP_RESET
Address: Operational Base + offset (0x40)
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 (0x44)
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

26.3 Application Description

This chapter is used to illustrate the operational behavior of how VIP module works. VIP module receive the ccir656 or sensor signal from external devices and translate it into YUV422/420 data, separate the data to Y and UV data, then store them to different Memory via AHB bus separately.

26.3.1 Hardware reset

When RESETN pin is set to low, it will cause everything including both Video Input Processor and AHB master modules to be reset to default state immediately.

26.3.2 Software reset

While configuring 0x40 register with 0x76543210, VIP will be software reset after 200 cycles of AHB.

26.3.3 Initial configuration

After HW/SW reset, Host must initially configure VIP by control registers via AHB bus. The configurations include external devices, frame data start address, frame size, output format….etc. VIP module can work in three modes: one frame stop mode、ping-pong mode、continuous mode.

26.3.4 One frame stop mode (Only frame 1 can be used in this mode)

Setup **VIP_CAPTURE_F1SA_<Y, UV>** as start memory of the continuous memory, configure register **0x10** to set working mode、output data format…etc, configure register **0x38** to set crop start point and configure register **0x30** to set you whole 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** (bit0) to start one frame stop mode. After one frame captured, VIP will automatic stop. After capturing, the image Y, UV data will be stored at main memory location defined by VIP_CAPTURE_F1SA_Y, VIP_CAPTURE_F1SA_U.V separately

26.3.5 Ping-Pong mode

Setup **VIP_CAPTURE_F1SA_<Y, UV>, and VIP_CAPTURE_F2SA_<Y, UV>**. configure register **0x10** to set working mode、output data format…etc, configure register **0x38** to set crop start point and configure register **0x30** to set you whole 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** (bit0) to start ping-pong mode. After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically, and host must assign new address pointer of frame1 and clear the frame1 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 host 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 register 0x2c is 2'b11).

26.3.6 Continuous mode

Setup **VIP_CAPTURE_F1SA_<Y, UV>, and VIP_CAPTURE_F2SA_<Y, UV>**. configure register **0x10** to set working mode、output data format…etc, configure register **0x38** to set crop start point and configure register **0x30** to set you whole 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** (bit0) to start continuous mode. If you setup register **0x4** to be value 0x1 you will get a complete interrupt from VIP after every frame end. After capturing, you just only display these frames by VIP. 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/UV start address; 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 host disable VIP directly even the F1 & F2 status are both in data ready state.

Chapter 27 NAND Flash Controller

27.1 Design Overview

27.1.1 Overview

NAND FLASH is an important device to store data. It is popular with electronic product. NAND Controller Can be good in transfer data between Host System and FLASH. Hardware ECC Can Correct Error at any position in one Code-Word. NAND Flash Timing can be configured by Host directly or by Internal DMA transfer.

27.1.2 Key features

- Support AHB 2.0 Slave Interface
- Support Internal DMA transfer and Host Transfer
- Support 2k Page and 4k Page by two configure
- Support 8 bit Flash interface
- Support Hardware ECC
- Support 4 Flash Device
- Support FF code auto correct Process

For detail information about NAND FLASH controller, pleaser refer **RK27xx NAND Flash Controller.pdf**。

Chapter 28 High-Speed ADC Interface

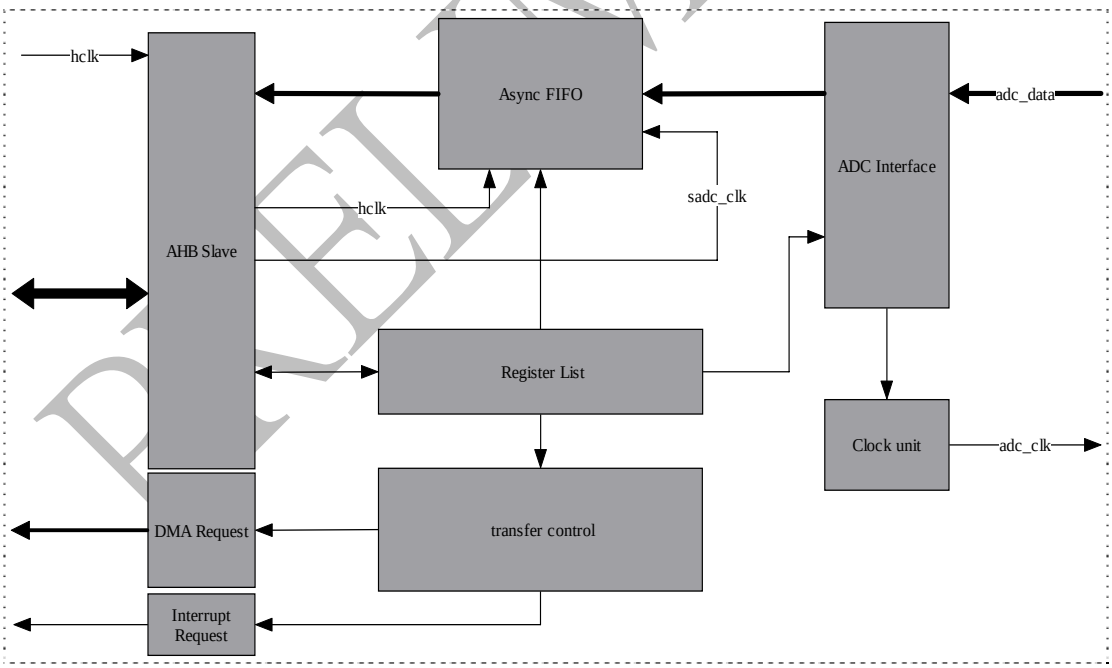
28.1 Design Overview

HS-ADC Interface Unit is interface unit of connected the High Speed AD Converter to AMBA AHB bus. That implement bus speed convert at low speed AD Converter bus to high speed AHB bus. HS-ADC Interface Unit fetch the bus data by the AD converter and store that to asynchronous FIFO after the AD clock is active when OS configure completion by DMA and HS-ADC Interface Unit. The HS-ADC Interface Unit generates the DMA request signal When data length of the asynchronous FIFO over then almost full level or almost empty level.

Key features

- Support the burst transfers and that type include SINGLE, INCR4, INCR8, INCR16.
- Support HS-ADC Interface Unit Enable and Disable. Notice that controller register can be modified when HS-ADC Interface Unit Disabled.
- Support 8-bit/10-bit data bus by the AD converter.
- Support the most significant bit negation.
- Support store to high 8-bit/10-bit or low 8-bit/10-bit at a haft word width. Sample the 8-bit data by the AD converter store to high 8-bit is between the data[15] to data[8] by a haft word width. And that have sign extend if store to low 8-bit/10-bit by a haft word width.
- Support DMA transfers mode and that generate DMA request from the event of almost full or almost empty in the asynchronous FIFO. The almost full/almost empty level can be configuration.

28.2 Architecture



28.3 Register

Register summary

Name	Offset	Size	Reset Value	Description
HSADC_DATA	0x00	w	--	data register

HSADC_CTRL	0x04	w	0x0000_0000	control register
HSADC_IER	0x08	w	0x0000_0000	interrupt enable/mask register
HSADC_ISR	0x0c	w	0x0000_0000	interrupt status register

Detail Register Description

HSADC_CTRL
Address: Operational Base + offset(0x04)
The controls register of A/D Converter.

Bit	Access	Reset Value	Description
31:28			Reserved
27:24	R/W	0	Define almost full trigger level: 0x0~"0xf" - configure valid range (Notes: 1 level indicate 4 entries data in the async FIFO. and this configure range mapping to 64 - 124 entries data in the async FIFO.)
23:20			Reserved
19:16	R/W	0	Define almost empty trigger level: 0x0~"0xf" - configure valid range (Notes: 1 level indicate 4 entries data in the async FIFO. and this configure range mapping to 0 - 60 entries data in the async FIFO.)
15:6			Reserved
5	R/W	0	DMA request mode: 1 - almost full generate DMA request signal (Notes: this mode generate DMA request signal from almost full condition and cancel DMA request signal from almost empty condition. so you need configure two level by almost full level and almost empty level) 0 - almost empty generate DMA request signal (Notes: this mode generate DMA request signal from almost empty condition and that only once DMA request.)
4	R/W	0	control the most significant bit negation: "1" - negation "0" - not negation
3	R/W	0	fetch the bus data by AD converter and that store to high 8-bit/10-bit or low 8-bit/10-bit at a haft word width before push to Async FIFO： "1" - store to high 8-bit/10-bit "0" - store to low 8-bit/10-bit (Notes: have sign extend if that configure of store to low 8-bit/10-bit)
2	R/W	0	The data bus width of AD converter： "1" - 10-bit "0" - 8-bit
1	R/W	0	the clock of AD converter and that clock supply to AD converter. Sample 8.192Mhz use to DAB clock： "1" - clock negation "0" - clock not negation
0	R/W	0	HS-ADC Interface Unit Enable Bit: "1" - enable (Notes: will return 1 when the hardware started transfer)

			"0" - disable (Notes: other bit can be modify only the hardware return 0)

HSADC_IER
Address: Operational Base + offset(0x08)
The interrupt enable control register of A/D Converter.

Bit	Access	Reset value	Description
31:2			
1	R/W	0	Interrupt enable/mask bit for enable/disable the empty interrupt flag of Async FIFO "1" - enable "0" - disable
0	R/W	0	Interrupt enable/mask bit for enable/disable the full interrupt flag of Async FIFO "1" - enable "0" - disable

HSADC_ISR
Address: Operational Base + offset(0x0C)
The interrupt statuses register of A/D Converter.

Bit	Access	Reset value	Description
31:2			
1	RW	0	Async FIFO empty interrupt flag. "1(R)" - This bit will be set to "1" when Async FIFO empty status and that only to read operation. "0(W)" - Write "0" to bit for clear the interrupt flag and that only to wrtie operation.
0	RW	0	Async FIFO full interrupt flag. "1(R)" - This bit will be set to "1" when Async FIFO full status and that only to read operation. "0(W)" - Write "0" to bit for clear the interrupt flag and that only to wrtie operation.

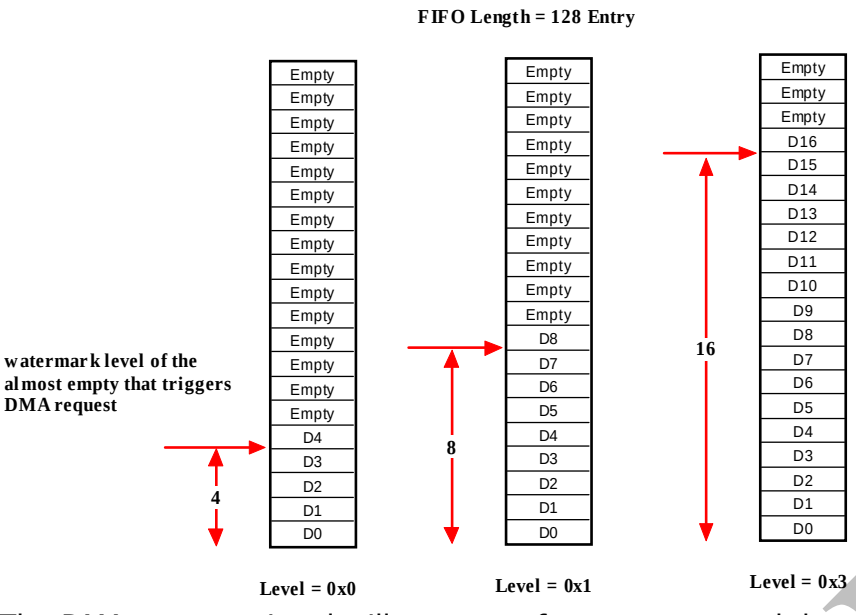
HSADC_DATA
Address: Operational Base + offset(0x00)
The data register of A/D Converter.

Bit	Access	Reset value	Description
31:0	R	--	the data of async FIFO for the DMA read.

28.4 Application Notes

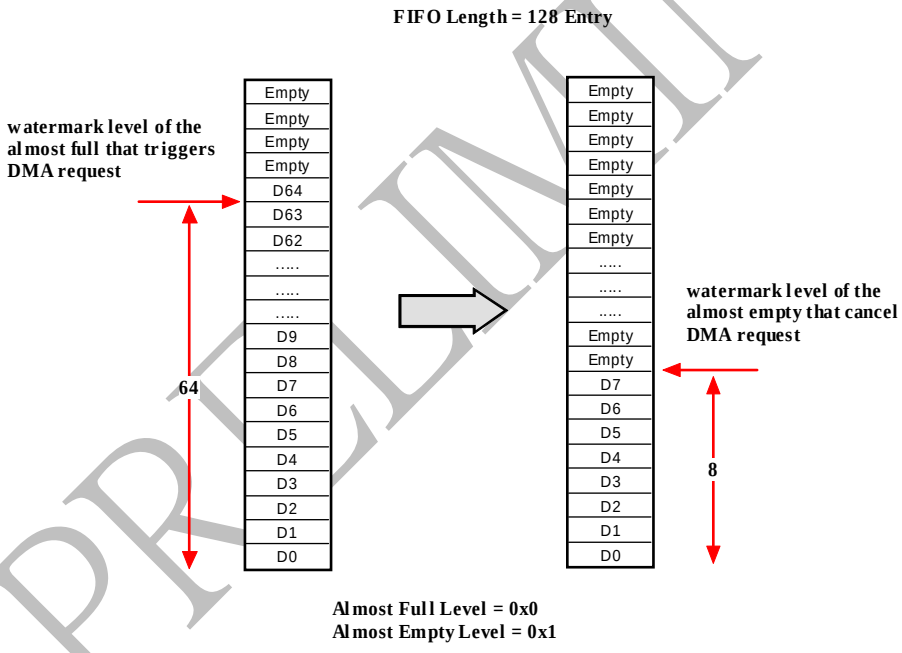
The following sections will describe the operation of DMA request and DMA transfers:

Almost empty triggers a DMA request by DMA request mode



The DMA request signal will generate from a watermark level triggers when store to FIFO data over the watermark level of almost empty and that watermark level can be configuration to HSADC_CTRL[19:16] by controls register . This DMA request mod doesn't care the watermark level of almost full. As show in figure 10 that sample to watermark level configuration.

Almost full triggers a DMA request by DMA request mode



The DMA request signal will generate from a watermark level triggers when store to FIFO data over the watermark level of almost full and at this DMA request mode that continue generates request signal when the number of FIFO data great then watermark level of almost empty. This DMA request mode need configuration double watermark level and that is watermark level of almost empty at the HSADC_CTRL[19:16] and watermark level of almost full at the HSADC_CTRL[27:24]. As show in figure 11 that sample to watermark level configuration.

Chapter 29 Audio codec

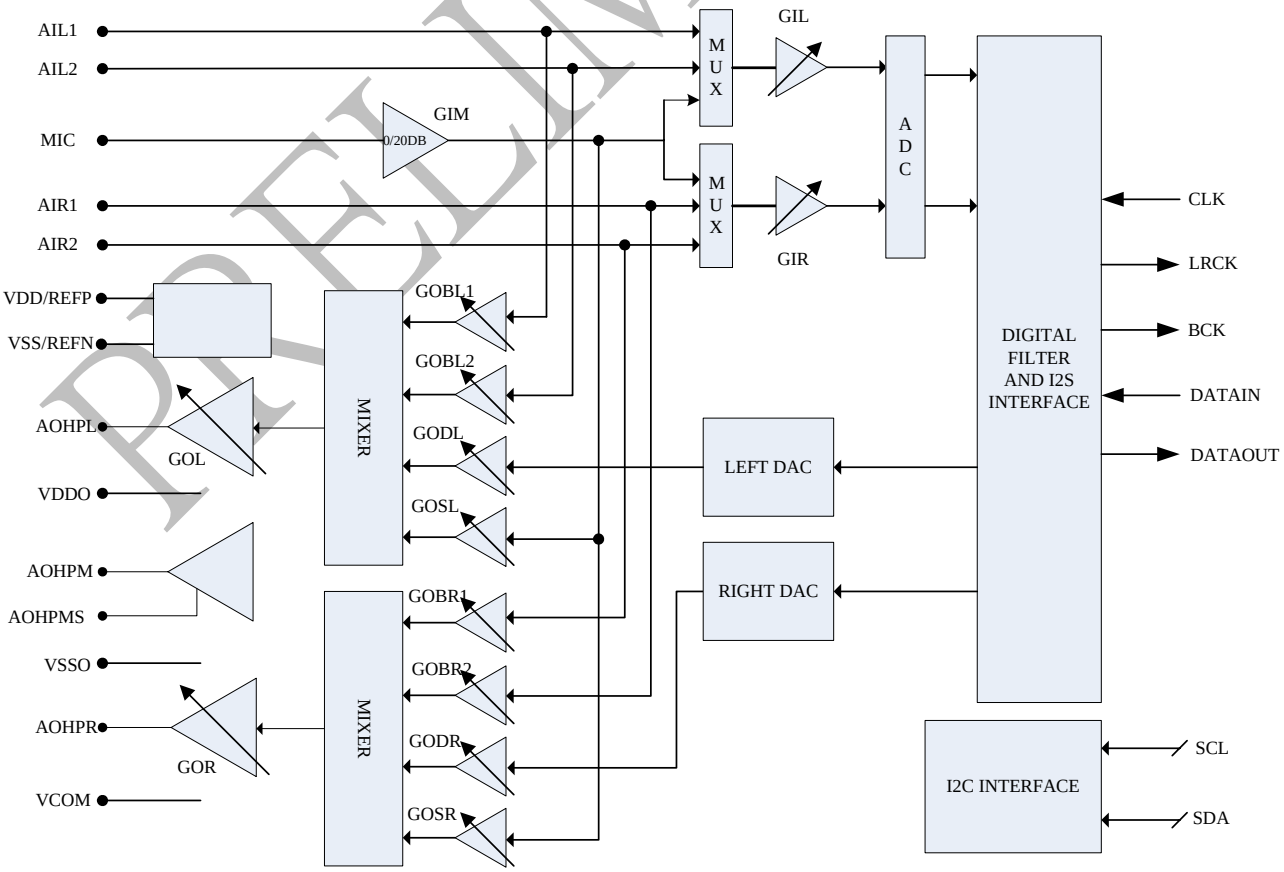
29.1 Design Overview

Audio codec in RK27xx is containing a stereo codec (ADC +DAC) and additional analog functions offering an ideal mixed signal front end for high quality audio applications. Audio codec receive or transmit audio data stream from I2S module, controlled by I2C module.

29.2 Feature

- 24bit audio codec with MIC/line input and headphone driver.
- Single +3.3V (from 2.7V to 3.6V) power supply for the analog part.
- High SNR performance:
  - 95DB for DAC to headphone
  - 85DB from line input to ADC
- THD: 0.06% @33mW/16Ohm for DAC to headphone output path
- High level output power @3.3V:
  - 40mW on 16Ohm load
  - 20mW on 32Ohm load
- Separate power-down modes for ADC and DAC (shutdown mode, stand-by mode)
- Reduction of audible glitches systems:
  - Pop reduction system
  - Soft mute mode
- 12MHz or 16.9344MHz operation.

29.3 BLOCK DIAGRAM



29.4 Function register

29.4.1 Registers Summary

Register Name	Function	Address	Reset value
AICR	Audio Interface Control	(REG_ADD, 00000)	h0C
CR1	Control Register 1	(REG_ADD, 00001)	hA8
CR2	Control Register 2	(REG_ADD, 00010)	h78
CCR1	Control Clock Register 1	(REG_ADD, 00011)	h00
CCR2	Control Clock Register 2	(REG_ADD, 00100)	h00
PMR1	Power Mode Register 1	(REG_ADD, 00101)	hFF
PMR2	Power Mode Register 2	(REG_ADD, 00110)	h03
CRR	Control Ramp Register	(REG_ADD, 00111)	h51
ICR	Interrupt Control Register	(REG_ADD, 01000)	h3F
IFR	Interrupt Flag Register	(REG_ADD, 01001)	h00
CGR1	Control Gain Register 1	(REG_ADD, 01010)	h00
CGR2	Control Gain Register 2	(REG_ADD, 01011)	h04
CGR3	Control Gain Register 3	(REG_ADD, 01100)	h04
CGR4	Control Gain Register 4	(REG_ADD, 01101)	h04
CGR5	Control Gain Register 5	(REG_ADD, 01110)	h04
CGR6	Control Gain Register 6	(REG_ADD, 01111)	h04
CGR7	Control Gain Register 7	(REG_ADD, 10000)	h04
CGR8	Control Gain Register 8	(REG_ADD, 10001)	h0A
CGR9	Control Gain Register 9	(REG_ADD, 10010)	h0A
CGR10	Control Gain Register 10	(REG_ADD, 10011)	h00

29.4.2 Detail Register Description

AICR: Audio Interface Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0
-	-	-	-	DAC_SERIAL	ADC_SERIAL	DAC_I2S	ADC_I2S

Bit 3: **DAC_SERIAL**: Selection of the DAC digital serial audio interface
Read/Write
0= Parallel interface
1= Serial interface

Bit 2: **ADC_SERIAL**: Selection of the ADC digital serial audio interface
Read/Write
0= Parallel interface
1= Serial interface

Bit 1: **DAC_I2S**: Working mode of the DAC digital serial audio interface
Read/Write
0= DSP mode
1= I2S mode

Bit 0: **ADC_I2S**: Working mode of the ADC digital serial audio interface
Read/Write
0= DSP mode
1= I2S mode

CR1: Control Register 1

R/W-1	R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0
SB_MICBIAS	**MONO**	**DAC_MUTE**	**HP_DIS**	**DACSEL**	**BYPASS1**	**BYPASS2**	**SIDETONE**

Bit 7: **SB_MICBIAS**: Microphone biasing buffer power-down
Read/Write
0= active
1= power-down

Bit 6: **MONO**: Stereo-to-mono conversion for DAC path
Read/Write
0= stereo
1= mono

Bit 5: **DAC_MUTE**: DAC soft mute mode
Read/Write
0= mute inactive, digital input signal transmitted to the DAC
1= puts the DAC in soft mute mode

Bit 4: **HP_DIS**: HeadPhone output signal disabled
Read/Write
0= Signal applied to headphone outputs
1= no signal on headphone outputs, acts as a mute signal

Bit 3: **DACSEL**: Mixer input selection
Read/Write
0= DAC output ignored in input of the mixer
1= DAC output selected as an input of the mixer

Bit 2: **BYPASS1**: Mixer input selection (line 1)
Read/Write
0= Bypass path ignored in input of the mixer
1= Bypass path selected as an input of the mixer

Bit 1: **BYPASS2**: Mixer input selection (line 2)
Read/Write
0= Bypass path ignored in input of the mixer
1= Bypass path selected as an input of the mixer

Bit 0: **SIDETONE**: Mixer input selection
Read/Write
0= Sidetone path ignored in input of the mixer
1= Sidetone path selected as an input of the mixer

CR2: Control Register 2

R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0
DAC_DEEMP	**DAC_ADWL1**	**DAC_ADWL0**	**ADC_ADWL1**	**ADC_ADWL0-**	**ADC_HPF**	**INSEL1**	**INSEL0**

Bit 7: **DAC_DEEMP**: DAC De-emphasize filter enable
Read/Write
0= inactive
1= enables the deemphasis filter

Bit 6-3: **DAC_ADWL[1:0], ADC_ADWL[1:0]**: Audio Data Word Length: for respectively DAC and ADC paths
Read/Write:

Bit 2: **ADC_HPF**: ADC High Pass Filter enable

Read/Write:
0= inactive

1= enables the ADC High Pass Filter

Bit 1-0: **INSEL[1:0]**: selection of the signal converted by the ADC Read/Write:
00 = Line 1 input
01 = Line 2 input
10 = Microphone input
11 = Mixer output

CCR1: Control Clock Register 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	-	-	CRYSTAL3	CRYSTAL2	CRYSTAL1	CRYSTAL0

Bit 3-0: **CRYSTAL[3:0]:** Selection of the MCLK frequency Read/Write
The sampling frequency value is given in the FREQ[3:0] table

CRYSTAL[3:0]	Master Clock Frequency
0000	12 MHz
0001	16.9344 MHZ

CCR2: Control Clock Register 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DFREQ3	DFREQ2	DFREQ1	DFREQ0	AFREQ3	AFREQ2	AFREQ1	AFREQ0

Bit 7-4: **DFREQ[3:0]:** Selection of the DAC sampling rate (Fs) Read/Write
The sampling frequency value is given in the FREQ[3:0] table

Bit 3-0: **AFREQ[3:0]:** Selection of the ADC sampling rate (Fs) Read/Write
The sampling frequency value is given in the FREQ[3:0] table

FREQ[3:0]	Sampling Rate (Fs)
0000	96kHz
0001	48kHz
0010	44.1kHz
0011	32kHz
0100	24kHz
0101	22.05kHz
0110	16kHz
0111	12kHz
1000	11.025kHz
1001	9.6kHz
1010	8kHz
1011	8kHz
1100	8kHz
1101	8kHz
1110	8kHz
1111	8kHz

PMR1: Power Mode Register 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SB_DAC	**SB_OUT**	**SB_MIX**	**SB_ADC**	**SB_IN1**	**SB_IN2**	**SB_MIC**	**SB_IND**

Bit 7: **SB_DAC**: DAC power-down mode
Read/Write
0= active
1= power-down

Bit 6: **SB_OUT**: Output Stage power-down mode
Read/Write
0= active
1= power-down

Bit 5: **SB_MIX**: Mixer and line output stage power-down
Read/Write
0= active
1= power-down

Bit 4: **SB_ADC**: ADC power-down mode
Read/Write
0= active
1= power-down

Bit 3: **SB_IN1**: Line 1 Input conditioning circuitry power-down mode
Read/Write
0= active
1= power-down

Bit 2: **SB_IN2**: Line 2 Input conditioning circuitry power-down mode
Read/Write
0= active
1= power-down

Bit 1: **SB_MIC**: Microphone Input conditioning circuitry power-down mode
Read/Write
0= active
1= power-down

Bit 0: **SB_IND**: Mixer to ADC circuitry power-down mode
Read/Write
0= active
1= power-down

PMR2: Power Mode Register 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
LRGI	**RLGI**	**LRGOD**	**RLGOD**	**GIM**	**SB_MC**	**SB**	**SB_SLEEP**

Bit 7-6: **LRGI, RLGI:** PGATM input gain coupling
Read/Write
00: Left and right channels gains are independent, respectively given by GIL and GIR
10: Left and right channels gain is given by GIL
01: Left and right channels gain is given by GIR
11: Left and right channels gain is given by GIL

Bit 5-4: **LRGOD, RLGOD:** DAC mixing gain coupling
Read/Write
00: Left and right channels gains are independent, respectively given by GODL and GODR
10: Left and right channels gain is given by GODL

01: Left and right channels gain is given by GODR
11: Left and right channels gain is given by GODL

Bit 3: **GIM**: Microphone amplifier gain control
Read/Write
0= 0 dB gain
1= 20 dB gain

Bit 2: **SB_MC**: Output Stage common mode buffer power-down
Read/Write
0= active (capacitor less headphone output configuration)
1= power-down (line output configuration)

Bit 1: **SB**: complete power-down mode
Read/Write
0= normal mode (active)
1= complete power-down

Bit 0 **SB_SLEEP**: sleep mode
Read/Write
0= normal mode (active)
1= sleep mode

CRR: Control Ramp Register

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1
-	RATIO1	RATIO0	KFAST2	KFAST1	KFAST0	TRESH1	TRESH0

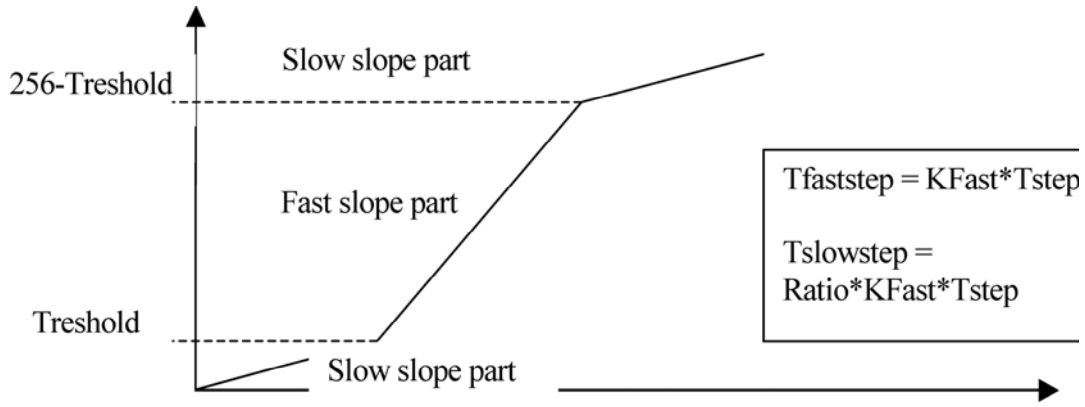
Bit 6-5: **RATIO[1:0]:** ratio between fast and slow steps
Read/Write:
00: Ratio =1 01: Ratio = 2
10: Ratio = 4 (default) 11: Ratio = 8

Bit 4-2: **KFAST[2:0]:** factor for step time in fast slope part
Read/Write:
000: KFast =1 001: KFast = 2
010: KFast = 4 011: KFast = 8
100: KFast =16 (default) 101: KFast = 32

Bit 1 - Bit 0: **TRESH[1:0]:** threshold between fast and slow slope parts
Read/Write:
00: Threshold = 0 01: Threshold = 32 (default)
10: Threshold = 64 11: Threshold = 128

Tstep duration = 31.25us

Fs (kHz)	Tstep (ratio)	Tstep (us)
8	0.25/Fs	31.25
9.	0.3/Fs	31.25
11.025	0.345/Fs	31.25
12	0.375/Fs	31.25
16	0.5/Fs	31.25
22.05	0.69/Fs	31.25
24	0.75/Fs	31.25
32	1/F	31.25
44.1	1.38/Fs	31.25
48	1.5/Fs	31.25
96	3/F	31.25



Step time on fast part: KFast*Tstep
Step time on slow part: Ratio*KFast*Tstep

Ramp Time duration: 224ms (default)

Tramp = KFast*Tstep*(256-2*Thresh) + R*KFast*Tstep*2*Thresh

Note:
When TRESH="11", the counter stays 2*Tslowstep on 127 at the middle of the falling and 2*Tslowstep on 128 at the middle of the rising.

ICR: Interrupt Control Register

R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
INT_FORM[1]	**INT_FORM[0]**	**JACK_MASK**	**CCMC_MASK**	**RUD_MASK**	**RDD_MASK**	**GUD_MASK**	**GDD_MASK**

Bit 7-6: **INT_FORM[1:0]:** Waveform and polarity of the IRQ signal
Read/Write
00: The generated IRQ is a high level
01: The generated IRQ is a low level
10: The generated IRQ is a high level pulse with an 8 MC_CLK cycles duration when using 8-bit parallel control interface or 8 MCLK cycles duration when using I2C control interface
11: The generated IRQ is a low level pulse with an 8 MC_CLK cycles duration when using 8-bit parallel control interface or 8 MCLK cycles duration when using I2C control interface

Bit 5: **JACK_MASK:** Mask for the JACK_EVENT flag
Read/Write
0: interrupt enabled
1: interrupt masked (no IRQ generation)

Bit 4: **CCMC_MASK:** Mask for the CCMC flag
Read/Write
0: interrupt enabled
1: interrupt masked (no IRQ generation)

Bit 3: **RUD_MASK:** Mask for the RAMP_UP_DONE flag
Read/Write
0: interrupt enabled
1: interrupt masked (no IRQ generation)

Bit 2: **RDD_MASK:** Mask for the RAMP_DOWN_DONE flag
Read/Write
0: interrupt enabled
1: interrupt masked (no IRQ generation)

Bit 1: **GUD_MASK:** Mask for the GAIN_UP_DONE flag
Read/Write
0: interrupt enabled
1: interrupt masked (no IRQ generation)

Bit 0: **GDD_MASK:** Mask for the GAIN_DOWN_DONE flag
Read/Write
0: interrupt enabled
1: interrupt masked (no IRQ generation)

Notes:
- When an interrupt is masked, the event do not generates any change on the IRQ signal, but the corresponding flag value is set to '1' in the IFR register.
- When the IRQ signal is active on level, the IRQ signal is set to the inactive level while the bits IFR[5:0] & (!ICR[5:0]) equals '0'.
- When the IRQ signal is a pulse, the IRQ signal is set to the inactive state until a new non-masked event occurs in IFR[5:0] or until a masked event is unmasked.
- If the 8-bit parallel control interface is selected, MC_CLK must not be stopped in order to propagate IRQ signal.

IFR: Interrupt Flag Register

R/W-0	R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
-	JACK	JACK_ EVENT	CCMC	RAMP_UP_ DONE	RAMP_DOWN_ DONE	GAIN_UP_ DONE	GAIN_DOWN_ DONE

Bit 6: **JACK**: Output Jack plug detection status
Read
0 = no jack
1 = output jack present

Bit 5: **JACK_EVENT**: Event on output Jack plug detection status
Read
0 = no event
1 = event detected (due to JACK flag change to '0' or '1')
Write
1 = Reset of the flag

Bit 4: **CCMC**: Output short circuit detection status – Reserved for future use
Read
0 = inactive
1 = indicates that a short circuit has been detected by the output stage.

Bit 3: **RAMP_UP_DONE**: End of output stage ramp up flag
Read
1 = the ramp-up sequence is completed (output stage is active).
Write
1 = Reset of the flag

Bit 2: **RAMP_DOWN_DONE**: End of output stage ramp down flag
Read
1 = the ramp-down sequence is completed (output stage in stand-by mode).
Write
1 = Reset of the flag

Bit 1: **GAIN_UP_DONE**: End of mute gain up sequence flag
Read
1 = the mute sequence is completed, the DAC input signal is transmitted to the DAC path.
Write
1 = Reset of the flag

Bit 0: **GAIN_DOWN_DONE**: End of mute gain down sequence flag

Read
1 = the mute sequence is completed, a 0 DC signal is transmitted to the DAC path.
Write
1 = Reset of the flag

CGR1: Control Gain Register n°1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GODR3	**GODR2**	**GODR1**	**GODR0**	**GODL3**	**GODL2**	**GODL1**	**GODL0**

Bit 7-4: **GODR[3:0]:** DAC mixing right channel gain programming value
Read/Write

Bit 3-0: **GODL[3:0]:** DAC mixing left channel gain programming value
Read/Write

CGR2: Control Gain Register n°2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
LRGOB1	**RLGOB1**	**-**	**GOBL14**	**GOBL13**	**GOBL12**	**GOBL11**	**GOBL10**

Bit 7-6: **LRGOB1, RLGOB1:** Line 1 mixing gain coupling
Read/Write
00: Left and right channels gains are independent, respectively given by GOBL1 and GOBR1
10: Left and right channels gain is given by GOBL1
01: Left and right channels gain is given by GOBR1
11: Left and right channels gain is given by GOBL1

Bit 4-0: **GOBL1[4:0]:** Line 1 mixing left channel gain programming value
Read/Write

CGR3: Control Gain Register n°3

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
			GOBR14	**GOBR13**	**GOBR12**	**GOBR11**	**GOBR10**

Bit 4-0: **GOBR1[4:0]:** Line 1 mixing right channel gain programming value
Read/Write

CGR4: Control Gain Register n°4

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
LRGOB2	**RLGOB2**		**GOBL24**	**GOBL23**	**GOBL22**	**GOBL21**	**GOBL20**

Bit 7-6: **LRGOB2, RLGOB2:** Line 2 mixing gain coupling
Read/Write
00: Left and right channels gains are independent, respectively given by GOBL2 and GOBR2
10: Left and right channels gain is given by GOBL2
01: Left and right channels gain is given by GOBR2
11: Left and right channels gain is given by GOBL2

Bit 4-0: **GOBL2[4:0]:** Line 2 mixing left channel gain programming value
Read/Write

CGR5: Control Gain Register n°5

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
			GOBR24	**GOBR23**	**GOBR22**	**GOBR21**	**GOBR20**

Bit 4-0: **GOBR2[4:0]:** Line 2 mixing right channel gain programming value
Read/Write

CGR6: Control Gain Register n°6

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
LRGOS	**RLGOS**		**GOSL4**	**GOSL3**	**GOSL2**	**GOSL1**	**GOSL0**

Bit 7-6: **LRGOS, RLGOS:** Microphone mixing gain coupling
Read/Write
00: Left and right channels gains are independent, respectively given by GOSL and GOSR
10: Left and right channels gain is given by GOSL
01: Left and right channels gain is given by GOSR
11: Left and right channels gain is given by GOSL

Bit 4-0: **GOSL[4:0]:** Microphone mixing left channel gain programming value
Read/Write

CGR7: Control Gain Register n°7

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0
			GOSR4	**GOSR3**	**GOSR2**	**GOSR1**	**GOSR0**

Bit 4-0: **GOSR[4:0]:** Microphone mixing right channel gain programming value
Read/Write

CGR8: Control Gain Register n°8

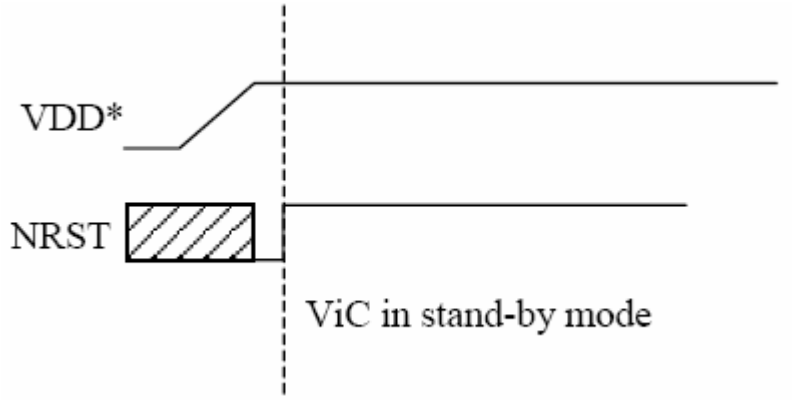
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1	R/W-0
LRGO	**RLGO**		**GOL4**	**GOL3**	**GOL2**	**GOL1**	**GOL0**

Bit 7-6: **LRGO, RLGO:** Output stages gain coupling
Read/Write
00: Left and right channels gains are independent, respectively given by GOL and GOR
10: Left and right channels gain is given by GOL
01: Left and right channels gain is given by GOR
11: Left and right channels gain is given by GOL

29.5 Application note

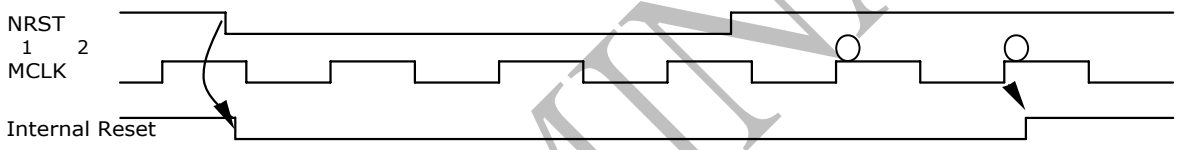
29.5.1 Power-on and power-off modes

When the power supply ramps up, audio codec enters the power-on mode. During the reset, the codec is put in stand-by in order to reduce audible pops.



The codec doesn't handle the power supply ramp down on its own. The application has to turn the codec in complete stand-by mode before the power supply starts to ramp down.

29.5.2 Asynchronous reset



The reset input signal is asynchronous and is twice resynchronized on the internal clock domain to prevent erroneous reset due to glitches.. The reset minimum duration is one MCLK cycle.

Except during the power-up mode, do not perform reset in order to avoid audible pops. Resetting the codec during normal operating mode will turn instantaneously the codec in stand-by mode. This will lead to generate a large audible pop.

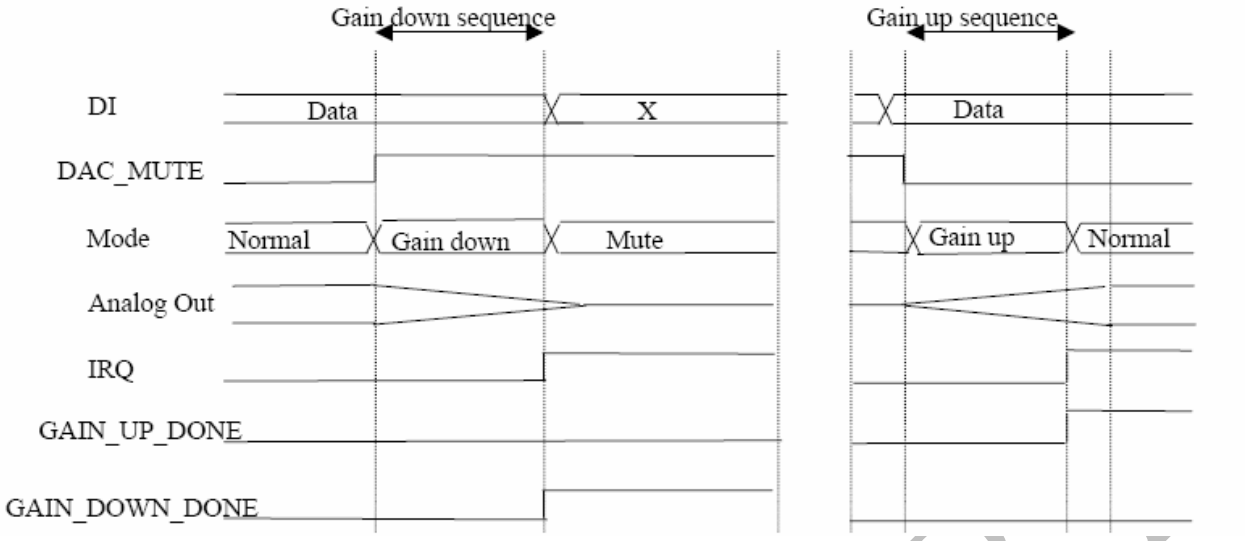
29.5.3 Soft mute mode

A soft mute is implemented in order to reduce audible parasites when the DAC enters or leaves the mute mode. It is controlled by the DAC_MUTE register bit.

Setting DAC_MUTE to '1'puts the DAC in mute mode. The codec decreases progressively the gain in the digital part (DWL block) from 0dB to -∞. When the gain down sequence is completed, the signal in output of the DAC is equal to 0 whatever the value of the digital input data is. The codec then generates an IRQ and set GAIN_DOWN_DONE register bit to '1'.

During soft mute mode, the DAC is still converting but the output final voltages (AOL, AOR) are equal to VREF/2.

In the opposite, when DAC_MUTE is set to '0', the DAC leaves the mute mode by increasing progressively the gain in the digital part from -∞ to 0dB. When the gain up sequence is completed, the DAC returns in normal mode. The codec then generates an IRQ and set GAIN_UP_DONE register bit to '1'.



The duration of gain down and gain up sequences are nearly independent of Fs as shown below:

Fs(kHz)	Time(ms)	Fs(kHz)	Time(ms)	Fs(kHz)	Time(ms)
96	17.72	24	17.25	11.025	17.73
48	17.72	22.05	17.73	9.6	17.98
44.1	17.73	16	17.25	8	17.25
32	17.96	12	17.25		

Do not change the value of DAC_MUTE while the effect of the previous change is not reached (working not guaranteed).

Do not enter in stand-by mode while the gain sequence is not completed (working not guaranteed).

29.5.4 Power-down and sleep modes

Twelve stand-by inputs allow putting independently the different parts of codec in power-down mode.

Different working modes are sum-up in the following table (non exhaustive table):

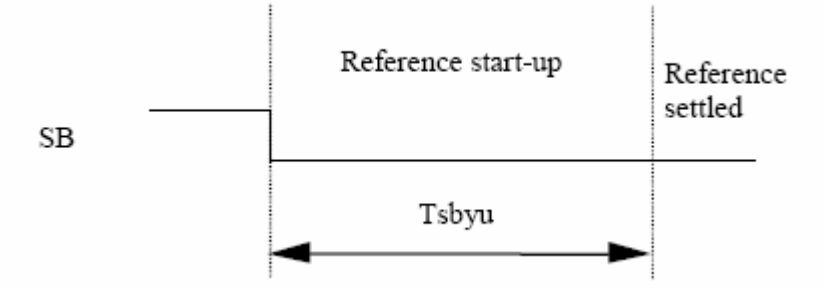
Mode	SB	SB SLEEP	SB DAC	SB MIX	SB OUT	SB MC	SB ADC	SB MIC	SB	SB IN1	SB IN2	SB IND	INSEL	DACSEL	BYPASS1	BYPASS2	SIDETONE	HP DIS	DAC MUTE
Reset mode (complete power-down mode)	1	1	1	1	1	0	1	1	1	1	1	1	0	1	0	0	0	0	1
Complete power-down mode	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Sleep mode	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Record mode, Line input 1	0	0	X	X	X	X	0	X	X	0	X	X	0	X	X	X	X	X	X
Record mode, Line input 2	0	0	X	X	X	X	0	X	X	X	0	X	1	X	X	X	X	X	X
Record mode, Microphone input	0	0	X	X	X	X	0	0	X	X	X	X	2	X	X	X	X	X	X
Record mode, Mixer output	0	0	X	0	0	X	0	X	X	X	X	0	3	X	X	X	X	X	X
Playback mode, DAC to 16 Ohm headphone out	0	0	0	0	0	0	X	X	X	X	X	X	X	1	0	0	0	0	0
Playback mode, DAC to 10 kOhm line out	0	0	0	0	1	X	X	X	X	X	X	X	X	1	0	0	0	1	0
Playback mode, Line1 to 16 Ohm headphone out	0	0	X	0	0	0	X	X	X	0	X	X	X	0	1	0	0	0	X
Playback mode, Line1 to 10 kOhm line out	0	0	X	0	1	X	X	X	X	0	X	X	X	0	1	0	0	1	X
Playback mode, Line2 to 16 Ohm headphone out	0	0	X	0	0	0	X	X	X	X	0	X	X	0	0	1	0	0	X
Playback mode, Line2 to 10 kOhm line out	0	0	X	0	1	X	X	X	X	X	0	X	X	0	0	1	0	1	X
Playback mode, MIC to 16 Ohm headphone out	0	0	X	0	0	0	X	0	X	X	X	X	X	0	0	0	1	0	X
Playback mode, MIC to 10 kOhm line out	0	0	X	0	1	X	X	0	X	X	X	X	X	0	0	0	1	1	X
Playback mode, all inputs mixed to 16 Ohm headphone out	0	0	0	0	0	0	X	0	X	0	0	X	X	1	1	1	1	0	0
Playback mode, all inputs mixed to 10 kOhm line out	0	0	0	0	1	X	X	0	X	0	0	X	X	1	1	1	1	1	0
Playback mode, all inputs mixed to 16 Ohm headphone out, record line or mic input	0	0	0	0	0	0	0	0	X	0	0	X	0 1 2	1	1	1	1	0	0
Playback mode, all inputs mixed to 10 kOhm line out, record line or mic input	0	0	0	0	1	X	0	0	X	0	0	X	0 1 2	1	1	1	1	1	0
Playback mode, all inputs mixed to 16 Ohm headphone out, record mixer output	0	0	0	0	0	0	0	0	X	0	0	0	3	1	1	1	1	0	0
Playback mode, all inputs mixed to 10 kOhm line out, record mixer output	0	0	0	0	1	X	0	0	X	0	0	0	3	1	1	1	1	1	0
"Default mode"	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0

29.5.5 Complete stand-by mode

This codec (including all functions i.e. ADC path, DAC path and references) is put in stand-by mode when the SB register bit equals '1'.

During the complete power-down mode, the power consumption is reduced to a minimum.

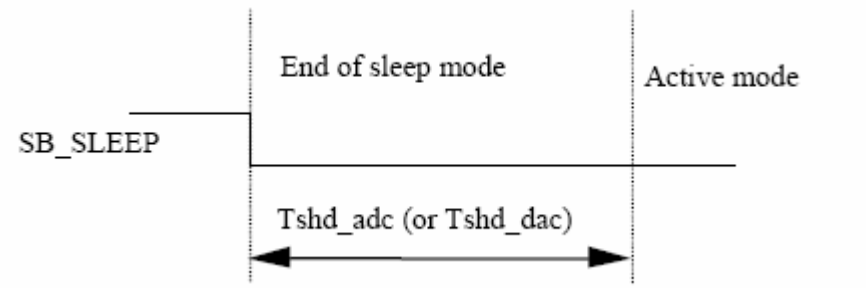
When SB is set to '0', codec leaves the complete power-down mode; it is necessary to wait some time before the Codec references settle. This time is equal to Tsbyu.



29.5.6 Sleep mode

When SB_SLEEP is set to '1', shCODIp-100.01-HD enters in sleep mode. The logical part and the analog functions - except the voltage and biasing references - enter in power-down mode. So, the power consumption is reduced without penalizing the start-up time.

When SB_SLEEP falls, shCODIp-100.01-HD leaves the corresponding stand-by mode; it is necessary to wait some time before the Codec reaches the normal mode. Depending on SB- DAC, SB_OUT, SB_MC, SD_ADC, SB_MIC, SB_IN1, SB_IN2 and SB_IND, this time is either called Tshd_adc for the ADC path or Tshd_dac for the DAC path.



29.5.7 Starting and stopping sequences (pop reduction)

Due to the large number of stand-by combinations and to be the most flexible, the handling of the sequencement from one working mode to another is let to the application. To help the SoC designer and the software designer in this task, some specific sequences are automatically performed by shCODIp-100.01-HD and an interrupt mechanism (IRQ signal and associated registers) warns the application when these sequences end.

The main idea of this section is to describe the sequences to perform to minimize the audible pop to the minimum for the headphone output.

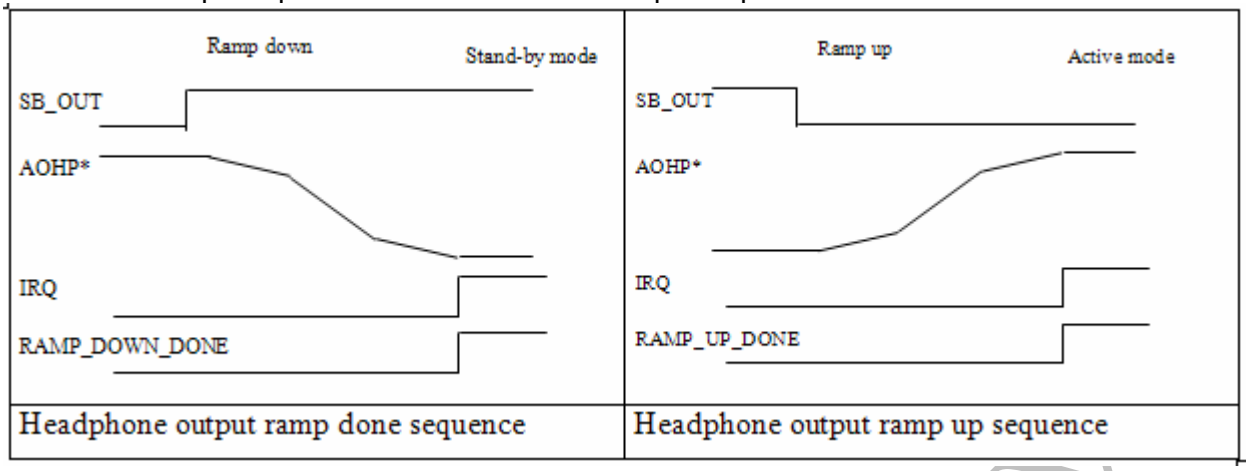
Note that the following sequences do not perform audible glitch reduction on the line output. The stage connected to the line output has to handle the pop reduction system

For Headphone stage ramping system,An internal mechanism is used to reduce output glitches when the headphone stage enters or leaves the power-down mode.

When the SB_OUT is set to '1', the headphone output voltages (AOHPL, AOHPR, and AOHPM) are slowly decreased in the same time from VDDA/2 down to 0. The output ramp waveform is programmable thanks to the CRR register (see CRR: Control Ramp Register).

When the SB_OUT is set to '0', the headphone output voltages (AOHPL, AOHPR, and AOHPM) are slowly increased in the same time from 0 to VDDA/2.

An interrupt request is sent when the ramp completes.



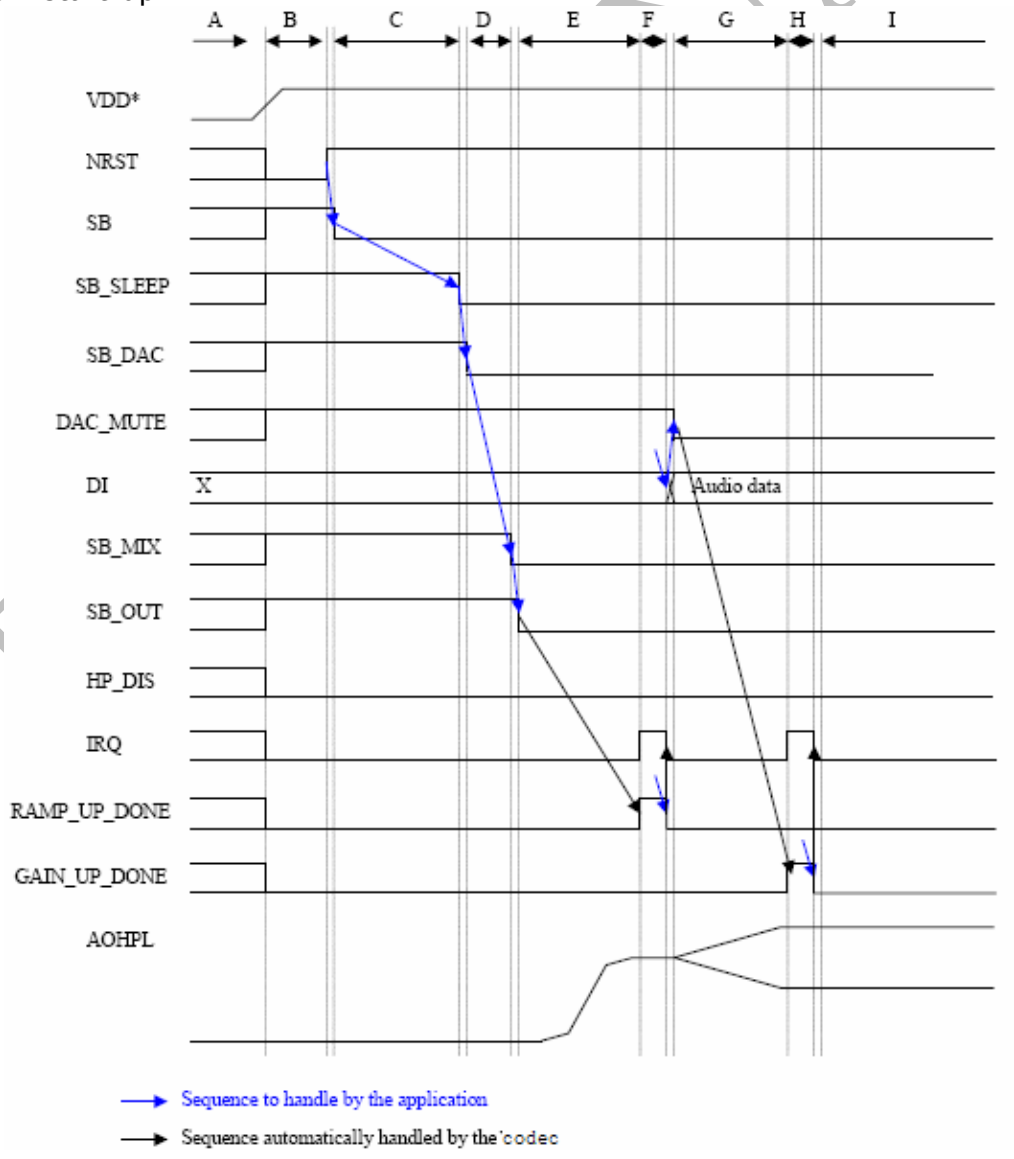
Headphone output ramp done sequence | Headphone output ramp up sequence

Do not change the level of SB_OUT as long as the sequence due to the previous change is not completed (working not guaranteed).

In order to prevent audible glitch, it is required to power-down the output stage (SB_OUT=1) before changing the type of output load (capacitor less load/capacitor coupled load) with SB_MC.

From power-on to DAC playback mode sequence

The intent of the following sequence is to prevent for large audible glitches due to the system start-up.



A, Initial state: the power supply is off.
B, Power supply ramp up: the application drives NRST to '0'.
C, Starting of codec reference: the application turns the Codec on sleep mode (SB register bit).
D: After waiting the Tsbyu duration (for example, on event generated by a timer at the application level), the application turns on the Codec DAC (SB_SLEEP and SB_DAC register bits)
E, Ramp up cycle: after waiting 1 ms (TBC), the application turns on the mixer and the headphone output stages.
F, IRQ generation: once the ramp up cycle completes, the Codec sets the RAMP_UP_DONEflag to '1' and generates an interrupt.
G, IRQ handling and gain up cycle: the application handles the interrupt and resets the RAMP_UP_DONE flag by writing '1' on it and releases the mute of the DAC (DAC_MUTE register bit). In the same time, the application sends valid audio data to the Codec DAC.
H, IRQ generation: once the gain up cycle completes, the Codec sets the GAIN_UP_DONE flagto '1' and generates an interrupt.
I, IRQ handling and active mode: the application handles the interrupt and resets theGAIN_UP_DONE flag by writing '1' on. The Codec DAC is now fully activated.

The sequence from C to I can be used to switch from the complete stand-by mode to the active mode.

The sequence from D to I can be used to switch from the sleep mode to the active mode.

29.5.8 Requirements on mixer and PGATM inputs selection and power-down modes

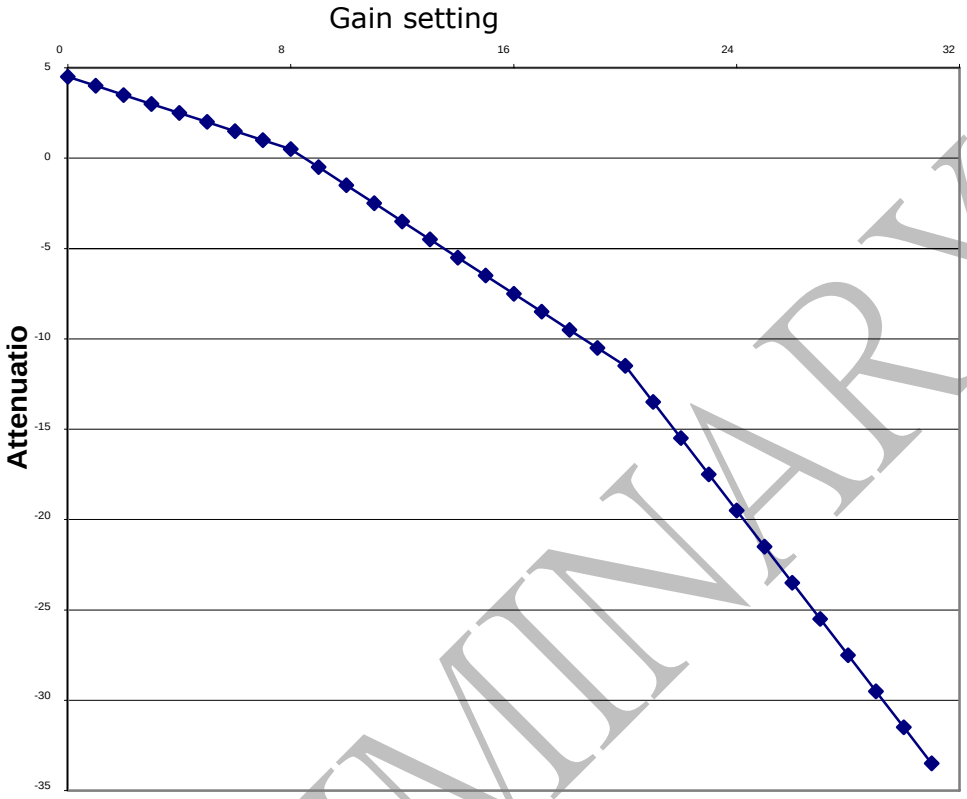
The following rules must be respected in order not to damage performances and to keep the functionality:

- If SB_IN1 is set to 1, INSEL must be different of '00' and BYPASS1 must be equal to 0.
- If SB_IN2 is set to 1, INSEL must be different of '01' and BYPASS2 must be equal to 0.
- If SB_MIC is set to 1, INSEL must be different of '10' and SIDETONE must be equal to 0.
- If SB_IND is set to 1, INSEL must be different of '11'.
- If SB_DAC is set to 1, DACSEL must be equal to 0.

29.5.9 Programmable attenuation

PGAT

The attenuation of PGAT may be programmed independently for the both channels through the registers bits GOL[4:0] and GOR[4:0]. The value of the gain is programmable from +4.5 to –33.5dB with a variable pitch as below:



The gain and output levels are obtained according to the following table:

GO[4]	GO[3]	GO[2]	GO[1]	GO[0]	Decimal decoded value	Gain Value (dB)	Maximal PGAT input amplitude (Vpp single-ended)	Maximal PGAT output amplitude (Vpp single-ended)
0	0	0	0	0	0	+4.5	0.425*VREF	0.71*VREF
0	0	0	0	1	1	+	0.451*VREF	0.71*VREF
						…		
0	0	1	1	1	7	+	0.637*VREF	0.71*VREF
0	1	0	0	0	8	+0.5	0.675*VREF	0.71*VREF
0	1	0	0	1	9	-0.5	0.757*VREF	0.71*VREF
						…		
1	0	0	1	1	19	-10.5	0.85*VREF	0.251*VREF
1	0	1	0	0	20	-11.5	0.85*VREF	0.225*VREF
1	0	1	0	1	21	-13.5	0.85*VREF	0.178*VREF
						…		
1	1	1	1	0	30	-31.5	0.85*VREF	0.023*VREF
1	1	1	1	1	31	-33.5	0.85*VREF	0.017*VREF

DAC path

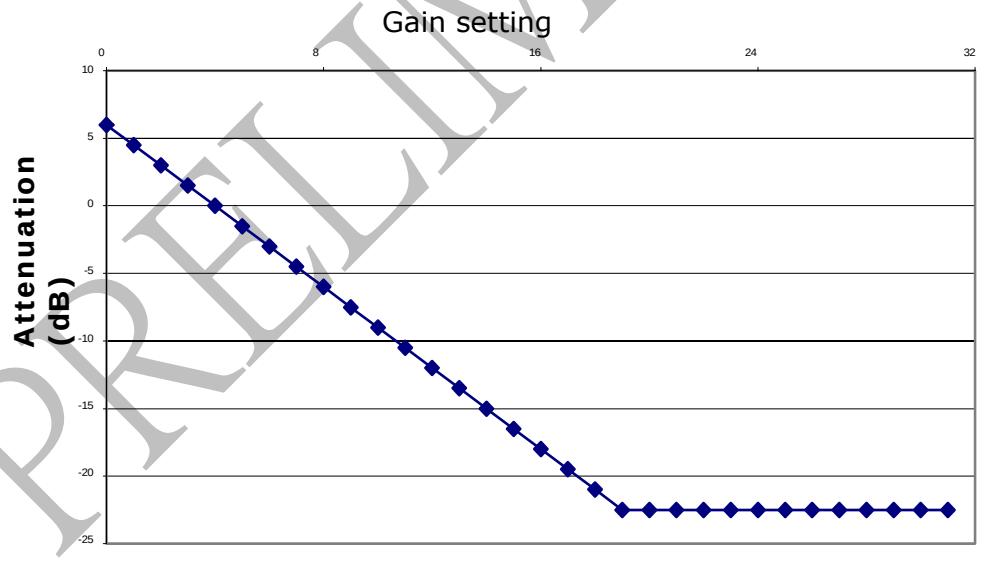
The attenuation of DAC path may be programmed independently for both channels through the pins GODL[3:0] and GODR[3:0]. The value of the gain is programmable from 0 to –22.5dB with a constant pitch as below.

GOD[3]	GOD[2]	GOD[1]	GOD[0]	Decimal decoded	Gain Value (dB)	Output amplitude (*) (Vpp single-ended)
0	0	0	0	0	0	0.85*VREF
0	0	0	1	1	-1.5	0.715*VREF
0	0	1	0	2	-	0.602* VREF
0	0	1	1	3	-4.5	0.506*VREF
				…		
x	y	z	t	i	-1.5*i	0.85/{10^(1.5*i/20)}*VRE
				…		
1	1	1	1	15	-22.5	0.064*VREF

(*) At output of Mixer, for maximal amplitude 0.85*VREF (Vpp single-ended) at input of Mixer on DAC path, and no signal on Bypass path

Bypass path, sidetone path

The attenuation of analog bypass and sidetone paths may be programmed independently for both channels through the pins GOBL1[4:0], GOBR1[4:0]; GOBL2[4:0], GOBR2[4:0]; GOSL[4:0] and GOSR[4:0]. The value of the gain is programmable from +6 to –22.5dB with a constant pitch as below.



G*[4]	G*[3]	G*[2]	G*[1]	G*[0]	Decimal decoded value	Gain Value (dB)	Maximal input amplitude (Vpp single-ended) Note (1)	Maximal output amplitude (Vpp single-ended) Note (2)
0	0	0	0	0	0	+6	0.425*VREF	0.85*VREF
0	0	0	0	1	1	+4.5	0.506*VREF	0.85*VREF
					…			
0	0	1	0	0	4	+0	0.85*VREF	0.85*VREF
0	0	1	0	1	5	-1.5	0.85*VREF	0.715*VREF

					…			
x	y	z	t	u	i	6-1.5*i	0.85*VREF	If i ≥4 0.85/{10^((6-1.5*i)/20)}*VREF
1	0	0	1	1	19	-22.5	0.85*VREF	0.064*VREF
					…	-22.5	0.85*VREF	0.064*VREF
1	1	1	1	1	31	-22.5	0.85*VREF	0.064*VREF

Note (1): at input of Mixer on analog Bypass or sidetone path, with no signal on DAC path
Note (2): at output of Mixer, with no signal on DAC path.

Appendix A – ARM7EJ-S Cache Controller

A.1 DESCRIPTION

The ARM7EJ-S cache controller is to provide a performance enhancement solution to ARM7EJ-S

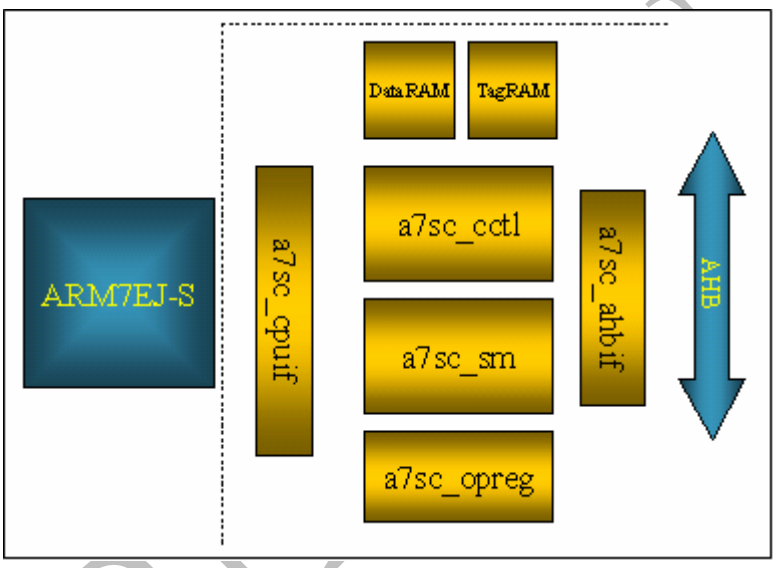
A.2 FEATURES

- ARM7EJ-S Interface.
- AMBA AHB 2.0 compatible interface.
- Unified ICACHE and DCACH
  - 2-Way, Configurable Cache Size 4KB/8KB/16KB/32KB
  - Write-through cache

  Pseudo-random replacement
  - Invalidate Single/Way
  - Cache Lockdown

  Performance counter (cache hit/miss)

A.3 BLOCK DIAGRAM



A.4 FUNCTION REGISTERS

A.4.1 Registers Summary:

Name	Offset	Reset Value	Access	Description
DevID	0x00	0x09500250	R/W	Some read only fields might be different in different configuration. Refer to the Detail register description for the detail.
CacheOp	0x04	0x00000000	R/W	
CacheLKDN	0x08	0x00000000	R/W	Cache Lock Down Register
MemMapA	0x10	0x00000000	R/W	
MemMapB	0x14	0x00000000	R/W	
MemMapC	0x18	0x00000000	R/W	
MemMapD	0x1C	0x00000000	R/W	
PFCNTRA_CTRL	0x20			

PFCNTRA	0x24			
PFCNTRB_CTRL	0x28			
PFCNTRA	0x2C			

A.4.2 Detail Register Description:

DevID

Basic device status/info register

Bit #	Access	Reset	Description
31	R/W	1'b0	Global Cache Enable
30:28	RO	3'b0	Reserved
27:16	RO	8h11	Design revision. Reset value depends on the deign revision.
15:10	RO	6'b0	Reserved
9:8	RO	2'b10	Configurable Cache Size
7:5	RO	3'b010	2-Way associative
4	RO	1'b1	8-word cache line size
3:0	RO	4'b0	Reserved

CacheOp

Control cache operations

Bit #	Access	Reset	Description
31:2	R/W		Look-up address for cache op.
1:0	R/W	2'b00	Cache Opcode. 2'b00:Nop, 2'b01:Invalidate Single Entry, 2'b10: Invalidate Way, 2'b11:Reserved. The Opcode filed is automatically cleared to 2'b00 after the invalidate process is complete.

CacheLKDN

Cache Lock-down controll

Bit #	Access	Reset	Description
31:2	R		Reserved
1:0	R/W	2'b00	Lock way selection. 2'b00: No lock, 2'b01:Lock Way0, 2'10:Lock Way1, 2'b11: Reserved

PFCNTR_CTRLA/B

Performance counter registers

Bit #	Access	Reset	Description
31:8	R		Reserved
7:4	R/W	4'b0	Count event. Below lists the countable events: "0": Cycle Counts: Counts every clock "1": Inst Counts: Counts instruction access "2": Cacheable Inst Counts: Counts cacheable instruction "3": Inst Hit Counts: Counts cache hit instructions "4": RESERVED "5": Data Counts: Counts data access "6": Data Write Counts: Counts data write access "7": Cacheable Data Counts: Counts cacheable data access "8": Data Read Hit Counts: Counts cache hit of data access "9" RESERVED
3:1	RO	3'b0	Reserved
0	R/W	1'b0	CNTR Enable. Set "1" to enable counterA/B to start count the specified event.

MemMap A/B/C/D

Setup region for uncacheable area

The base must be aligned to 32MB boundary. The region size must be 32MB/64MB/128MB

Bit #	Access	Reset	Description
31:25	R/W	7'b0	Memory map base
7:0	R/W	8'b0	Map Mask(Size) 8b0000_0000: Invalid setup 8b1111_1110: 32MB 8b1111_1100: 64MB 8b1111_1000: 128MB Others: Reserved

PFCNTRA/B

Performance Counter Value

Bit #	Access	Reset	Description
31:0	R/W	32'b0	Counter value

A.4.2 configurable cache size

Use CACHESIZE to set Cache Size: 4KB (00), 8KB (01), 16KB(10) , 16KB(11)

A.5 OPERATIONS

Cache operations

Invalidate Cache Entry

Users could invalidate specified cache entry via ***CacheOp*** register. Set OpCode in bit[1:0] with 2'b01, and fill the specified address in bit[31:2], the cache controller search the matched address in Tag ram and set the entry as invalidate. The user must poll the bit[1:0] in ***CacheOp*** register to be 2'b00 to make sure the invalidate process is complete.

Invalidate Cache Way

Users could also invalidate whole cache way by setting bit[1:0] in ***CacheOp*** register to 2'b10, and fill the specified way in bit[31]. The cache controller then invalidates all the entries in the specified way. Users MUST poll the bit[1:0] in ***CacheOp*** register to be 2'b00 to make sure the invalidate process is complete.

Cache lock-down

User could set the bit[1:0] in ***CacheLKDN*** register to select the way to lock. When one way is set to be lock, all cache update is going to another way. For example, if the way0 is set to be locked, all following cache refill is going to way1. The content in way0 will not be replaced until lock is cleared.

Cache Enable/Disable

Set the bit31 in ***DevID*** register 1'b1/1'b0 to enable/disable the cache controller.

Memory Mapping Setup

The ARM7EJ cache controller provides 4 memory map registers for users to setup uncacheable memory regions. The uncacheable area can be set from 32MB (1x32MB) to 512MB (4x128MB). All accesses from ARM7EJ located within the setup area are uncachable and read or write directly to AHB bus. Users can use these registers to setup the peripheral I/O mapping in system to ensure all I/O access are not cached. All region must be 32MB aligned.

Performance Counter

The ARM7EJ cache controller provides two optional performance counters for user to evaluate the cache efficiency. The counter counts the specified event when the counter is enabled. Two counters can work independently. Due to the design nature, for some events, like data counts, the counter value will be increased while the user access the register. These counters are implanted optionally, please contact Rockchips for the availability.