RK28XX
Technical Reference Manual

Revision 0.1

Rockchip

Feb 2009

Revision History

Date	Revision	Description
2009-02-18	0.1	Frist release
2009-03-20	0.2	Host interface bypass to lcdc modified Modify error for Pull Up or Down register set Modify Chapter 31 GPIO Pull Up or Pull Down setting Modify LCDC Port MUX List Add RK2808 description

TABLE OF CONTENT

Figure Index

Table Index

Acronym descriptions

CXCLK	clock for DSP Core
XHCLK	clock for AHB bus inside DSP System
XPCLK	clock for APB bus inside DSP System
R	Read only
RW	Capable of both read and write
R/W	Capable of both read and write

Chapter 1 Introduction

1.1 Overview

RK28xx is a highly-integrated, high-performance, low-power digital multimedia processor which is based on Dual Core (DSP+CPU) architecture with hardware accelerator. It is designed for high-end multimedia product applications such as PMP, GPS and Mobile TV etc.

RK28xx can support decode and encode for various types of video standards such as H.264/RMVB/MPEG-4/AVS/WMV9 by software and dedicated coprocessors. Specially, highest performace for video decode will reach fluent replay for video with H.264 @ 1280x720 formats. Meanwhile, RK28xx can also support many types of mobile TV standards by software and dedicated hardware accelerators, such as DVB-T, CMMB, T-DMB etc. By providing a complete set of peripheral interface, RK28xx can support very flexible applications , including SDRAM, Nor Flash, Nand Flash, LCDC , Sensor, USB OTG , SD/MMC , Wi-Fi , High-speed ADC , I2C, I2S , UART , SPI , PWM etc.

This document will provide guideline on how to use RK28xx correctly and efficiently. In them, the chapter 1 and chapter 2 will introduce the features, block diagram, and signal descriptions and system configuration of RK28xx, the chapter 3 through chapter 37 will describe the full function of each module in detail.

1.2 Features

- **System Operation**
  - Dual Core Architecture (ARM926EJC + DSP) , including hardware accelerator
  - Support system boot sequentially from ARM to DSP
  - Support address remap function
  - For two cores, all modules have unified address space
  - Selectable JTAG debug method
    - ARM9 debug only (default)
    - DSP debug only
    - ARM9+DSP dual core debug
  - Selectable booting method
    - Boot from NOR Flash
    - Boot from Embedded ROM (default)

- **Memory Organization**
  - Internal memory space for ARM processor
    - Internal 16KB SRAM for ARM9 ICache
    - Internal 16KB SRAM for ARM9 DCache
    - Internal 8KB SRAM for ARM9 ITCM
    - Internal 16KB SRAM for ARM9 DTCM
  - Internal memory space for DSP processor
    - Internal 64KB SRAM for DSP Instruction L1 Memory
      (Also config as 32KB Memory+32KB ICache by software)
    - Internal 64KB SRAM for DSP Data L1 Memory
    - Internal 64KB SRAM for DSP Instruction L2 Memory
    - Internal 64KB SRAM for DSP Data L2 Memory
  - Embedded 8KB ROM for ARM9 Boot
  - Embedded 2KB SRAM for communication between two cores
  - Embedded 90KB SRAM for share among ARM,DSP

- **Communication between two cores**
  - Support share memory and interactive interrupt method to complete communication

- Processor Interface Unit (PIU)
  - Built-in three Command/reply protocols registers and three Semaphore registers to accessed by two cores
  - Support three semaphore-related interrupts and one command-reply-related interrupt between two cores

- **Clock & Power Management**
  - Three on-chip PLLs for ARM9 subsystem, DSP subsystem and Other logic
  - Support different DSP Core and internal AHB Bus clock ratio :
    1:1, 1:2, 1:3, 1:4, up to 1:16 mode
  - Support different DSP internal AHB Bus and internal APB Bus clock ratio:
    1:1, 1:2, 1:3, 1:4, up to 1:16 mode
  - Support different ARM9 core and AHB Bus clock ratio:
    1:1, 1:2, 1:3 and 1:4 mode
  - Support different ARM AHB Bus and ARM APB Bus clock ratio:
    1:1, 1:2 and 1:4 mode
  - Max frequency of every key clock domain
    - 400MHz Max frequency for DSP Core
    - 350MHz Max frequency for ARM Core
  - 6 types of work modes by clock gating to save power :
    - Normal mode : Normal operating mode
    - Slow mode : Low frequency clock (24MHz) without PLL
    - Deep Slow mode : More Low frequency clock (32.768KHz) without PLL
    - Idle mode : The clock for only CPU is stopped ,
      Wake up by any interrupts to CPU from idle mode
    - Sleep mode : The clock for only DSP is stopped ,
      Wake up from sleep mode by some interrupts to DSP or register set from CPU
    - Stop mode : All clocks will be stopped , and SDRAM into Self-refresh, all PLLs into power-down mode,
      Wake up from stop mode by external pin or RTC alarm interrupt
  - Support power supply shut down for 4 domain separately

- **Memory Interface**
  - Static/SDRAM Memory controller
    - Dynamic memory interface support , including SDR-SDRAM and Mobile SDRAM
    - Asynchronous static memory device support including SRAM, ROM and Nor Flash with or without asynchronous page mode
    - Support 1 chip selects for (Mobile) SDRAM and 2 chip selects for static memory
    - Support 32bits data bus (Mobile) SDRAM and 8/16 bits data bus static memory
    - Support industrial standard (Mobile) SDRAM from 16MB to 128 MB devices
    - 4Mbytes access space per static memory support
    - Support (Mobile) SDRAM and Static Memory power-down mode
    - Support (Mobile) SDRAM self-refresh mode
    - Programmable arbitration priority for 5 slave data ports
  - Nand Flash controller
    - Support 4 chip selects for nand flash
    - support 8bits wide data
    - Flexible CPU interface support
    - Embedded 4x512B size buffer to improve performace
    - Support internal DMA transfer from/to flash
    - 512B、2KB、4KB page size support
    - Support hardware ECC

  - SD/MMC controller
    - Two Embedded SD/MMC Controllers, one is 4bit data bus , another is 8bit data bus
    - Compliant with SD Memory/SDIO with 1bit and 4bit data bus
    - Compliant with MMC V3.3 and V4.0 with 1/4/8bit data bus
    - Support combined single 32x32bits FIFO for both transmit and receive operations
    - Support FIFO over-run and under-run prevention by stopping card clock
    - Variable SD/MMC card clock rate 0 – 52 MHz which depends on AHB clock frequency
    - Controllable SD/MMC card clock to save power consumption
    - Support card detection and initialization , and write protection
    - Support transfer block size of 1 to 65365Bytes
    - DMA based or Interrupt based operation

- **VIDEO interface**
  - 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
    - Programmable Hsync and Vsync porality
    - Support 8 MegaPixels
  - LCD controller
    - Embedded DMA function
    - Support one SCALE window and one no SCALE window
    - YUV422/YUV420/RGB565/RGB888 Input are Supported in SCALE window
    - RGB565/RGB888 Input and 4 AREAS are Supported in NO SCALE window
    - Support Virtual Display
    - Build in scaler engine from 1/8 to 8
    - Support 16 grade alpha blending and transparent operation.
    - Support Blank/Black Function
    - Support LCD Pannel resolution up to 1280x760
    - Compatible with MCU Pannel
    - Support MCU PANNEL Bypass Mode and SCALE Mode
    - Compatible with RGB Delta/no-Delta Pannel
    - Compatible with RGB Series/Parallel 24bits (max) Output
    - Compatible with CCIR656 output
    - Support Interlace and Progressive Output
    - Support LCDC high-z control
    - Support LCDC interface bypass from Host interface

- **DMA Controller**
  - Two DMA Controllers in chip
  - DW_DMA Controller integrated inside ARM9 subsystem
    - Three DMA Channels support to use by audio , sd/mmc and system data transfer
    - 8 hardware request handshaking support
    - Support hardware and software trigger DMA transfer mode
    - Build-in 3 data FIFO : 64Bytes/32Bytes/16Bytes
    - Scatter/Gather transfer support
    - LLP transfer support
    - Two masters for on-the-fly support
    - The master interface only support undefined length INCR transfer
  - 3D-DMA Controller(XDMA) integrated inside DSP subsystem
    - This DMA focus on data transfer for video process and mobile TV

application
    - 16 configurable DMA channels, 4 channels support 3-dimensional data transfer
    - 8/16/32/64bit data transfer support and configurable burst length (INCR/INCR4/INCR8)
    - Programmable source and destination addresses with a post-modification option
    - Configurable external channel triggering (edge or level)
    - Support chaining-channels ,linked list-transfer and auto-channel initialization operating mode
    - Pause and resume operations supported to save power
    - Eight-stage memory buffer FIFO

- **Interrupt Controller**
  - Two Interrupt Controller in chip
  - DW_INTC integrated inside ARM9 subsystem
    - Support 32 IRQ normal interrupt sources and 4 FIQ fast interrupt sources
    - Vectored interrupts support
    - Software interrupts support
    - Programmable interrupt priorities
    - Programmable High/Low Level sensitive or Negative / Positive edge triggered interrupts
  - ICU (Interrupt Control unit) integrated inside DSP subsystem
    - 48 interrupt sources , each may be linked to different interrupt inputs for DSP core
    - Software triggering to all 48 interrupt sources
    - Configurable source interrupt polarity (low/high)
    - External interrupt source with software configuration to edge/level sensitive

- **USB interface**
  - Complies with the OTG Supplement to the USB2.0 Specification
  - Operates in High-Speed and Full-Speed mode
  - Support Session Request Protocol(SRP) and Host Negotiation Protocol(HNP)
  - Support 6 channels in host mode
  - 6 endpoints , 3 in and 3 out
  - Built-in one 1777 x 35bits FIFO

- **HOST interface**
  - 16 bits data bus for data transfer
  - 2KB internal Dual Port SRAM buffer
  - Interrupt request for data exchange
  - Support Host interface function disable
  - Support address self-increment when accessing buffer by MCU interface

- **Low_speed Peripheral interface**
  - Serial Peripheral Interface (SPI) Master Controller
    - Support two slave devices connection
    - Compatible with Motorola SPI , TI Synchronous Serial Protocol or National Semiconductor Microwire interface
    - Dynamic control of serial bit rate of data transfer by programmable sclk_out frequency, which is half of PCLK in max mode
    - FIFO depth for transmit and receive are also 16x16bits
    - Programmable data item size ,from 4 to 16bits
    - DMA based and interrupt based operation
  - Serial Peripheral Interface (SPI) Slave Controller
    - Compatible with Motorola SPI , TI Synchronous Serial Protocol or National Semiconductor Microwire interface

  - Dynamic control of serial bit rate of data transfer by sclk_in from master device
  - FIFO depth for transmit and receive are also 16x16bits
  - Programmable data item size ,from 4 to 16bits
  - DMA based and interrupt based operation
- UART0
  - Based on the 16550 industry standard
  - UART0 support modem function and Serial data transfer
  - Programmable serial data baud rate , up to 1.5Mbps
  - DMA based and interrupt based operation
  - FIFO depth for data transfer is 32x8bits
- UART1
  - Based on the 16550 industry standard
  - UART1 support IrDA 1.0 SIR mode and Serial data transfer
  - Programmable serial data baud rate , up to 1.5Mbps
  - In IrDA SIR mode, support configurable baud data rate up to 115.2K and a pulse duration as specified in the IrDA physical layer specification
  - DMA based and interrupt based operation
  - FIFO depth for data transfer is 32x8bits
- I2C controller
  - 2 I2C controllers integrated in chip
  - Multi masters operation support
  - Software programmable clock frequency and transfer rate up to 100Kbit/s in standard mode or up to 400Kbit/s in Fast mode
  - 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 32KHz to 96 KHz
  - Support I2S, Left-Justified and Right-Justified digital serial data format
- 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 96 individually programmable input/output pins
  - 16 GPIOs with external interrupt capability
- Timers in CPU system
  - Built-in Three 32 bits timer modules
  - Support for two operation modes : free-running and user-defined count
- Timers in DSP system
  - Built-in two 32 bits timer modules
  - Support for 5 various counting modes : Single Count mode, Auto-restart mode , Free-running , Event Count mode and Watchdog Timer mode
  - Pulse Width Modulation(PWM) mechanism
  - Three possible input clock signals: internal , external and cascaded
- Watchdog Timer (WDT)
  - Watchdog function (Generate a system reset or an interrupt)
  - Built-in 32 bits programmable 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
  - No off-chip components required
  - DNL less than +/-1 LSB , INL less than +/-1.5 LSB
  - Supply 2.8V to 3.6V for analog interface
- eFuse
  - 64-bit serial eFuse macro
  - Be programmed one bit at a time, but all 64bits can be read at the same time.
  - 2.9V (+/-200mV) & 2.5V(+/-50mV) Programm voltage

- **Operation Temperature Range**
  - TBD

- **Operation Voltage Range**
  - Core: 1.2V
  - I/O : 3.3V/2.5V/1.8V (2.5V for USB OTG PHY, 1.8V for Mobile SDRAM)

- **Package Type**
  - RK2806 BGA256 (14mmX14mm body size), for PMP application
  - Other package type TBD

- **Power**
  - TBD

1.3 Block Diagram

The following figure shows block diagram of RK28xx.
RK28xx can be divided into two sub system : DSP System and CPU System.

- **DSP System**
  - XDMA : three-dimensional DMA , used to data transfer for video decoder or other algorithm
  - High-Speed ADC Interface : focus on completing data reveiver from tuner in DVB-T,DAB, T-DMB,GPS application with software method.
  - ICU : Interrupt controller for DSP processor
  - PIU : processor interface unit, used to complete communication between DSP and CPU
  - PMU ：power management unit, used to control clock and reset to save power for modules inside DSP system
  - General reg file : focus on general control on DSP system by software method, composed of some register groups
  - Share Memx : can be accessed by DSP , CPU or Demodulator, which is switched by software programm
- **CPU System**
  - DW_DMA : used to data transfer for audio and low-speed peripheral
  - SCU : focus on clock gating , clock frequency switch, reset control , power on/off and system mode switch for CPU system to save power
  - PMU : used to complete power on/off switch control for RK28xx
  - INTC : Interrupt controller for CPU processor
  - General reg file : focus on general control on CPU system by software method, composed of some register groups, including IO mux control,IO PAD pull up/down control and other system control signals .

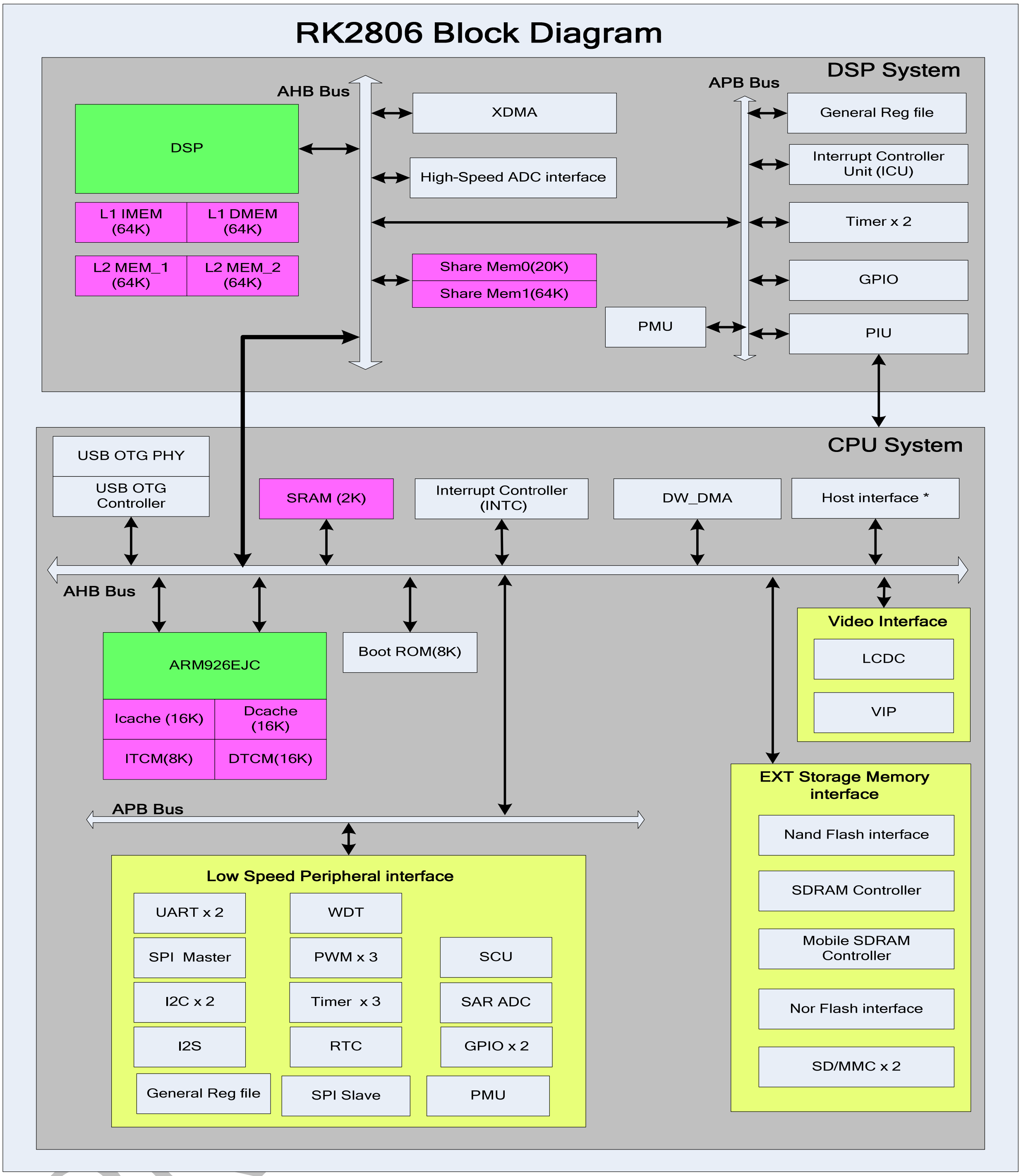


Fig. 1-1 RK28xx Block Diagram

Note:
*: RK2806 have no build in host interface

Chapter 2 Pin Description

2.1 RK2806 PIN Placement

Table 2-1 RK2806 Pin Placement

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	LCDC_HSYNC	LCDC_DCLK	TRST_N	TDI	VIP_DIN[0]	TDO	VIP_DIN[4]	VIP_VSYNC	GPIO_F[5]/PWM3	GPIO_E[3]/SPI_RXD	HSADC_D[2]/TS_D[2]	HSADC_D[4]/TS_D[4]	HSADC_D[5]/TS_D[5]	GPIO_B[1]/ST_CS1/SD0_PCA	RECOVER	I2S_SCLK	A
B	GPIO_C[1]/LCDC_D[17]	LCDC_DE	LCDC_VSYNC	TMS	VIP_DIN[1]	RTCK	VIP_DIN[6]	VIP_CLKIN	GPIO_F[2]/PWM0	EWAKEUP_STOP	HSADC_D[3]/TS_D[3]	HSADC_D[6]/TS_D[6]	GPS_CLK/HSADC_CLK	GPIO_B[3]/UART0_RTS	I2S_LRCK	I2S_SDI	B
C	GPIO_D[6]/LCDC_D[14]	GPIO_D[5]/LCDC_D[13]	GPIO_C[0]/LCDC_D[16]	GPIO_D[7]/LCDC_D[15]	TCK	VIP_DIN[2]	VIP_DIN[7]	VIP_HREF	GPIO_F[7]/SPI_TXD	EWAKEUP_POWER	HSADC_D[7]/TS_D[7]	HSADC_D[8]/TS_D[8]	GPIO_B[2]/UART0_CTS	I2S_SDO	I2C0_SDA/GPIO_E[4]	GPIO_B[0]/SPI0_CS1/SD1_PCA	C
D	GPIO_C[6]/LCDC_D[22]	GPIO_C[5]/LCDC_D[21]	GPIO_D[4]/LCDC_D[12]	GPIO_C[7]/LCDC_D[23]	LCDC_D[7]	VIP_DIN[3]	GPIO_F[6]/VIP_CLKOUT	NPOR	GPIO_E[1]/SPI_CLKIN	HSADC_D[0]/TS_D[0]	HSADC_D[9]/TS_FAIL	GPIO_E[7]/UART1_SIR_OUT/I2C1_SCL	I2S_CLK	I2C0_SCL/GPIO_E[5]	GPIO_G[6]/SD1_D[3]	GPIO_G[5]/SD1_D[2]	D
E	GPIO_D[0]/LCDC_D[8]	GPIO_D[3]/LCDC_D[11]	GPIO_D[2]/LCDC_D[10]	GPIO_D[1]/LCDC_D[9]	LCDC_D[6]	VIP_DIN[5]	EXTCLK	EXTMSDR_SEL	GPIO_E[2]/SPI_SS_IN	HSADC_D[1]/TS_D[1]	GPIO_E[6]/UART1_SIR_IN/I2C1_SDA	GPIO_F[0]/UART1_RX	GPIO_F[1]/UART1_TX	GPIO_B[6]/SPI0_TXD/SD0_D[6]	GPIO_G[3]/SD1_D[0]	XOUT24M	E
F	LCDC_D[1]	LCDC_D[3]	LCDC_D[2]	LCDC_D[5]	LCDC_D[4]	VCCIO	VDD	VCCIO	GND	VCCIO	VDD	GPIO_B[7]/SPI0_RXD/SD0_D[7]	GPIO_G[4]/SD1_D[1]	GPIO_G[2]/SD1_CMD	VSSA_CODECPLL	XIN24M	F
G	SDR_CLK	LCDC_D[0]	GPIO_C[3]/LCDC_D[19]	GPIO_C[2]/LCDC_D[18]	GPIO_C[4]/LCDC_D[20]	VDD	GND	GND	GND	GND	VCCIO	GPIO_G[0]/UART0_RX/SD1_DET	GPIO_G[7]/SD1_CLK	GPIO_G[1]/UART0_TX/SD1_WP	VSSA_ARMPLL	VDDA_CODECPLL	G
H	SDR_D[0]	SDR_D[4]	SDR_D[3]	SDR_D[2]	SDR_D[1]	VDDSDR	VSSSDR	GND	GND	GND	GND	GPIO_E[0]	GPIO_B[4]/SPI0_CS0/SD0_D[4]	GPIO_B[5]/SPI0_CLKO/SD0_D[5]	VSSA_DSPPLL	VDDA_ARMPLL	H
J	SDR_D[10]	SDR_D[8]	SDR_D[7]	SDR_D[6]	SDR_D[5]	VDD	GND	GND	GND	GND	VCCIO	GPIO_H[0]/SD0_CMD	GPIO_H[4]/SD0_D[3]	GPIO_F[3]/PWM1/SD0_DET	GPIO_H[5]/SD0_CLK	VDDA_DSPPLL	J
K	SDR_D[14]	SDR_D[13]	SDR_D[12]	SDR_D[11]	SDR_D[9]	VDDSDR	VSSSDR	VSSSDR	GND	GND	GND	VDD	GPIO_H[1]/SD0_D[0]	GPIO_H[2]/SD0_D[1]	GPIO_H[3]/SD0_D[2]	GPIO_F[4]/PWM2/SD0_WP	K
L	SDR_D[19]	SDR_D[18]	SDR_D[17]	SDR_D[16]	SDR_D[15]	VDD	VDDSDR	VDDSDR	VSSSDR	VDD	VCCIO	VGATE_EFUSE	FLASH_RDY	FLASH_CLE	FLASH_ALE	FLASH_WP	L
M	SDR_D[20]	SDR_D[22]	SDR_D[21]	SDR_D[23]	SDR_DQM[1]	SDR_DQM[2]	SDR_A[11]	SDR_RASN	VSS_RTC	DVDD_USB	DVSS_USB	FSOURCE_EFUSE	FLASH_RDN	FLASH_D[7]	GPIO_A[7]/FLASH_CS3	FLASH_WRN	M
N	SDR_D[24]	SDR_D[26]	SDR_D[25]	SDR_A[2]	SDR_A[4]	SDR_DQM[3]	SDR_A[12]	SDR_CKE	DVDD_33_RTC	ID	AVSS_USB	OTG_DRVVBUS	FLASH_D[0]	FLASH_D[4]	FLASH_CS0	GPIO_A[6]/FLASH_CS2	N
P	SDR_D[27]	SDR_D[28]	SDR_A[0]	SDR_A[3]	SDR_A[5]	SDR_A[8]	SDR_CSN	ST_OEN	PWR_GOOD	VBUS	AVDD25_USB	AVSS_USB	VSSA_ADC	FLASH_D[1]	FLASH_D[5]	GPIO_A[5]/FLASH_CS1	P
R	SDR_D[29]	SDR_D[30]	SDR_BA[0]	SDR_A[13]	SDR_A[6]	SDR_A[9]	SDR_WEN	ST_WEN	PWR_STROBE	XOUT32K	AVSS_USB	RKELVIN	ADC_AIN[1]	VDDA_ADC	FLASH_D[2]	FLASH_D[6]	R
T	SDR_D[31]	SDR_BA[1]	SDR_A[1]	SDR_DQM[0]	SDR_A[7]	SDR_A[10]	SDR_CASN	AVDD_RTC	RTCINT_OUT	XIN32K	AVDD33_USB	DP	DM	ADC_AIN[2]	ADC_AIN[0]	FLASH_D[3]	T
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

Note:

1. The different color means the different power supply domain.
2. Only the GPIO of port A and E can use as ARM interrupt port.

2.2 RK2806 PIN Description

The following table shows all of the pins for RK2806.

The first column in the pin function description is default function after power on reset, and function in the last two columns will be implemented by software set.

The detailed register descriptions are IOMUX_A_CON and IOMUX_B_CON in chapter 34. As for GPIO_n[i] (n = A~H; i = 0~7), we can control Pull up or Pull Down or no resistor for them by software set. The value for Pull up/down type in the following table is default after power on reset. The detailed register descriptions are in chapter 34.

Notes I --- Input pins
O --- Output pins
B --- Bidirectional pins
P --- Power supply pins (digital and analog)
G --- Ground supply pins (digital and analog)
A --- Analog IO pins
OSC --- Oscillator IO pins

Table 2-2 RK2806 Pin Description

PIN location	PIN Name	Pin Type (default)	Pull Up/Down (default)	Pin Function Description (default)	Pin Function Description (Function MUX1)	Pin Function Description (Function MUX2)
A1	LCDC_HSYNC	O	N/A	LCDC HORIZONTAL SYNC SIGNAL OUTPUT		
A2	LCDC_DCLK	O	N/A	LCDC DOT CLOCK OUT		
A3	TRST_N	I	Pull Down	JTAG TRST		
A4	TDI	I	Pull Up	JTAG TDI		
A5	VIP_DIN[0]	I	Pull Down	VIP DATA BIT0		
A6	TDO	O	N/A	JTAG TDO		
A7	VIP_DIN[4]	I	Pull Down	VIP DATA BIT4		
A8	VIP_VSYNC	I	Pull Down	VIP VERTICAL SYNC SIGNAL OUTPUT		
A9	GPIO_F[5]/PWM3	B	Pull Down	GPIO GROUP F BIT5	PWM3 OUT SIGNAL	
A10	GPIO_E[3]/SPI_RXD	B	Pull Down	GPIO GROUP E BIT3	SPI SLAVE RXD	
A11	HSADC_D[2]/TS_D[2]	I	Pull Down	HS-ADC I PATH DATA BIT2		
A12	HSADC_D[4]/TS_D[4]	I	Pull Down	HS-ADC I PATH DATA BIT4		
A13	HSADC_D[5]/TS_D[5]	I	Pull Down	HS-ADC I PATH DATA BIT5		
A14	GPIO_B[1]/ST_CS1/SD0_PCA	B	Pull Up	GPIO GROUP B BIT1	SRAM CHIP SELECT 1	SD/MMC0 POWER CONTROL
A15	RECOVER	B	Pull Up	RECOVER KEY INPUT, ENTER LOADER RECOVER MODE, LOW ACTIVE		
A16	I2S_SCLK	B	Pull Down	I2S SERIAL DATA CLOCK		
B1	GPIO_C[1]/LCDC_D[17]	B	Pull Up	GPIO GROUP C BIT1	LCDC DATA BIT17	
B2	LCDC_DE	B	Pull Down	GPIO2 BIT26	LCDC DATA ENABLE SIGNAL	
B3	LCDC_VSYNC	B	Pull Down	GPIO2 BIT25	LCDC VERTICAL SYNC SIGNAL OUTPUT	
B4	TMS	I	Pull Up	JTAG TMS		
B5	VIP_DIN[1]	I	Pull Down	VIP DATA BIT1		
B6	RTCK	O	N/A	JTAG RTCK		
B7	VIP_DIN[6]	I	Pull Down	VIP DATA BIT6		
B8	VIP_CLKIN	I	Pull Down	VIP CLOCK INPUT		
B9	GPIO_F[2]/PWM0	B	Pull Down	GPIO GROUP F BIT2	PWM0 OUT SIGNAL	
B10	EWAKEUP_STOP	I	Pull Down	EXTERNAL WAKEUP SIGNAL FROM STOP MODE		
B11	HSADC_D[3]/TS_D[3]	I	Pull Down	HS-ADC I PATH DATA BIT3		
B12	HSADC_D[6]/TS_D[6]	I	Pull Down	HS-ADC I PATH DATA BIT6		

B13	GPS_CLK/HSADC_CLK	B	Pull Down	GPIO2 BIT24	GPS CLOCK IN	HS-ADC CLOCK OUT
B14	GPIO_B[3]/UART0_RTS	B	Pull Up	GPIO GROUP B BIT3	UART0 MODEM RTS OUT	
B15	I2S_LRCK	B	Pull Down	I2S LRCK	GPIO2 BIT30	
B16	I2S_SDI	B	Pull Down	I2S INPUT DATA	GPIO2 BIT27	
C1	GPIO_D[6]/LCDC_D[14]	B	Pull Up	GPIO GROUP D BIT6	LCDC DATA BIT14	
C2	GPIO_D[5]/LCDC_D[13]	B	Pull Up	GPIO GROUP D BIT5	LCDC DATA BIT13	
C3	GPIO_C[0]/LCDC_D[16]	B	Pull Up	GPIO GROUP C BIT0	LCDC DATA BIT16	
C4	GPIO_D[7]/LCDC_D[15]	B	Pull Up	GPIO GROUP D BIT7	LCDC DATA BIT15	
C5	TCK	I	Pull Up	JTAG TCK		
C6	VIP_DIN[2]	I	Pull Down	VIP DATA BIT2		
C7	VIP_DIN[7]	I	Pull Down	VIP DATA BIT7		
C8	VIP_HREF	I	Pull Down	VIP HORIZONTAL SYNC SIGNAL		
C9	GPIO_F[7]/SPI_TXD	B	Pull Down	GPIO GROUP F BIT7	SPI SLAVE TXD	
C10	EWAKEUP_POWER	I	Pull Down	EXTERNAL WAKEUP SIGNAL FROM POWER OFF MODE		
C11	HSADC_D[7]/TS_D[7]	I	Pull Down	HS-ADC I PATH DATA BIT7		
C12	HSADC_D[8]/TS_D[8]	I	Pull Down	HS-ADC I PATH DATA BIT8		
C13	GPIO_B[2]/UART0_CTS	B	Pull Up	GPIO GROUP B BIT2	UART0 MODEM CTS IN	
C14	I2S_SDO	B	Pull Down	I2S OUTPUT DATA	GPIO2 BIT28	
C15	I2C0_SDA/GPIO_E[4]	B	Pull Up	I2C0 SDA	GPIO GROUP E BIT4	
C16	GPIO_B[0]/SPI0_CS1/SD1_PCA	B	Pull Up	GPIO GROUP B BIT0	SD/MMC1 POWER CONTROL	SPI MASTER CHIP SELECT 1
D1	GPIO_C[6]/LCDC_D[22]	B	Pull Up	GPIO GROUP C BIT6	LCDC DATA BIT22	
D2	GPIO_C[5]/LCDC_D[21]	B	Pull Up	GPIO GROUP C BIT5	LCDC DATA BIT21	
D3	GPIO_D[4]/LCDC_D[12]	B	Pull Up	GPIO GROUP D BIT4	LCDC DATA BIT12	
D4	GPIO_C[7]/LCDC_D[23]	B	Pull Up	GPIO GROUP C BIT7	LCDC DATA BIT23	
D5	LCDC_D[7]	O	N/A	LCDC DATA BIT7		
D6	VIP_DIN[3]	I	Pull Down	VIP DATA BIT3		
D7	GPIO_F[6]/VIP_CLKOUT	B	Pull Down	GPIO GROUP F BIT6	VIP CLOCK OUT	
D8	NPOR	I	N/A	POWER ON RESET, LOW ATIVE		
D9	GPIO_E[1]/SPI_CLKIN	B	Pull Down	GPIO GROUP E BIT1	SPI SLAVE CLOCK IN	
D10	HSADC_D[0]/TS_D[0]	I	Pull Down	HS-ADC I PATH DATA BIT0		
D11	HSADC_D[9]/TS_FAIL	I	Pull Down	HS-ADC I PATH DATA BIT9	TS STREAM FAIL INPUT	
D12	GPIO_E[7]/UART1_SIR_OUT/I2C1_SCL	B	Pull UP	GPIO GROUP E BIT7	UART1 IRDA DATA OUT	I2C1 SCL
D13	I2S_CLK	B	Pull Down	I2S MAIN CLOCK OUT	GPIO2 BIT29	
D14	I2C0_SCL/GPIO_E[5]	B	Pull UP	I2C0 SCL	GPIO GROUP E BIT5	
D15	GPIO_G[6]/SD1_D[3]	B	Pull Down	GPIO GROUP G BIT6	SD/MMC1 DATA BIT3	
D16	GPIO_G[5]/SD1_D[2]	B	Pull Down	GPIO GROUP G BIT5	SD/MMC1 DATA BIT2	
E1	GPIO_D[0]/LCDC_D[8]	B	Pull Up	GPIO GROUP D BIT0	LCDC DATA BIT8	
E2	GPIO_D[3]/LCDC_D[11]	B	Pull Up	GPIO GROUP D BIT3	LCDC DATA BIT11	
E3	GPIO_D[2]/LCDC_D[10]	B	Pull Up	GPIO GROUP D BIT2	LCDC DATA BIT10	
E4	GPIO_D[1]/LCDC_D[9]	B	Pull Up	GPIO GROUP D BIT1	LCDC DATA BIT9	
E5	LCDC_D[6]	O	N/A	LCDC DATA BIT6		
E6	VIP_DIN[5]	I	Pull Down	VIP DATA BIT5		
E7	EXTCLK	I	Pull Down	EXT CLOCK INPUT		
E8	EXTMSDR_SEL	I	Pull Down	MOBILE SDRAM SELECT		
E9	GPIO_E[2]/SPI_SS_IN	B	Pull Down	GPIO GROUP E BIT2	SPI SLAVE SELECT	
E10	HSADC_D[1]/TS_D[1]	I	Pull Down	HS-ADC I PATH DATA BIT1		
E11	GPIO_E[6]/UART1_SIR_IN/I2C1_SDA	B	Pull UP	GPIO GROUP E BIT6	UART1 IRDA DATA IN	I2C1 SDA

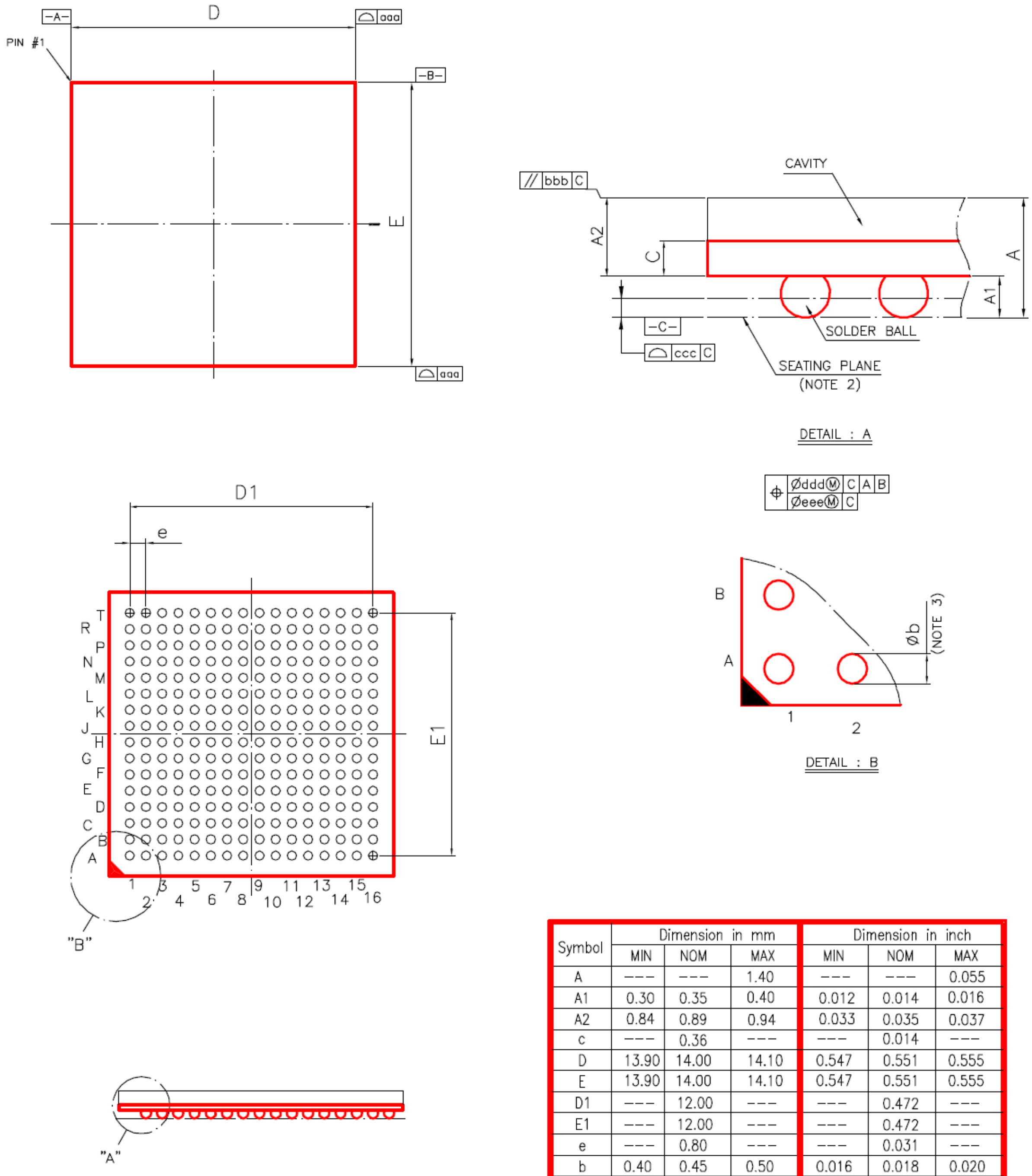
E12	GPIO_F[0]/UART1_RX	B	Pull Down	GPIO GROUP F BIT0	UART1 SERIAL DATA IN	DSP PWM0 OUT SIGNAL
E13	GPIO_F[1]/UART1_TX	B	Pull Down	GPIO GROUP F BIT1	UART1 SERIAL DATA OUT	DSP PWM1 OUT SIGNAL
E14	GPIO_B[6]/SPI0_TXD/SD0_D[6]	B	Pull Up	GPIO GROUP B BIT6	SPI MASTER TX DATA	SD/MMC0 DATA BIT6
E15	GPIO_G[3]/SD1_D[0]	B	Pull Down	GPIO GROUP G BIT3	SD/MMC1 DATA BIT0	
E16	XOUT24M	O OSC	N/A	CRYSTAL 24MHZ OUTPUT PIN		
F1	LCDC_D[1]	O	N/A	LCDC DATA BIT1		
F2	LCDC_D[3]	O	N/A	LCDC DATA BIT3		
F3	LCDC_D[2]	O	N/A	LCDC DATA BIT2		
F4	LCDC_D[5]	O	N/A	LCDC DATA BIT5		
F5	LCDC_D[4]	O	N/A	LCDC DATA BIT4		
F6	VCCIO	P	N/A	IO POWER (3.3 V)		
F7	VDD	P	N/A	CORE POWER (1.2V)		
F8	VCCIO	P	N/A	IO POWER (3.3 V)		
F9	GND	G	N/A	GROUND (0 V)		
F10	VCCIO	P	N/A	IO POWER (3.3 V)		
F11	VDD	P	N/A	CORE POWER (1.2V)		
F12	GPIO_B[7]/SPI0_RXD/SD0_D[7]	B	Pull Up	GPIO GROUP B BIT7	SPI MASTER RX DATA	SD/MMC0 DATA BIT7
F13	GPIO_G[4]/SD1_D[1]	B	Pull Down	GPIO GROUP G BIT4	SD/MMC1 DATA BIT1	
F14	GPIO_G[2]/SD1_CMD	B	Pull Down	GPIO GROUP G BIT2	SD/MMC1 COMMAND	
F15	VSSA_CODECPLL	G	N/A	CODEC PLL ANALOG GROUND(0V)		
F16	XIN24M	O OSC	N/A	CRYSTAL 24MHZ INPUT PIN		
G1	SDR_CLK	O	O	SDRAM CLOCK OUTPUT		
G2	LCDC_D[0]	O	N/A	LCDC DATA BIT0		
G3	GPIO_C[3]/LCDC_D[19]	B	Pull Up	GPIO GROUP C BIT3	LCDC DATA BIT19	
G4	GPIO_C[2]/LCDC_D[18]	B	Pull Up	GPIO GROUP C BIT2	LCDC DATA BIT18	
G5	GPIO_C[4]/LCDC_D[20]	B	Pull Up	GPIO GROUP C BIT4	LCDC DATA BIT20	
G6	VDD	P	N/A	CORE POWER (1.2V)		
G7	GND	G	N/A	GROUND (0 V)		
G8	GND	G	N/A	GROUND (0 V)		
G9	GND	G	N/A	GROUND (0 V)		
G10	GND	G	N/A	GROUND (0 V)		
G11	VCCIO	P	N/A	IO POWER (3.3 V)		
G12	GPIO_G[0]/UART0_RX/SD1_DET	B	Pull Down	GPIO GROUP G BIT0	UART0 SERIAL DATA IN	SD/MMC1 CARD DETECT
G13	GPIO_G[7]/SD1_CLK	B	Pull Down	GPIO GROUP G BIT7	SD/MMC1 CLOCK	
G14	GPIO_G[1]/UART0_TX/SD1_WP	B	Pull Down	GPIO GROUP G BIT1	UART0 SERIAL DATA OUT	SD/MMC1 WRITE PROTECT
G15	VSSA_ARMPLL	G	N/A	ARM PLL ANALOG GROUND(0V)		
G16	VDDA_CODECPLL	P	N/A	CODEC PLL ANALOG POWER(1.2V)		
H1	SDR_D[0]	B	N/A	SDRAM DATA BIT0		
H2	SDR_D[4]	B	N/A	SDRAM DATA BIT4		
H3	SDR_D[3]	B	N/A	SDRAM DATA BIT3		
H4	SDR_D[2]	B	N/A	SDRAM DATA BIT2		
H5	SDR_D[1]	B	N/A	SDRAM DATA BIT1		
H6	VDDSDR	P	N/A	SDRAM IO POWER(1.8V OR 3.3V)		
H7	VSSSDR	G	N/A	SDRAM IO GROUND(0V)		
H8	GND	G	N/A	GROUND (0 V)		
H9	GND	G	N/A	GROUND (0 V)		
H10	GND	G	N/A	GROUND (0 V)		

H11	GND	G	N/A	GROUND (0 V)		
H12	GPIO_E[0]	B	Pull Down	GPIO GROUP E BIT0		
H13	GPIO_B[4]/SPI0_CS0 /SD0_D[4]	B	Pull Up	GPIO GROUP B BIT4	SPI MASTER CHIP SELECT 0	SD/MMC0 DATA BIT4
H14	GPIO_B[5]/SPI0_CLK O/SD0_D[5]	B	Pull Up	GPIO GROUP B BIT5	SPI MASTER CLOCK	SD/MMC0 DATA BIT5
H15	VSSA_DSPPLL	G	N/A	DSP PLL ANALOG GROUND(0V)		
H16	VDDA_ARMPLL	P	N/A	ARM PLL ANALOG POWER(1.2V)		
J1	SDR_D[10]	B	N/A	SDRAM DATA BIT10		
J2	SDR_D[8]	B	N/A	SDRAM DATA BIT8		
J3	SDR_D[7]	B	N/A	SDRAM DATA BIT7		
J4	SDR_D[6]	B	N/A	SDRAM DATA BIT6		
J5	SDR_D[5]	B	N/A	SDRAM DATA BIT5		
J6	VDD	P	N/A	CORE POWER (1.2V)		
J7	GND	G	N/A	GROUND (0 V)		
J8	GND	G	N/A	GROUND (0 V)		
J9	GND	G	N/A	GROUND (0 V)		
J10	GND	G	N/A	GROUND (0 V)		
J11	VCCIO	P	N/A	IO POWER (3.3 V)		
J12	GPIO_H[0]/SD0_CMD	B	Pull Down	GPIO GROUP H BIT0	SD/MMC0 COMMAND	
J13	GPIO_H[4]/SD0_D[3]	B	Pull Down	GPIO GROUP H BIT4	SD/MMC0 DATA BIT3	
J14	GPIO_F[3]/PWM1/SD 0_DET	B	Pull Down	GPIO GROUP F BIT3	PWM1 OUT SIGNAL	SD/MMC0 CARD DETECT
J15	GPIO_H[5]/SD0_CLK	B	Pull Down	GPIO GROUP H BIT5	SD/MMC0 CLOCK OUT	
J16	VDDA_DSPPLL	P	N/A	DSP PLL ANALOG POWER(1.2V)		
K1	SDR_D[14]	B	N/A	SDRAM DATA BIT14		
K2	SDR_D[13]	B	N/A	SDRAM DATA BIT13		
K3	SDR_D[12]	B	N/A	SDRAM DATA BIT12		
K4	SDR_D[11]	B	N/A	SDRAM DATA BIT11		
K5	SDR_D[9]	B	N/A	SDRAM DATA BIT9		
K6	VDDSDR	P	N/A	SDRAM IO POWER(1.8V OR 3.3V)		
K7	VSSSDR	G	N/A	SDRAM IO GROUND(0V)		
K8	VSSSDR	G	N/A	SDRAM IO GROUND(0V)		
K9	GND	G	N/A	GROUND (0 V)		
K10	GND	G	N/A	GROUND (0 V)		
K11	GND	G	N/A	GROUND (0 V)		
K12	VDD	P	N/A	CORE POWER (1.2V)		
K13	GPIO_H[1]/SD0_D[0]	B	Pull Down	GPIO GROUP H BIT1	SD/MMC0 DATA BIT0	
K14	GPIO_H[2]/SD0_D[1]	B	Pull Down	GPIO GROUP H BIT2	SD/MMC0 DATA BIT1	
K15	GPIO_H[3]/SD0_D[2]	B	Pull Down	GPIO GROUP H BIT3	SD/MMC0 DATA BIT2	
K16	GPIO_F[4]/PWM2/SD 0_WP	B	Pull Down	GPIO GROUP F BIT4	PWM2 OUT SIGNAL	SD/MMC0 WRITE PROTECT
L1	SDR_D[19]	B	N/A	SDRAM DATA BIT19		
L2	SDR_D[18]	B	N/A	SDRAM DATA BIT18		
L3	SDR_D[17]	B	N/A	SDRAM DATA BIT17		
L4	SDR_D[16]	B	N/A	SDRAM DATA BIT16		
L5	SDR_D[15]	B	N/A	SDRAM DATA BIT15		
L6	VDD	P	N/A	CORE POWER (1.2V)		
L7	VDDSDR	P	N/A	SDRAM IO POWER(1.8V OR 3.3V)		
L8	VDDSDR	P	N/A	SDRAM IO POWER(1.8V OR 3.3V)		
L9	VSSSDR	G	N/A	SDRAM IO GROUND(0V)		

L10	VDD	P	N/A	CORE POWER (1.2V)		
L11	VCCIO	P	N/A	IO POWER (3.3 V)		
L12	VGATE_EFUSE	A	N/A	GATE POWER SUPPLY OF EFUSE PROGRAM , CONNECT TO VDD		
L13	FLASH_RDY	I	Pull Up	NAND FLASH READY/BUSY SINGAL INPUT		
L14	FLASH_CLE	O	N/A	NAND FLASH CLE		
L15	FLASH_ALE	O	N/A	NAND FLASH ALE		
L16	FLASH_WP	O	N/A	NAND FLASH WRITE PROTECT		
M1	SDR_D[20]	B	N/A	SDRAM DATA BIT20		
M2	SDR_D[22]	B	N/A	SDRAM DATA BIT22		
M3	SDR_D[21]	B	N/A	SDRAM DATA BIT21		
M4	SDR_D[23]	B	N/A	SDRAM DATA BIT23		
M5	SDR_DQM[1]	O	N/A	SDRAM DQM BIT1		
M6	SDR_DQM[2]	O	N/A	SDRAM DQM BIT2		
M7	SDR_A[11]	O	N/A	SDRAM/SRAM ADDR BIT11		
M8	SDR_RASN	O	N/A	SDRAM RASN		
M9	VSS_RTC	G	N/A	RTC GROUND (0V)		
M10	DVDD_USB	P	N/A	USB DIGITAL POWER SUPPLY (1.2V)		
M11	DVSS_USB	G	N/A	USB DIGITAL GROUND (0V)		
M12	FSOURCE_EFUSE	A	N/A	SOURCE POWER SUPPLY OF EFUSE PROGRAM, CONNECT TO GROUND		
M13	FLASH_RDN	O	N/A	NAND FLASH RDN		
M14	FLASH_D[7]	B	N/A	NAND FLASH DATA BIT7		
M15	GPIO_A[7]/FLASH_CS3	B	Pull Up	GPIO GROUP A BIT7	NAND FLASH CHIP SELECT 3	
M16	FLASH_WRN	O	N/A	NAND FLASH WRN		
N1	SDR_D[24]	B	N/A	SDRAM DATA BIT24		
N2	SDR_D[26]	B	N/A	SDRAM DATA BIT26		
N3	SDR_D[25]	B	N/A	SDRAM DATA BIT25		
N4	SDR_A[2]	O	N/A	SDRAM/SRAM ADDR BIT2		
N5	SDR_A[4]	O	N/A	SDRAM/SRAM ADDR BIT4		
N6	SDR_DQM[3]	O	N/A	SDRAM DQM BIT3		
N7	SDR_A[12]	O	N/A	SDRAM/SRAM ADDR BIT12		
N8	SDR_CKE	O	N/A	SDRAM CLOCK ENABLE		
N9	DVDD_33_RTC	P	N/A	RTC DIGITAL POWER SUPPLY (3.3V)		
N10	ID	I	N/A	USB MINNI-RECEPTABLE IDENTIFIER		
N11	AVSS_USB	G	N/A	USB ANALOG GROUND (0V)		
N12	OTG_DRVVBUS	O	N/A	USB DRIVE VBUS CONTROLE SIGNAL		
N13	FLASH_D[0]	B	N/A	NAND FLASH DATA BIT0		
N14	FLASH_D[4]	B	N/A	NAND FLASH DATA BIT4		
N15	FLASH_CS0	O	N/A	NAND FLASH CHIP SELECT 0		
N16	GPIO_A[6]/FLASH_CS2	B	Pull Up	GPIO GROUP A BIT6	NAND FLASH CHIP SELECT 2	
P1	SDR_D[27]	B	N/A	SDRAM DATA BIT27		
P2	SDR_D[28]	B	N/A	SDRAM DATA BIT28		
P3	SDR_A[0]	O	N/A	SDRAM/SRAM ADDR BIT0		
P4	SDR_A[3]	O	N/A	SDRAM/SRAM ADDR BIT3		
P5	SDR_A[5]	O	N/A	SDRAM/SRAM ADDR BIT5		
P6	SDR_A[8]	O	N/A	SDRAM/SRAM ADDR BIT8		
P7	SDR_CSN	O	N/A	SDRAM CHIP SELECT		

P8	ST_OEN	O	N/A	SRAM OEN		
P9	PWR_GOOD	I	Pull Up	RTC POWER GOOD INPUT		
P10	VBUS	P	N/A	USB 5V POWER SUPPLY		
P11	AVDD25_USB	P	N/A	USB ANALOG POWER SUPPLY (2.5V)		
P12	AVSS_USB	G	N/A	USB ANALOG GROUND (0V)		
P13	VSSA_ADC	G	N/A	10BIT ADC ANALOG GROUND (0V)		
P14	FLASH_D[1]	B	N/A	NAND FLASH DATA BIT1		
P15	FLASH_D[5]	B	N/A	NAND FLASH DATA BIT5		
P16	GPIO_A[5]/FLASH_CS1	B	Pull Up	GPIO GROUP A BIT5	NAND FLASH CHIP SELECT 1	
R1	SDR_D[29]	B	N/A	SDRAM DATA BIT29		
R2	SDR_D[30]	B	N/A	SDRAM DATA BIT30		
R3	SDR_BA[0]	O	N/A	SDRAM BAND ADDRESS BIT0		
R4	SDR_A[13]	O	N/A	SDRAM/SRAM ADDR BIT13		
R5	SDR_A[6]	O	N/A	SDRAM/SRAM ADDR BIT6		
R6	SDR_A[9]	O	N/A	SDRAM/SRAM ADDR BIT9		
R7	SDR_WEN	O	N/A	SDRAM WEN		
R8	ST_WEN	O	N/A	SRAM WEN		
R9	PWR_STROBE	I	Pull Up	RTC POWER STROBE INPUT		
R10	XOUT32K	O OSC	N/A	CRYSTAL 32KHZ OUTPUT PAD		
R11	AVSS_USB	G	N/A	USB ANALOG GROUND (0V)		
R12	RKELVIN	A	N/A	TRANSMITTER RESISTOR TUNE PIN		
R13	ADC_AIN[1]	A	N/A	10BIT ADC CHANNEL1 INPUT		
R14	VDDA_ADC	P	N/A	10BIT ADC ANALOG POWER AND REFERENCE VOLTAGE (3.3V)		
R15	FLASH_D[2]	B	N/A	NAND FLASH DATA BIT2		
R16	FLASH_D[6]	B	N/A	NAND FLASH DATA BIT6		
T1	SDR_D[31]	B	N/A	SDRAM DATA BIT31		
T2	SDR_BA[1]	O	N/A	SDRAM BAND ADDRESS BIT1		
T3	SDR_A[1]	O	N/A	SDRAM/SRAM ADDR BIT1		
T4	SDR_DQM[0]	O	N/A	SDRAM DQM BIT0		
T5	SDR_A[7]	O	N/A	SDRAM/SRAM ADDR BIT4		
T6	SDR_A[10]	O	N/A	SDRAM/SRAM ADDR BIT10		
T7	SDR_CASN	O	N/A	SDRAM CASN		
T8	AVDD_RTC	P	N/A	RTC ANALOG POWER (1.2V)		
T9	RTCINT_OUT	O	N/A	RTC INTERRUPT OUTPUT		
T10	XIN32K	I OSC	N/A	CRYSTAL 32KHZ INPUT PAD		
T11	AVDD33_USB	P	N/A	USB ANALOG POWER SUPPLY (3.3V)		
T12	DP	A	N/A	USB D+ SIGNAL		
T13	DM	A	N/A	USB D- SIGNAL		
T14	ADC_AIN[2]	A	N/A	10BIT ADC CHANNEL2 INPUT		
T15	ADC_AIN[0]	A	N/A	10BIT ADC CHANNEL0 INPUT		
T16	FLASH_D[3]	B	N/A	NAND FLASH DATA BIT3		

2.3 BGA256 package outline



Symbol	Dimension in mm			Dimension in inch		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.40	---	---	0.055
A1	0.30	0.35	0.40	0.012	0.014	0.016
A2	0.84	0.89	0.94	0.033	0.035	0.037
c	---	0.36	---	---	0.014	---
D	13.90	14.00	14.10	0.547	0.551	0.555
E	13.90	14.00	14.10	0.547	0.551	0.555
D1	---	12.00	---	---	0.472	---
E1	---	12.00	---	---	0.472	---
e	---	0.80	---	---	0.031	---
b	0.40	0.45	0.50	0.016	0.018	0.020
aaa	0.10			0.004		
bbb	0.20			0.008		
ccc	0.12			0.005		
ddd	0.15			0.006		
eee	0.08			0.003		
MD/ME	16/16			16/16		

Fig. 2-1 BGA256 package outline

Chapter 3 System Configuration

3.1 BUS Architecture

3.1.1 CPU system AHB Bus architecture

In CPU system, the bus matrix is composed of three AHB-Lite buses and two multi-masters buses. RK28xx will provide two arbiters on multi-masters bus named as EXP BUS and ARMD BUS. The default priority for every master is as the following table. In them, the priority of every master in ARMD BUS is programmable; however, the priority in EXP BUS is fixed.

Priority of Masters in ARMD BUS

Master No.	Priority	Master
4	Highest	AHB2AHB
3		DW_DMA Master1
2		Deblocking Master1
1	Lowest	ARM DATA

Priority of Masters in EXP BUS

Master No.	Priority	Master
4	Highest	Deblocking Master0
3		DW_DMA Master0
2		USB OTG Master
1	Lowest	VIP Master

3.1.2 DSP system AHB Bus architecture

DSP system is based on a multi-layer AHB-Lite bus architecture. There are totally 6 masters. And fixed priority arbitration will be used in slave layer when different masters will access a same slave at the same time. The priority for 6 masters is as follows.

Priority of Masters in DSP system

Priority	Master
Highest	EXT AHB Master (CPU System)
	XDMA Data Port0
	XDMA Data Port1
	XDMA Manager
	DSP Data Port
Lowest	DSP Instruction Port

3.1.3 Data Path description for CPU and DSP

The following list shows the valid acess path for CPU and DSP.

Table 3-1 Valid access path list for CPU and DSP

Device		CPU	DSP
CPU System	ITCM	V	X
	DTCM	V	X
	Boot ROM	V	V
	SRAM	V	V
	NAND Flash Interface	V	V
	Static/SDRAM Controller Register	V	V

	Port		
	Nor Flash0	V	V
	Nor Flash1	V	V
	SDRAM	V	V
	SD/MMC0	V	V
	SD/MMC1	V	V
	Host Interface	V	V
	USB OTG	V	V
	DW_DMA	V	V
	INTC	V	V
	LCDC	V	V
	VIP	V	V
	ARMD BUS Arbiter	V	V
	APB UART0	V	V
	APB UART1	V	V
	APB Timer0/1/2	V	V
	APB eFuse	V	V
	APB GPIOA-D	V	V
	APB GPIOE-H	V	V
	APB I2S	V	V
	APB I2C0	V	V
	APB I2C1	V	V
	APB SPI Master	V	V
	APB SPI Slave	V	V
	APB WDT	V	V
	APB PWM	V	V
	APB RTC	V	V
	APB SAR-ADC	V	V
	APB SCU	V	V
	APB Reg File	V	V
DSP System	L1 DMEM	V	V
	L2 MEM_1	V	V
	L2 MEM_2	V	V
	Share Mem0	V	V
	Share Mem1	V	V
	XDMA	X	V
	High-Speed ADC	X	V
	PMU	V	V
	ICU	V	V
	TIMER0	V	V
	TIMER1	V	V
	GPIO	V	V
	ASHB MST	V	V
	ASHB SLV	V	V
	PIU	V	V
	APB Reg file	V	V

3.1.4 CPU ITCM/DTCM Application Notes

As for ITCM/DTCM (Instruction tightly coupled memory and Data tightly coupled memory) , they are not devices in AHB bus , only accessed by CPU .They are always disabled at reset and can be accessed after enable them by software. As for the detailed information , you can refer to the Appendix A .

Pay more attention that ITCM/DTCM works in the same frequency as CPU , so it can get higher performance. In general , the cycle latency for read/write ITCM/DTCM of CPU is one cycle as illustration in Fig.2-4 and Fig.2-5. However, when CPU frequency is beyond 300MHz , reading operation will be inserted one wait cycle to meet timing requirement, which results in two cycles latency for reading operation as illustration in Fig.2-6.

Insertion one wait cycle will be completed by software set in bit 12 of CPU_APB_REG5. Refer to Chapter 34 (General Register File in CPU System) for detailed descriptions.

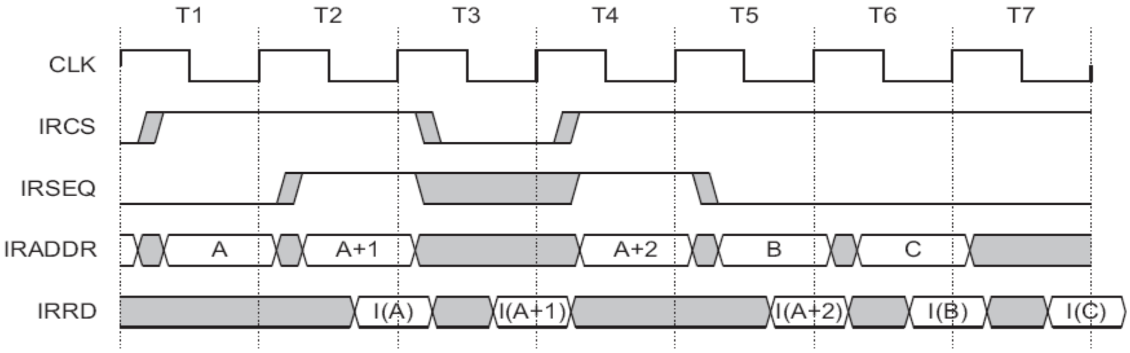


Fig. 3-1 ITCM access timing with zero wait state

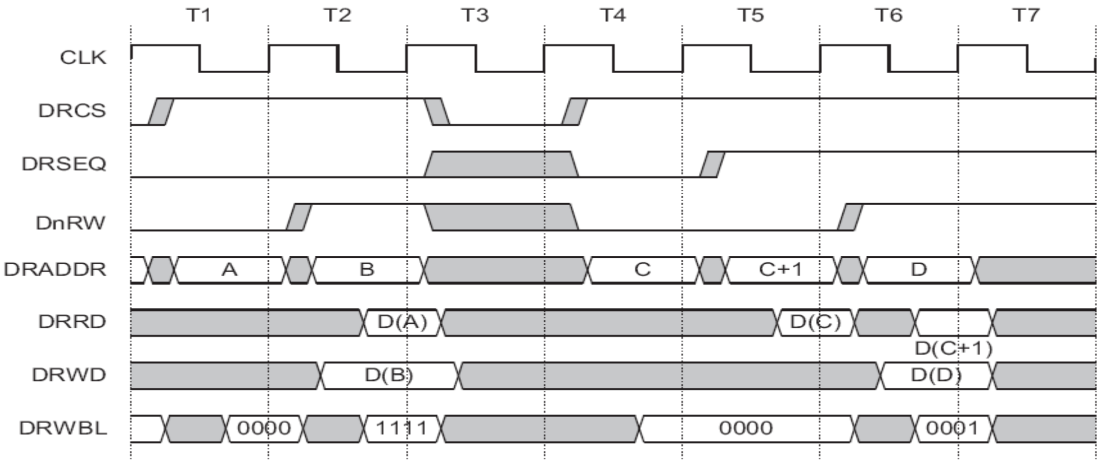


Fig. 3-2 DTCM access timing with zero wait state

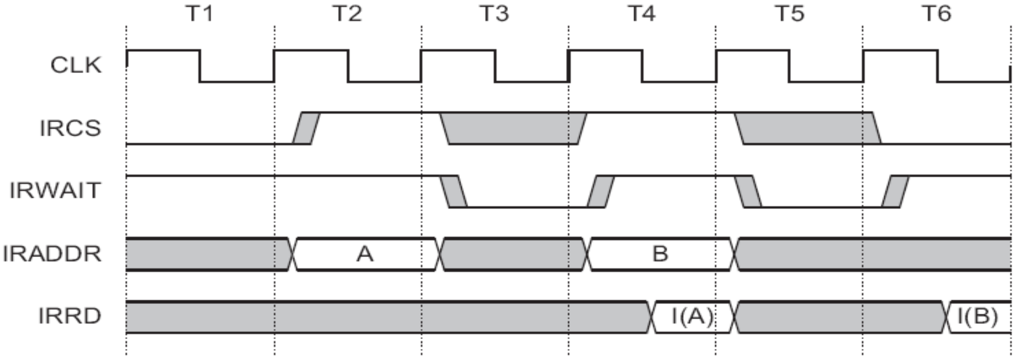


Fig. 3-3 ITCM/DTCM access timing with one wait state

3.1.5 SDRAM and Mobile SDRAM Interface Application Notes

External memory controller in RK28xx supports Static Memory,SDRAM and mobile SDRAM interface. There are one register slave port and five data slave ports. In other words, it can support simultaneous read/write for five masters in different bus . Refer to the following diagram.

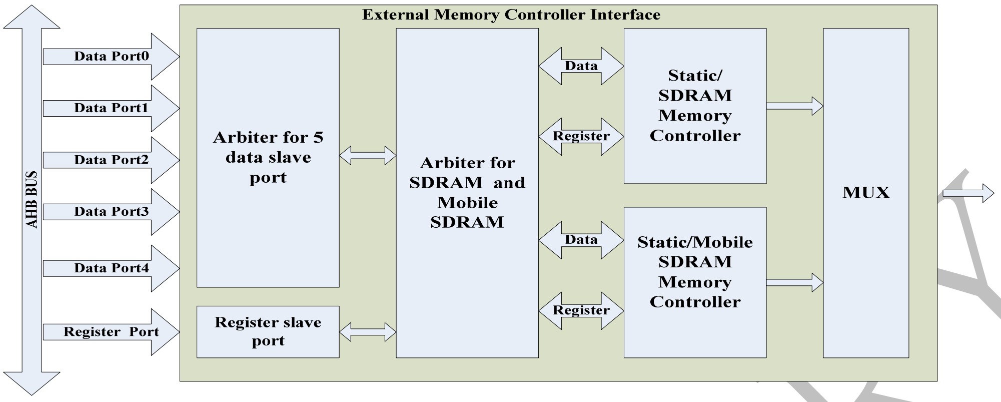

External Memory Controller Interface
AHB BUS
Data Port0
Data Port1
Data Port2
Data Port3
Data Port4
Register Port
Arbiter for 5 data slave port
Register slave port
Arbiter for SDRAM and Mobile SDRAM
Data
Register
Static/ SDRAM Memory Controller
Data
Register
Static/Mobile SDRAM Memory Controller
MUX

Fig. 3-4 External memory controller architecture

- **Switch for SDRAM and Mobile SDRAM**

The function switch for SDRAM or Mobile SDRAM will be programmable, which is bit 15 of CPU_APB_REG4. And bit 24 will set different IO voltage requirement for Mobile SDRAM. Please refer to Chapter 34 (General Register File in CPU System) for detailed descriptions. In RK28xx the default select is SDRAM after power on reset. Another, the function for static memory interface will be integrated both in SDRAM and Mobile SDRAM Controller.

- **Priority for five Data Ports**

In RK28xx the five data ports of external memory interface are separately connected to five-layer bus. The default priority for five data ports is as follows . And it is software programmable by bit[14:0] of CPU_APB_REG4 according to different requirement for every masters. Please refer to Chapter 34 (General Register File in CPU System) for detailed descriptions.

Priority of data ports for sdram / mobile sdram controller

Priority	Master	Port number
Highest	LCDC BUS	Data Port0
	EXP BUS	Data Port1
	ARMD BUS	Data Port2
	ARMI BUS	Data Port3
Lowest	DSP BUS	Data Port4

- **Read pipe set**

In RK28xx when SDRAM frequency is beyond 120MHz , in order to meet timing requirement and ensure the function , one read pipe register must be inserted in read data path, which is software programmable by bit 22 of CPU_APB_REG4 register. When bit 22 of CPU_APB_REG4 will be set to 1'b1 and enable read pipe function, you must set bit [8:6] to 3'b001 of register SDR_SCTLR to ensure correct read operation.

Notes: *As for detailed description of register CPU_APB_REG4 and SDR_SCTLR , please refer to Chapter 34 (General Register File in CPU System) and Chapter 3 (Static Memory/SDRAM controller) or Chapter 4 (Static Memory/Mobile SDRAM controller)*

In Chapter 3 and Chapter 4, the Static/SDRAM memory controller and Static/Moible SDRAM memory controller with single data port will be described separately in more detailed.

3.2 System Address Map

RK28xx has fixed address maps for on-chip memory or registers and off-chip peripheral, and the 32-bit address bus can address up to 4GB of memory.

For CPU and DSP, unified address space is used except IPs inside DSP system , which have different address map between CPU and DSP.

For CPU System, RK28xx will provide address decode re-map function. It can speed-up whole system performance. The detail memory map is as follows.

3.2.1 System Memory Map for CPU

The following list shows address space for every devices CPU can access.

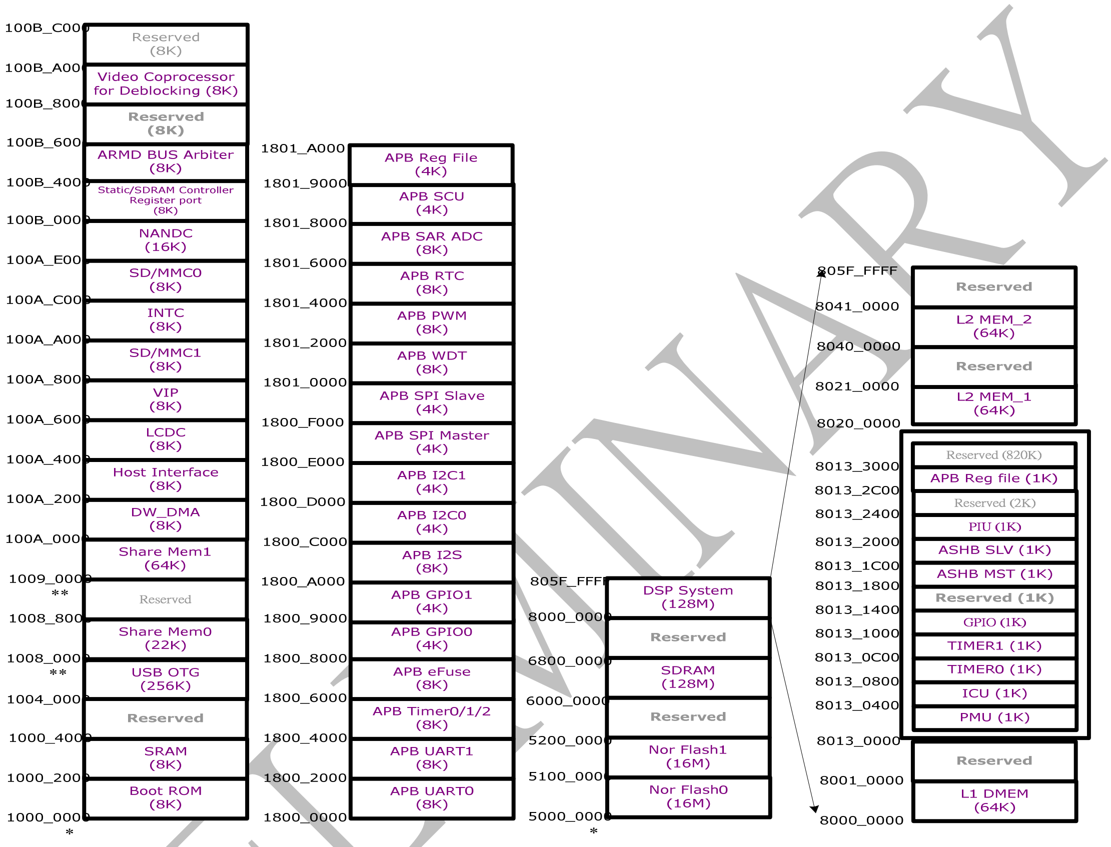


Fig. 3-5 System Memory Map for CPU

Notes:

** shows address space is after remap , really in default state (before remap) the address space for Boot ROM and Nor Flash0 are also 0x0000_0000. Refer to the next detailed description.*

*** Before CPU will access Share Mem0 and Share Mem1, some special bits in register CPU_APB_REG5 must be set. Refer to Chapter 34 (General Register File in CPU System) for detailed descriptions.*

RK28xx provides remap function as the following diagram. Before remap (power-on-reset state), the address space for Boot ROM and Nor Flash0 are also 0x0000_0000, which is the first instructioin PC value for CPU. And the value for one input pin (bt_mode) will decide that the zero address is for Boot Rom or for Nor Flash0. After remap, the address space for them will change to the real physical address. At this time the 0x0000_0000 address will assigned to ITCM/DTCM after ITCM/DTCM is enabled as the following Fig. 2-3, since ITCM/DTCM is in disable state initially.

Remap operation is software programmable by setting a special bit in register CPU_APB_REG5. Refer to detailed description in Chapter 34 (General Register File in CPU System).

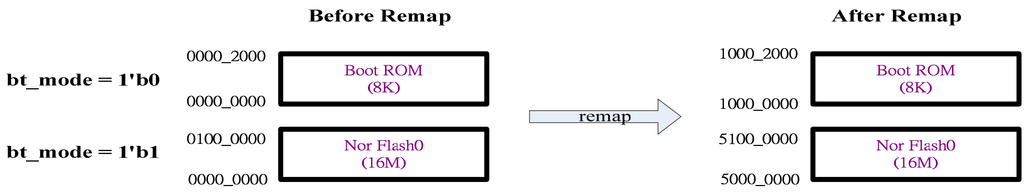


Fig. 3-6 Remap address description

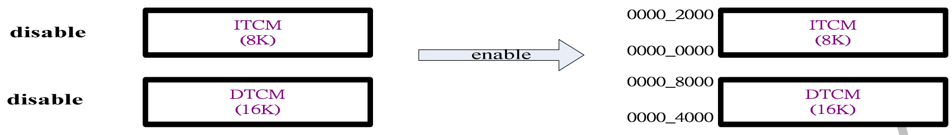


Fig. 3-7 ITCM/DTCM address map

3.2.2 System Memory Map for DSP

The following list shows address space for every device DSP can access.

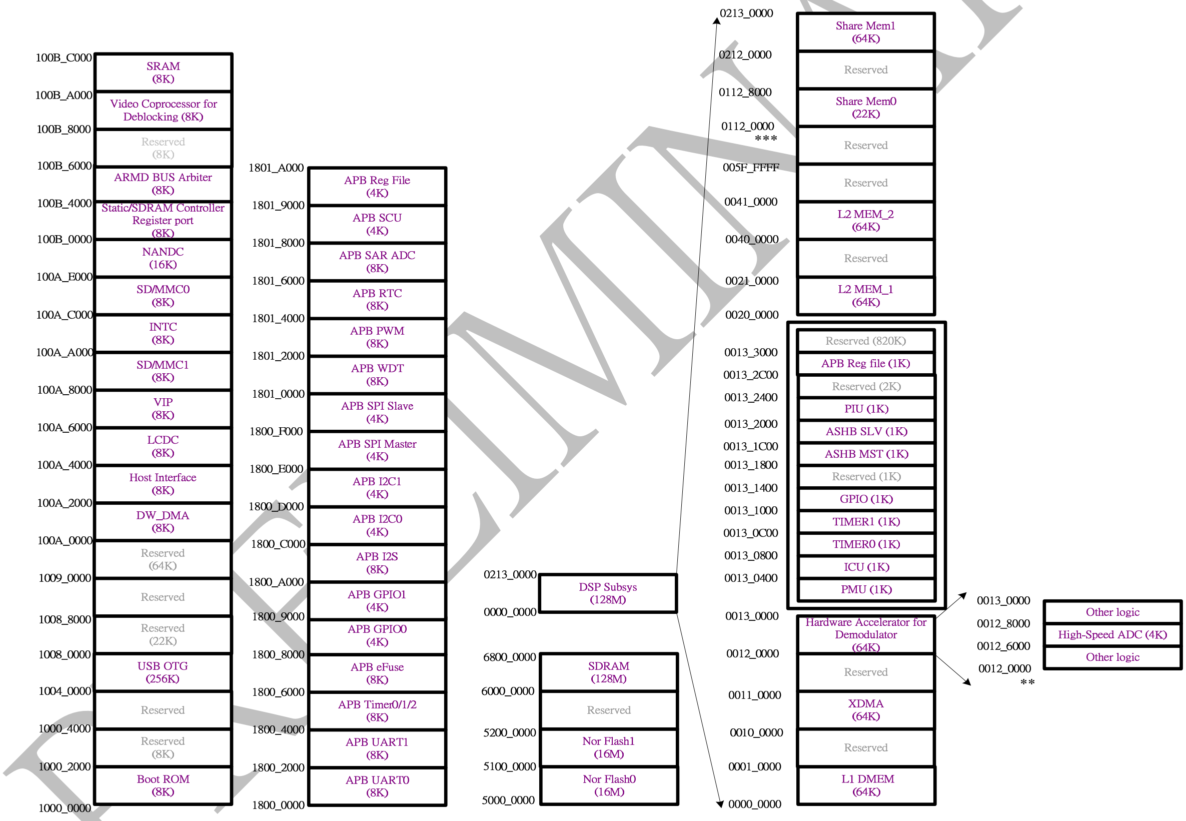


Fig. 3-8 System Memory Map for DSP

Notes

***: other logic shows demodulator except High Speed ADC interfaces. The detailed address space map is described in another Chapter. Please refer to Chapter 37 (Demodulator).*

****: The page address control is used in high-address bit[25:24] for Share Mem0 and Share Mem1 when DSP will access Share Mem0 and Share Mem1. The high-address bits descriptions are in register DSP_APB_REG4. Refer to Chapter 35 (General Register File in DSP System) for detailed descriptions.*

Another, before DSP will access Share Mem0 and Share Mem1, some special bits in register CPU_APB_REG5 must be set. Refer to Chapter 34 (General Register File in CPU

System) for detailed descriptions

3.3 System mode configuration

3.3.1 Debug mode

Table 3-2 RK28xx Debug mode descriptions

	op_mode[1:0] *	**Description**
OP_MODE0	2'b00	CPU Debug
OP_MODE1	2'b01	DSP Debug
OP_MODE2	2'b10	CPU + DSP Debug
OP_MODE3	2'b11	Reserved

***Notes**: * op_mode [1:0] is input pins for RK28xx*

3.3.2 CPU Boot mode

Table 3-3 RK28xx boot mode descriptions

	bt_mode *	**Description**
BOOT_MODE0	1'b0	Boot from Embedded ROM
BOOT_MODE1	1'b1	Boot from NOR Flash bank0

***Notes:** * bt_mode is input pin for RK28xx*

3.4 System Interrupt connection

RK28xx provides an interrupt controller(INTC) for CPU processor, which has 40 general interrupt sources and 2 fast interrupt sources for internal blocks or external devices , and separately generates one IRQ and one FIQ to CPU. Each interrupts triggered type is high level, not programmable. The detailed interrupt sources connection is in the following table 2-4. For detailed interrupt controller setting, please refer to Chapter 11 (Interrupt Controller).

Another , for DSP processor there are also an interrupt control unit (ICU),which has 48 maskable interrupt sources and one nmi interrupt , then generates INT0,INT1,INT2,VINT and NMI interrupt to DSP. The interrupt polarity(low/high) and triggered type(edge/level) for each interrupt sources are configurable. The detailed interrupt sources connection is in the following table 2-5. For detailed ICU setting, please refer to Chapter 12 (Interrupt Control Unit).

Table 3-4 Interrupt sources connection for CPU

Type	**Source #**	**Source Description**	**Polarity**
FIQ	1	Software Interrupt	-
	0	APB GPIOA-D	High level
IRQ	39	DSP system access error int *	High level
	38	DSP master interface error int **	High level
	37	Software Interrupt	-
	36	APB SCU	High level
	35	DSP interrupt by software set	High level
	34	DSP slave interface error int ***	High level
	33	SD/MMC 1	High level
	32	XDMA	High level
	31	PIU command/reply	High level
	30	PIU Semphore 2	High level
	29	PIU Semphore 1	High level
	28	PIU Semphore 0	High level
	27	APB RTC	High level
	26	APB SAR-ADC	High level

IRQ	25	APB PWM3	High level
	24	APB PWM2	High level
	23	APB PWM1	High level
	22	APB PWM0	High level
	21	APB WDT	High level
	20	APB UART1	High level
	19	APB UART0	High level
	18	APB TIMER2 inside CPU system	High level
	17	APB TIMER1 inside CPU system	High level
	16	APB TIMER0 inside CPU system	High level
	15	APB SPI Slave	High level
	14	APB SPI Master	High level
	13	APB I2S	High level
	12	APB I2C1	High level
	11	APB I2C0	High level
	10	Arbiter in EXP BUS	High level
	9	Arbiter in ARMD BUS	High level
	8	USB OTG	High level
	7	APB GPIO E-H	High level
	6	APB GPIO A-D	High level
	5	VIP	High level
	4	SD/MMC 0	High level
	3	LCDC	High level
	2	NANDC	High level
	1	Host Interface	High level
	0	DW_DMA	High level

Notes:

** When Masters inside DSP system access any slave devices and hresp is error, DSP system access error int will be generated.*

*** When Masters inside CPU system access address space inside DSP system and hresp is error, DSP master error int will be generated.*

**** When Masters inside DSP system access address space inside CPU system and hresp is error, DSP slave error int will be generated.*

In general, the three above interrupts will be used to debug only . Refer to Appendix B for the detailed descriptions.

Table 3-5 Interrupt sources connection for DSP

Source #	Source Description	Polarity
nmi	CPU interrupt by software set	High level
47	Reserved	High level
46	VIP	High level
45	LCDC	High level
44	DW_DMA	High level
43	APB GPIO E-H	High level
42	Reserved	High level
41	Reserved	High level
40	Reserved	High level
39	Reserved	High level
38	Reserved	High level
37	Reserved	High level
36	Reserved	High level
35	High-Speed ADC Interface	High level
34	PIU command & reply	High level
33	PIU Semphore 2	High level

32	PIU Semphore 1	High level
31	PIU Semphore 0	High level
30	XDMA channel 15	High level
29	XDMA channel 14	High level
28	XDMA channel 13	High level
27	XDMA channel 12	High level
26	XDMA channel 11	High level
25	XDMA channel 10	High level
24	XDMA channel 9	High level
23	XDMA channel 8	High level
22	XDMA channel 7	High level
21	XDMA channel 6	High level
20	XDMA channel 5	High level
19	XDMA channel 4	High level
18	XDMA channel 3	High level
17	XDMA channel 2	High level
16	XDMA channel 1	High level
15	XDMA channel 0	-
14	Software Interrupt	-
13	Software Interrupt	-
12	Software Interrupt	-
11	Software Interrupt	-
10	Software Interrupt	-
9	Software Interrupt	-
8	Software Interrupt	-
7	Software Interrupt	-
6	ASHB snoop inside DSP system	High level
5	Software Interrupt	-
4	ASHB error interrupt	High level
3	XDMA breakpoint interrupt	High level
2	XDMA error interrupt	High level
1	TIMER 1 inside DSP system	High level
0	TIMER 0 inside DSP system	High level

3.5 System DMA hardware request connection

RK28xx provides 2 DMA controllers: XDMA inside DSP system and DW_DMA inside CPU system. As for XDMA, there are 48 hardware request ports; the trigger type (edge or level) for each of them is programmable. Each XDMA channel is associated with a trigger source, so each channel has been assigned three hardware request ports really. Another, 16 hardware request ports are used in DW_DMA, the trigger type for each of them is high level, not programmable.

Please refer to chapter 9 (DW_DMA) and chapter 10 (XDMA) for detailed usage.

The following tables will describe hardware request connection.

Table 3-6 hardware request connection for DW_DMA

Source #	Source Description	Polarity
0	SD/MMC 0	HIGH level
1	LCDC req0	HIGH level
2	LCDC req1	HIGH level
3	LCDC req2	HIGH level
4	LCDC req3	HIGH level
5	SD/MMC 1 or LCDC req4 *	HIGH level
6	APB I2S txd	LOW level

7	APB I2S rxd	LOW level
8	APB SPI Master txd	HIGH level
9	APB SPI Master rxd	HIGH level
10	APB SPI Slave txd	HIGH level
11	APB SPI Slave rxd	HIGH level
12	APB UART0 txd	LOW level
13	APB UART0 rxd	LOW level
14	APB UART1 txd	LOW level
15	APB UART1 rxd	LOW level

Notes: ** No. 5 hardware request is for SD/MMC 1 or LCDC req4, which is software programmable in CPU_APB_REG4 register. Please refer to Chapter 34 (General Register File in CPU System) for the detailed descriptions.*

Table 3-7 hardware request connection for XDMA

Channel #	**Source Description***	**Polarity**
0	Reserved	High level
1	TIMER1 inside DSP system / XDMA manager port / Reserved	High level
2	High-speed ADC interface	High level
3	Reserved	-
4	Reserved	-
5	Demodulator dma req1	High level
6	Demodulator dma req2	High level
7	Demodulator dma req3	High level
8	Reserved	-
9	Reserved	-
10	TIMER0 inside DSP system	High level
11	TIMER1 inside DSP system	High level
12	Demodulator dma req4	High level
13	Reserved	-
14	Reserved	-
15	Reserved	-

Notes: * Each channel is associated with three hardware trigger sources, which is software configurable; user can select one trigger source to its assigned channel.

Chapter 4 Static/SDRAM Memory Controller

4.1 Design Overview

4.1.1 Overview

The Static/SDRAM Controller is a memory controller that you can control Synchronous DRAMs – SDR-SDRAM, – as well as Static memories – SRAMs and FLASHes

4.1.2 Features

AMBA AHB Interface Features

- AMBA AHB bus-compatible
- Supports all types of AMBA bursts
- Supports AHB data widths of 32 bits
- Supports AHB address width of 32 bits
- Supports busy and early terminations on AHB transactions
- Does not generate split, retry, or error responses on the AMBA bus
- Two-clock-cycle latency from AMBA bus hsel_mem assertion to issue of memory command, depending on optional registering of memory control and data signals
- Supports shared memory address and data buses between SDRAM and Static memories

SDRAM Interface Features

- Glueless connection to all JEDEC-compliant SDRAM
- Supports up to 16 SDRAM address bits
- SDR-SDRAM data width is 32 bits
- Programmable row and column address bit widths up to:
  - 15-bit column address
  - 16-bit row address
  - 2-bit bank address
- Supports 2K to 64K rows, 256 to 32K columns, and 4 banks
- Supports up to 3 chip selects, with a maximum of 4 GB of address space per chip select
- SDRAM timing parameters – tRAS, tRCD, tRP, tWR, tWTR, tRCAR, tXSR, and tRC, – can be programmed to values supported by different SDRAM vendors
- Supports auto refresh with programmable refresh intervals
- Supports self-refresh
- Supports SDRAM power-down mode
- Programmable immediate precharge or delayed precharge modes
- Supports 1 to 4 (programmable) open banks for performance; pages can be non-contiguous –Least Recently Used (LRU) algorithm used during page miss replacements

Static Memory Interface Features

- Supports asynchronous SRAMs and page-mode FLASHes
- Supports up to three sets of timing registers
- Configurable address width of up to 32 bits
- Limited synchronous SRAM and FLASH interface support
- Synchronous SRAM and FLASH frequency could be 1, 1/2, 1/3, 1/4, and so on of the AHB frequency

4.2 Architecture

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

4.2.1 Overview

The Static Memory/SDRAM controller can provide an interface between each of the following memory devices and an AMBA AHB 2.0 bus

- JEDEC-standard SDR-SDRAM
- Asynchronous SRAM, with or without page-mode
- Asynchronous FLASH, with or without page-mode
- Limited synchronous SRAM and FLASH interface support

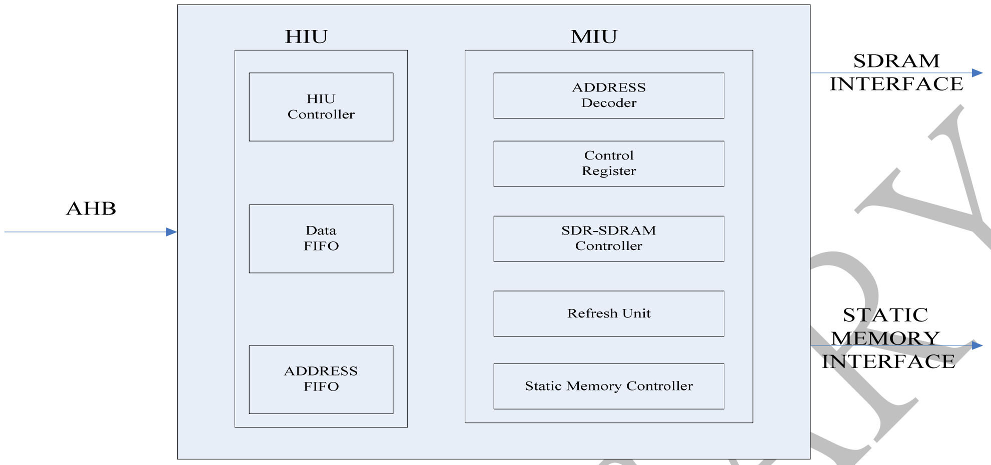


Fig. 4-1 Static Memory/SDRAM block diagrams

4.2.2 Block Descriptions

AMBA Host Interface Unit (HIU)

The AMBA Host Interface Unit (HIU) is the interface between the memctl and the AMBA Advanced High-performance Bus (AHB). The HIU generates memory read/write requests or control register read/write requests to the MIU block, which correspond to transfers on the AMBA bus; the HIU does not distinguish between an SDRAM request and an SRAM/FLASH request.

The HIU consists of the following sub-blocks:

- Address FIFO – Buffers the request of the AMBA AHB and sends memory/register access requests to the MIU; also contains some control information for a read/write transfer
- Write Data FIFO – Buffers write data to the memory and control registers
- Read Data FIFO – Buffers the read data from the memory
- HIU Control – Controls all the HIU sub-blocks by generating the control logic for read and write transfers

Memory Interface Unit (MIU)

The memory interface unit (MIU) is the interface for both SDRAM and Static memories; it generates appropriate address, data, and control signals corresponding to memory read/write transfers. The MIU contains two sets of modules, which are enabled depending on whether you choose the SDRAM or Static memory.

If you choose the SDRAM controller, the MIU includes the following modules:

- SDRAM controller – Generates the SDRAM control signals
- Refresh unit – Generates the SDRAM refresh request at appropriate intervals
- Address decoder – Generates the row, column, and bank addresses that correspond to the logical address provided by the host interface and Decodes and generates the address to SRAM/FLASH from the AHB address
- Control register – Holds the SDRAM control and configuration registers, and holds the control registers and timing registers for Static memories.
- Static control unit – Generates the SRAM/FLASH control signals

4.3 Registers

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

4.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SDR_SCONR	0x00	W	0x1C4F68	SDRAM configuration register
SDR_STMG0R	0x04	W	0x1E5D696	SDRAM timing register0
SDR_STMG1R	0x08	W	0x7008	SDRAM timing register1
SDR_SCTLR	0x0C	W	0x3009	SDRAM control register
SDR_SREFR	0x10	W	0x410	SDRAM refresh register
SDR_SCSLR0_LOW	0x14	W	0x0	Chip select register0 (lower 32bits)
SDR_SCSLR1_LOW	0x18	W	0x5100	Chip select register1 (lower 32bits)
SDR_SCSLR2_LOW	0x1C	W	0x6000	Chip select register2 (lower 32bits)
SDR_SMSKR0	0x54	W	0x149	Mask register 0
SDR_SMSKR1	0x58	W	0x249	Mask register 1
SDR_SMSKR2	0x5C	W	0xC	Mask register 2
SDR_CSREMAP0_LOW	0x84	W	0x50000000	Remap register for chip select0 (lower 32 bits)
SDR_SMTMGR_SET0	0x94	W	0x1154C	Static memory timing register Set0
SDR_SMTMGR_SET1	0x98	W	0x791950	Static memory timing register Set1
SDR_SMTMGR_SET2	0x9C	W	0x1C1950	Static memory timing register Set2
SDR_FLASH_TRPDR	0xA0	W	0xC8	FLASH memory tRPD timing register
SDR_SMCTLR	0xA4	W	0x1201	Static memory control register

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

4.3.2 Detail Register Description

SDR_SCONR
Address: Operational Base + offset (0x00)
SDRAM Config Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	RW	0x1	Reserved
19	RW	0x1	Reserved
18	RW	0x1	Reserved
17:15	RW	0x0	Reserved
14:13	RW	0x1	Specifies SDRAM data width in bits; fixed in 32bits; No use
12:9	RW	0x7	Number of address bits for column address; 15 – reserved 7-14 – correspond to 8-15 bits 0-6 – reserved
8:5	RW	0xb	Number of address bits for row address;

			10-15 – correspond to 11-16 bits 0-10 – reserved
4:3	RW	0x1	Number of bank address bits; bit Values of 0-3 correspond to 1-4 bits, and therefore select 2-16 banks
2:0	-	-	Reserved.

SDR_STMG0R
Address: Operational Base + offset (0x04)
SDRAM Timing Register0

bit	Attr	Reset Value	Description
25:22	RW	0x7	Active-to-active command period; values of 0-15 correspond to t_rc of 1-16 clocks
31:27 21:18	RW	0x0 0x9	Exit self-refresh to active or auto-refresh command time; minimum time controller should wait after taking SDRAM out of self-refresh mode before issuing any active or auto-refresh commands; values 0-511 correspond to t_xsr of 1-512 clocks
17:14	RW	0x7	Auto-refresh period; minimum time between two auto-refresh commands; values 0-15 correspond to t_rcar of 1-16 clocks.
13:12	RW	0x1	For writes, delay from last data in to next precharge command; values 0-3 correspond to t_wr of 1-4 clocks
11:9	RW	0x3	Precharge period; values of 0-7 correspond to t_rp of 1-8 clocks
8:6	RW	0x2	Minimum delay between active and read/write commands; values 0-7 correspond to t_rcd values of 1-8 clocks
5:2	RW	0x5	Minimum delay between active and precharge commands; values of 0-15 correspond to T_RAS_MIN of 1-16 clocks
26 1:0	RW	0x0 0x2	Delay in clock cycles between read command and availability of first data 0 – 1 clock 1 – 2 clocks 2 – 3 clocks 3 – 4 clocks 4 – Reserved 5 – Reserved 6, 7 – reserved

SDR_STMG1R
Address: Operational Base + offset (0x08)
SDRAM Timing Register1

bit	Attr	Reset Value	Description
31:22	-	-	Reserved.
21:20	RW	0x0	Reserved.
19:16	RW	0x7	Number of auto-refreshes during initialization; values 0-15 correspond to 1-16 auto-refreshes
15:0	RW	0x8	Number of clock cycles to hold SDRAM inputs stable after power up, before issuing any commands.

SDR_SCTLR
Address: Operational Base + offset (0x0C)

SDRAM Control Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	-	-	Reserved.
19	-	-	Reserved.
18	-	-	Reserved.
17	RW	0x0	Reserved.
16:12	RW	0x3	Number of SDRAM internal banks to be open at any time; values of 1-16 correspond to 0-15 banks open.
11	R	0x0	Read only. When "1," indicates SDRAM is in self-refresh mode. When "self_refresh/deep_power_mode" bit (bit 1 of SCTLR) is set,it may take some time before SDRAM is put into self-refresh mode, depending on whether all rows or one row are refreshed before entering self-refresh mode defined by full_refresh_before_sr bit. Before gating clock in self-refresh mode, ensure this bit is set
9	RW	0x0	Set to 1, forces controller to do update of SDRAM mode register; bit is cleared by controller once it has finished mode register update
8:6	R/W	0x0	Indicates number of registers inserted in read data path for SDRAM in order to correctly latch data; values 0-7 indicate 0-7 registers
5	R/W	0x0	Controls number of refreshes done by SDR_memctl after SDRAM is taken out of self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just 1 row before entering self-refresh mode
4	R/W	0x0	Controls number of refreshes done by SDR_memctl before putting SDRAM into self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just one row before entering self-refresh mode
3	R/W	0x1	Determines when row is precharged: 0 – Immediate precharge; row precharged at end of read/write operation 1 – Delayed precharge; row kept open after read/write operations
2	R/W	0x0	Forces to put SDRAM in power-down mode;
1	R/W	0x0	Forces to put SDRAM in self-refresh mode. Bit can be cleared by writing to this bit or by clear_sr_dp pin, generated by external power management unit
0	R/W	0x1	Forces to initialize SDRAM; bit reset to 0 by SDR_memctl once initialization sequence is complete

SDR_SREFR

Address: Operational Base + offset (0x10)

SDRAM Refresh Interval Register

bit	Attr	Reset Value	Description
31:24	R	0x0	Reserved.
23:16	RW	0x0	Reserved.
15:0	RW	0x410	Number of clock cycles between consecutive refresh cycles;

SDR_SCSLR0
Address: Operational Base + offset (0x14)
chip_select_register0

bit	Attr	Reset Value	Description
31:16	R	0x0000	Upper 16bits of base address for static memory bank0
15:0	-	-	Reserved.

SDR_SCSLR1
Address: Operational Base + offset (0x18)
chip_select_register1

bit	Attr	Reset Value	Description
31:16	R	0x5100	Upper 16bits of base address for static memory bank1
15:0	-	-	Reserved.

SDR_SCSLR2
Address: Operational Base + offset (0x1C)
chip_select_register0

bit	Attr	Reset Value	Description
31:16	R	0x6000	Upper 16bits of base address for SDRAM
15:0	-	-	Reserved.

SDR_SMSKR0
Address: Operational Base + offset (0x54)
Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	R/W	0x1	Register determines which timing parameters of memory connect to static memory bank0; primarily used for specifying static memories 0 – register set 0 , set in SDR_SMTMGR_SET0 1 – register set 1 , set in SDR_SMTMGR_SET1 2 – register set 2 , set in SDR_SMTMGR_SET2
7:5	R/W	0x2	Type of memory connected to static memory bank0: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R/W	0x9	size of memory connected to static memory bank0; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

SDR_SMSKR1
Address: Operational Base + offset (0x58)
Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.

10:8	R/W	0x2	Register determines which timing parameters of memory connect to static memory bank1; primarily used for specifying static memories 0 – register set 0 , set in SDR_SMTMGR_SET0 1 – register set 1 , set in SDR_SMTMGR_SET1 2 – register set 2 , set in SDR_SMTMGR_SET2
7:5	R/W	0x2	Type of memory connected to static memory bank1: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R/W	0x9	size of memory connected to static memory bank1; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

SDR_SMSKR2
Address: Operational Base + offset(0x5C)
Address Mask Registers

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:5	R/W	0x0	Type of memory connected to SDRAM 0 – SDRAM Others – Reserved
4:0	R/W	0xC	size of memory connected to SDRAM; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10 – 32MB 11 – 64MB 12 – 128MB 13~17 – Reserved

SDR_CSREMAP0_LOW
Address: Operational Base + offset (0x84)
REMAP REGISTER0

bit	Attr	Reset Value	Description
31:16	R	0x5000	Represent lower remap register bits for static memory bank0; compared with corresponding AHB address to generate chip select0; number of compared bits depends on size of memory selected by chip select0 (specified SDR_SMSKR0):

			64KB – bits 31:16 compared 128 KB – bits 31:17 compared
15:0	-	-	Reserved.

SDR_SMTMGR_SET0

Address: Operational Base + offset (0x94)

Static Memory Timing Register - Set0

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set0
27	RW	0x0	Valid if register set0 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x0	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between "read to write", or "write to read", or "read to read" when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x5	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0xc	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

SDR_SMTMGR_SET1

Address: Operational Base + offset (0x98)

Static Memory Timing Register – Set1

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set1
27	RW	0x0	Valid if register set1 is used to control low-frequency

			synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0xf	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between "read to write", or "write to read", or "read to read" when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

SDR_SMTMGR_SET2

Address: Operational Base + offset (0x9C)

Static Memory Timing Register – Set2

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set2
27	RW	0x0	Valid if register set2 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode

			1 – device supports page mode
22:19	RW	0x3	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x4	Static memory idle cycles between "read to write", or "write to read", or "read to read" when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

SDR_FLASH_TRPDR
Address: Operational Base + offset (0xA0)
FLASH Timing Register

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.
11:0	RW	0xC8	FLASH reset/power-down high to read/write delay; values correspond to sm_rp_n high to read/write delay minus one

SDR_SMCTLR
Address: Operational Base + offset (0xA4)
Static Memory Control Register

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:13	RW	0x0	Width of Static memory data bus controlled by Static memory register SET2: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
12:10	RW	0x4	Width of Static memory data bus controlled by Static memory register SET1: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
9:7	RW	0x4	Width of Static memory data bus controlled by Static memory register set 0: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits

			100 – 8 bits
6:4	-	-	Reserved.
3:1	RW	0x0	FLASH write-protection mode; writing 0 forces FLASH memory bootblock to write protect; the three bits correspond to three register sets
0	RW	0x1	FLASH reset/power-down mode; after reset, controller internally performs a power-down for FLASH and then sets this bit to 1 to force FLASH to power-down mode during normal operation: 0 – Forces FLASH to power-down mode 1 – Takes FLASH out of power-down mode

4.4 Functional Description

4.4.1 Operation

Basic Access Operation

1. A page-hit sigle-cycle write:

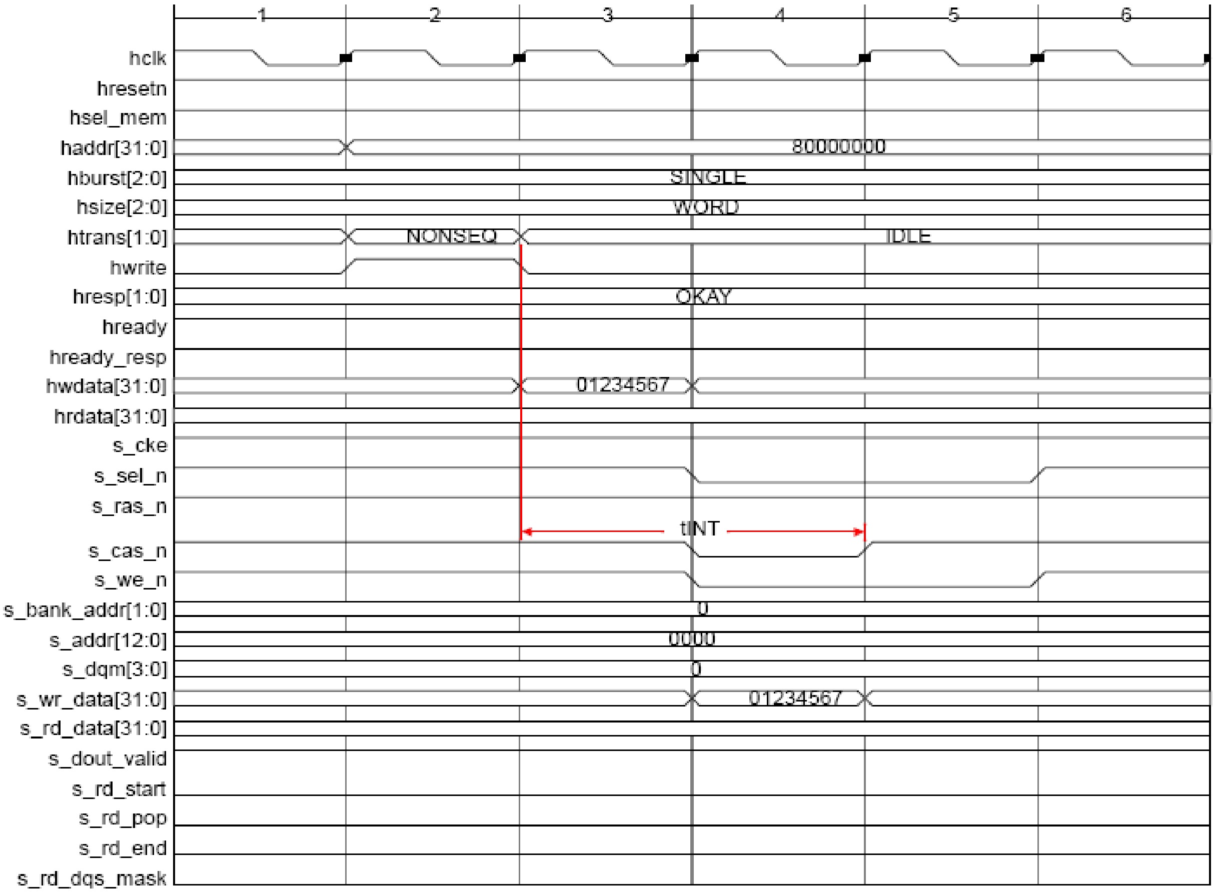


Fig. 4-2 SDRAM Page-Hit Single Write

2、A page-miss single-cycle write:

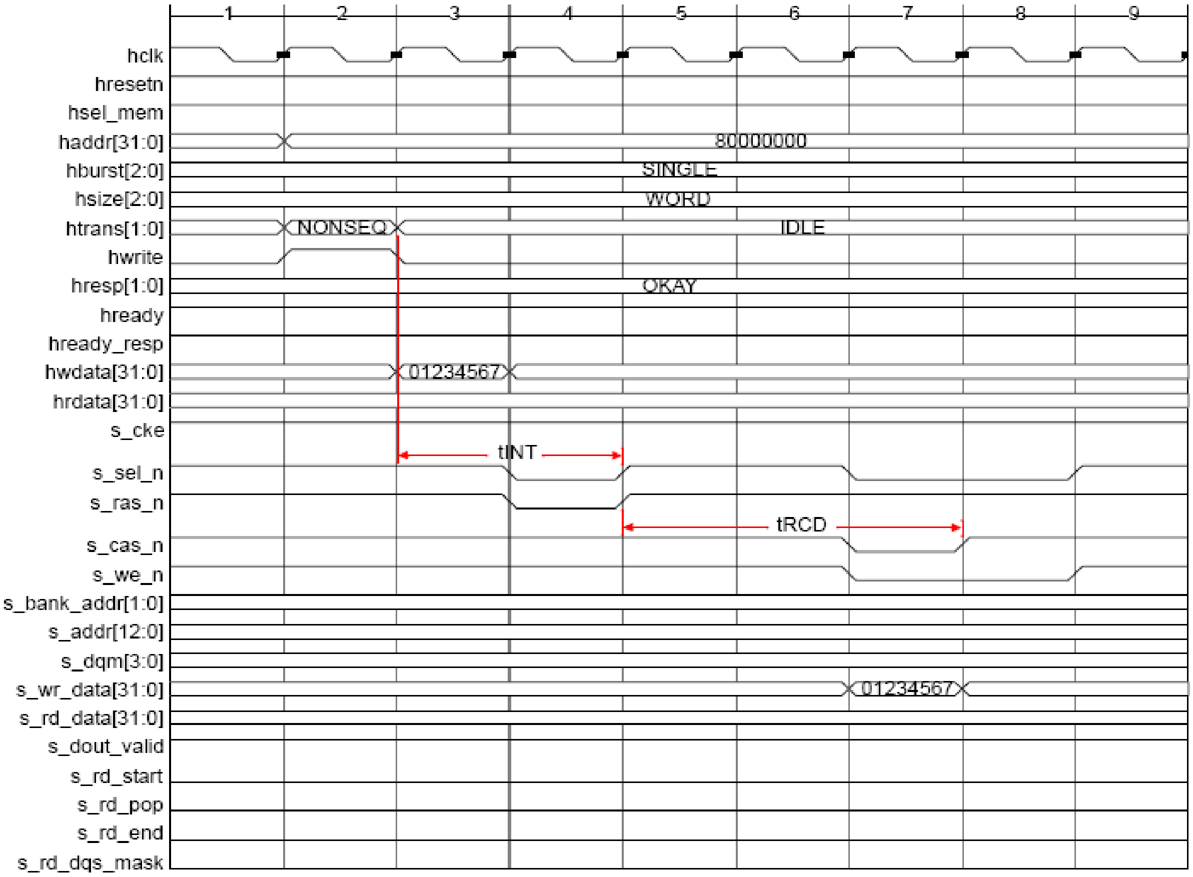


Fig. 4-3 SDRAM Page-Miss Single Write

3 、A page-hit burst write (hburst == INCR8)

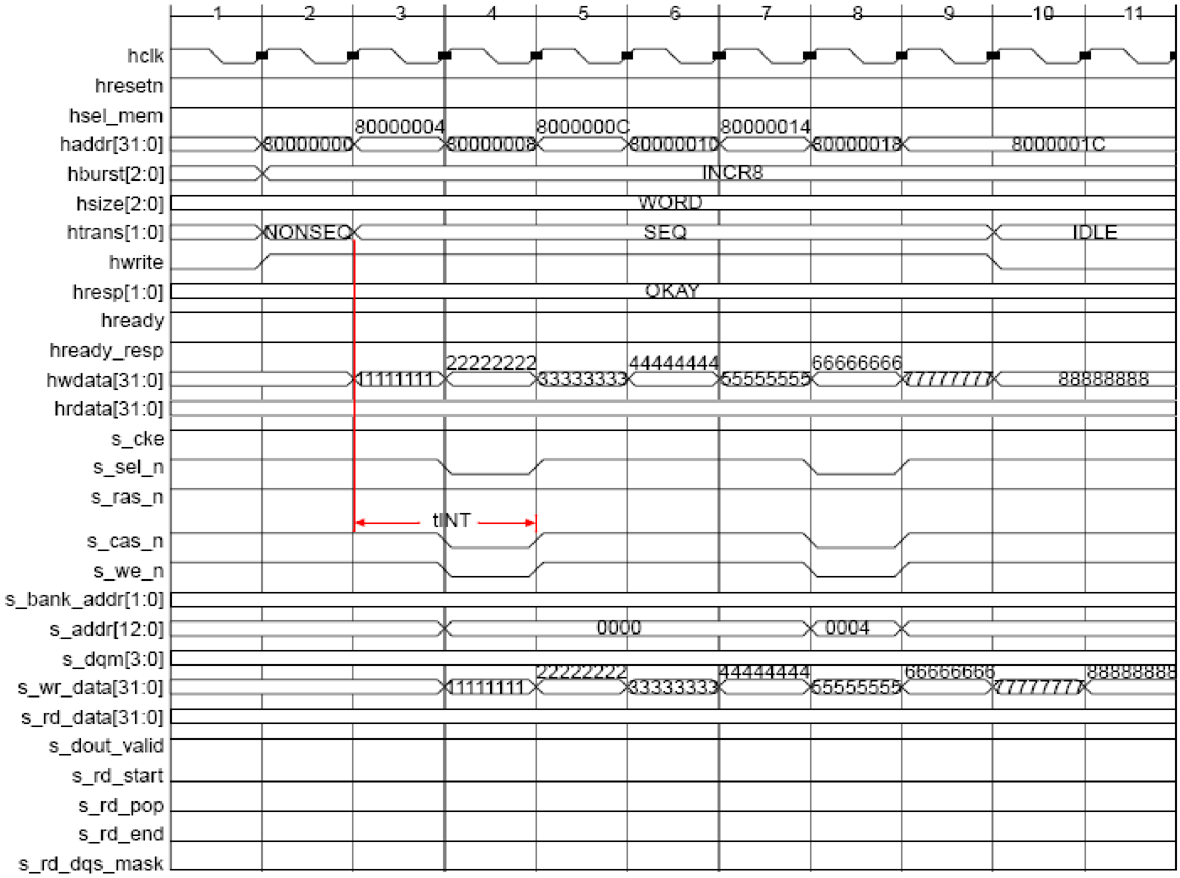


Fig. 4-4 SDRAM Page-Hit Busrt Write

4、A page-hit single-cycle read：

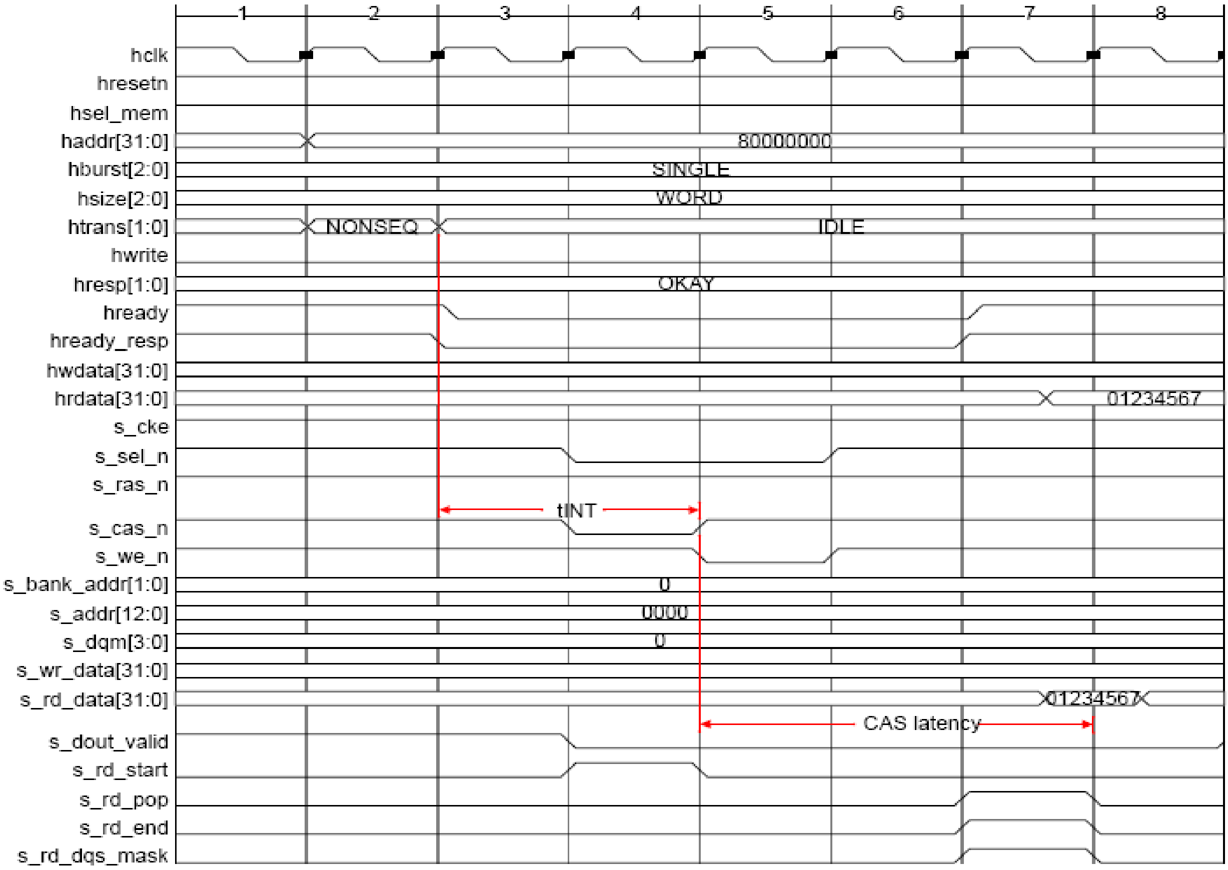


Fig. 4-5 SDRAM Page-Hit Single Read

5、a page-miss single-cycle read:

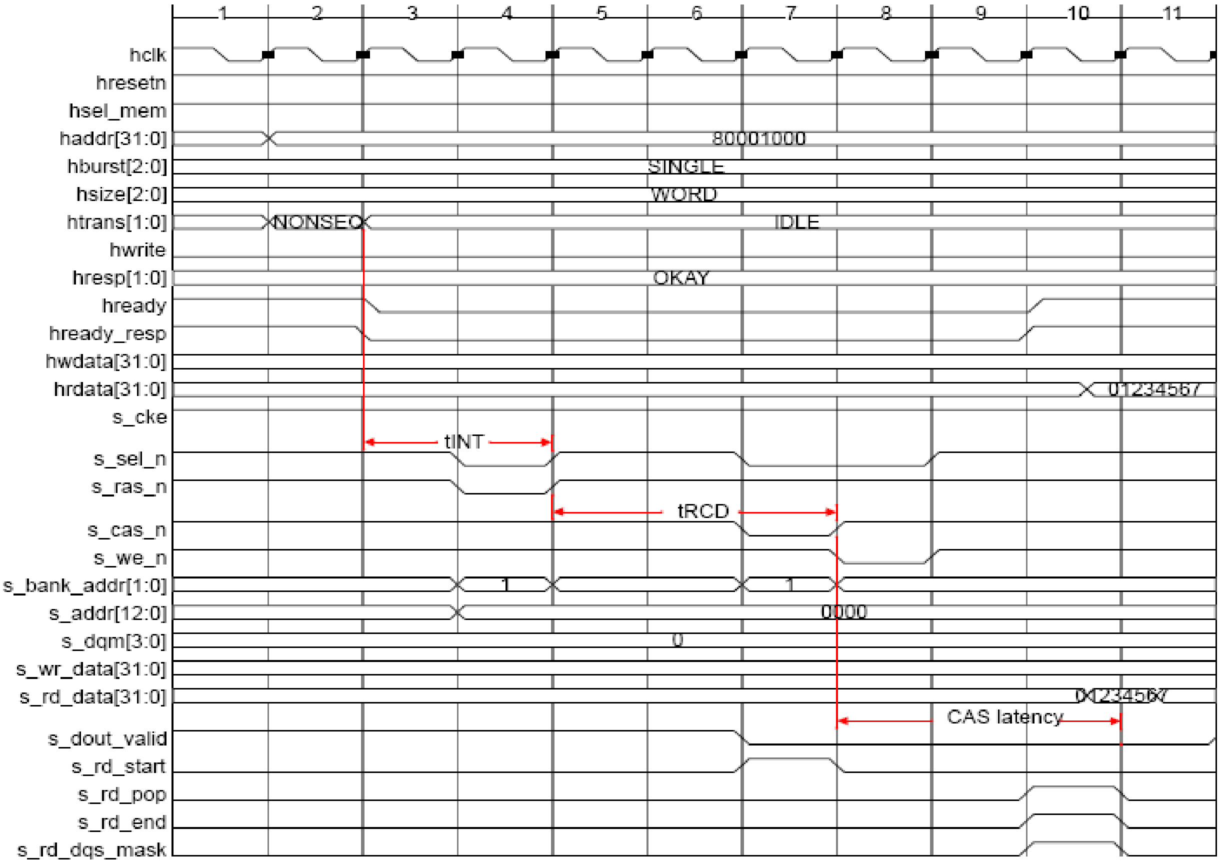


Fig. 4-6 SDRAM Page-Miss Single Read

6、A page-hit burst read:

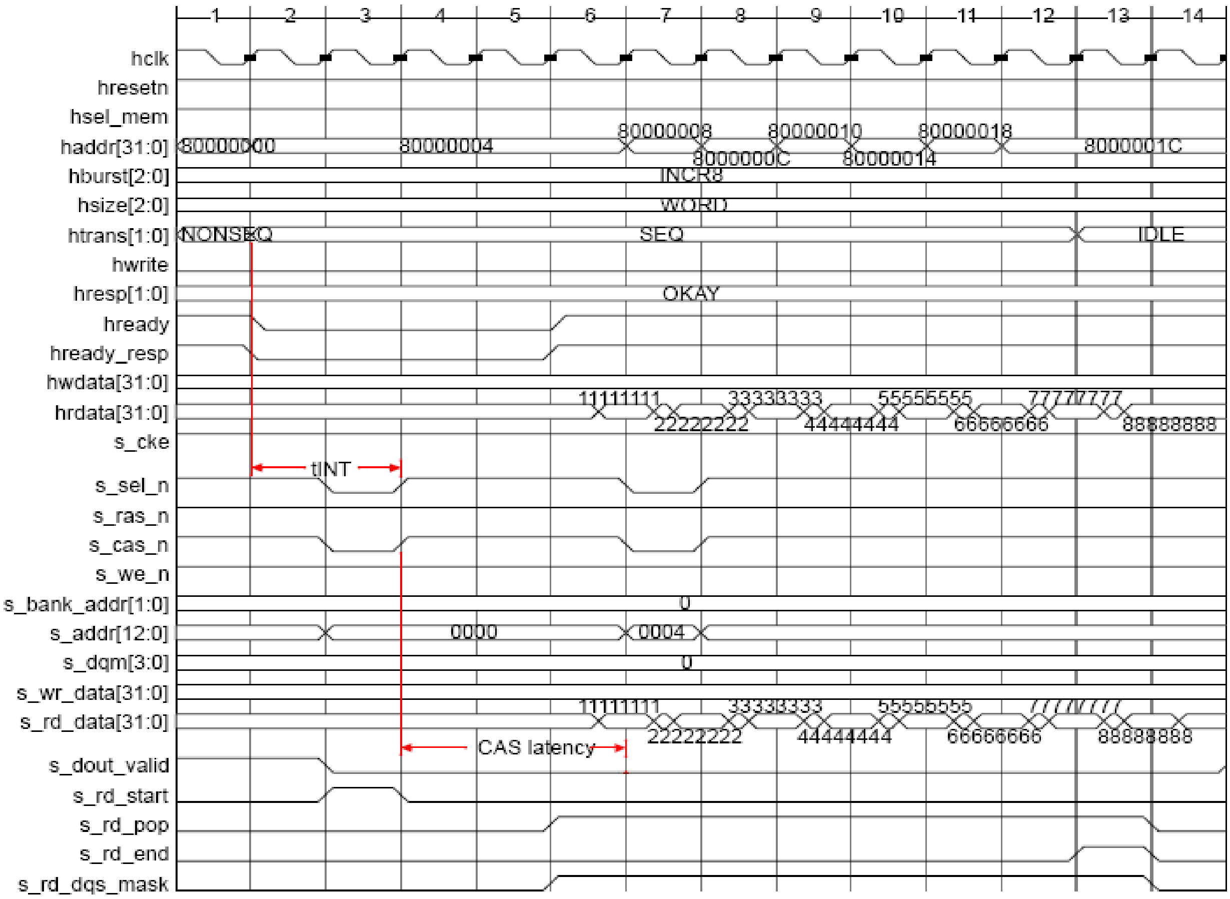


Fig. 4-7 SDRAM Page-Hit Busrt Read

Power-On Initialization

The SDR-SDRAM controller follows the JEDEC-recommended SDR-SDRAM power-on initialization sequence as follows:

1. Apply power and start clock; maintain a NOP condition at the inputs
2. Maintain stable power, stable clock, and NOP input conditions for a minimum of t_init clock cycles
3. Issue precharge commands for all banks of the device
4. Issue auto-refresh commands, depending on the value num_init_ref in the programmable register
5. Issue a set-mode register command to initialize the mode registe the commands issued to the SDRAM by the controller during the power-on initialization are shown as followed

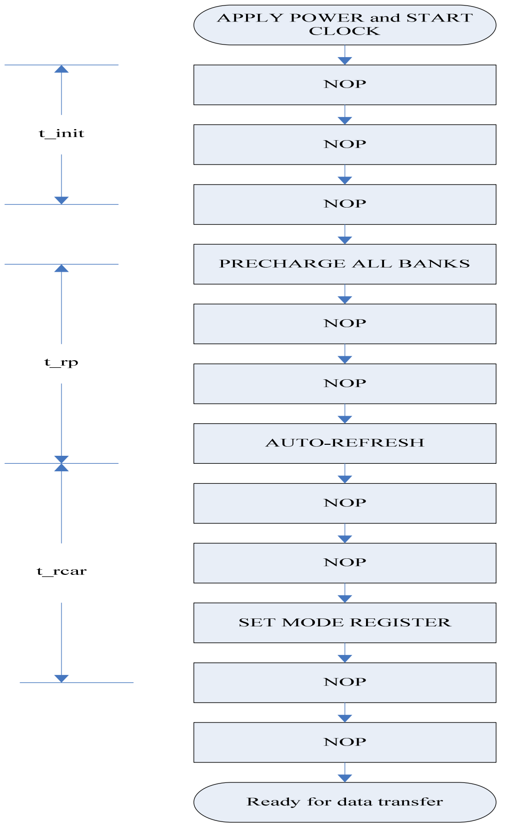


Fig. 4-8 Static Memory/SDRAM Controller power on sequence

Read / Write Operation

The SDRAM Controller converts all AHB bursts to 4-word bursts on the SDRAM side. The memory bursts are concatenated to achieve continuous data flow for long AHB bursts. You can terminate the memory read/write burst with either a precharge command or terminate command, depending on which precharge mode – immediate precharge or delayed precharge – that you program. You can also terminate the write burst with a subsequent write burst

The SDRAM Controller supports two precharge modes – immediate precharge and delayed precharge. If you program for an immediate precharge mode, then the SDRAM Controller closes the open row after a read or write access. If you program for a delayed precharge mode, then the SDR_memctl keeps the row open after an access. The SDR_memctl can keep multiple banks open at the same time, depending on the value of num_open_bank in the programmable register. When the number of open banks reaches the num_open_bank and an access to a new bank comes, the SDRAM Controller will close the oldest bank (the bank opened first) before opening the new bank.

Set-Mode Register

The SDRAM Controller automatically sets the SDR-SDRAM mode register during the power-up initialization. During normal operation, if you want to set the mode register, you need to set set_mode_reg (bit 9 of SCTLR) to 1. After the memory controller finishes the mode register setting, it clears the set_mode_reg to 0.

The "burst length" field and the "burst type" field of the SDR-SDRAM-mode register are fixed by the SDRAM Controller to "010" (burst length 4) and "0" (sequential burst), respectively. The SDRAM Controller programs the "CAS latency" field and the "operating mode" field of the mode register according to the values provided by the user in the control and timing registers.

Auto-Refresh

Auto-refresh commands are issued when the refresh control block issues refresh requests. During normal refresh operations, the SDRAM Controller always refreshes one row at a time. It is important for the user to program the tREF refresh interval register after a reset. If you need to refresh the SDRAM while a burst is active, normally the

SDRAM Controller will issue the refresh command after the ongoing burst completes. However, if the ongoing burst is an AHB INCR burst, the SDRAM Controller will stop the burst, issue the refresh command, and then resume the burst.

The SDRAM Controller takes into account the maximum time it takes to complete a worst-case burst. This is the time to complete a read burst corresponding to an INCR16 burst on the AMBA bus, and with an AMBA-to-SDRAM data width ratio of 2:1. It is reasonable to assume 50 cycles for this worst-case burst, with 32 cycles for the data and the remaining 17 cycles for various latencies for the worst case.

The t_ref value can be calculated using the following equation:

t_ref = refresh_period / clock_period

Where refresh_period = typically 7.8/15.6 s.

The tREF is the value of a free-running counter that the refresh logic in the SDRAM Controller operates on. When the count expires, the refresh logic gives a refresh request to the SDRAM controller.

Since the 64 ms refresh period is the same for most SDRAMs, the total number of rows in the SDRAM limits the minimum operating frequency for the SDR_memctl. While calculating the minimum frequency, use the following equation:

tREF > 50*(1/f).

Typically, the refresh cycle is 15.6 s or 7.8 s, depending on the refresh rate; The table is summarized as followed:

Number of rows	tREF	Min Frequency
64K	(64ms-(50/f))/65536	51Mhz
32K	(64ms-(50/f))/32768	26Mhz
16K	(64ms-(50/f))/163904	13Mhz
8K	(64ms-(50/f))/8192	6Mhz
4K	(64ms-(50/f))/4096	3Mhz
2K	(64ms-(50/f))/2048	1.5Mhz

The refresh logic in the SDR_memctl is inactive when the SDR_memctl forces the SDRAM into self-refresh or power-down mode.

Self-Refresh

You can put the SDRAM into self-refresh mode, at which point the SDRAM retains data without external clocking and auto-refresh. The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit the self-refresh mode

You can force the SDRAM Controller to enter self-refresh mode by programming bit 1 of the SDRAM control register (SCTLR) (Address 32’hxxxx_xx0C). The SDRAM Controller forces the SDRAM to come out of self-refresh mode when bit 2 of the SCTLR is set to 0. You can set this bit to 0 by either programming the SDRAM control register or driving the clear_sr_dp pin high. You can use the clear_sr_dp pin when the code resides in the SDRAM, and the SDRAM itself is in self-refresh mode.

Bits 4 and 5 of the SCTLR specify the type of refresh done by the SDRAM Controller just prior to entering self-refresh mode and just after entering self-refresh mode. Programming bit 4 of the SCTLR to 0 forces the SDRAM Controller to refresh only one row before putting the SDRAM into self-refresh mode. The default value of 1 forces the SDRAM Controller to perform auto-refreshes for all rows. Bit 5 does the same, except that it controls the refresh pattern just after coming out of self-refresh mode.

Since it takes time between programming the control register bit to the SDRAM entering self-refresh mode, the SDRAM Controller provides a read-only register bit (bit 11 of the SDRAM control register) to indicate that the SDRAM is already in self-refresh mode. If you want to gate off the clock to the SDRAM Controller when the SDRAM is in self-refresh mode, you should ensure this bit is set to 1 before you stop the clock.

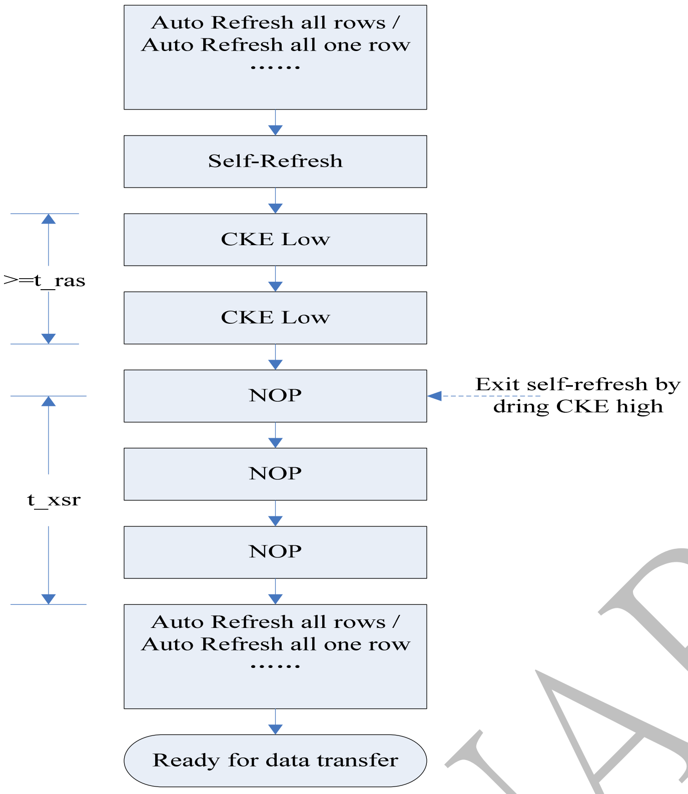


Fig. 4-9 Static Memory/SDRAM Controller self_reflesh mode

The SDRAM must remain in self-refresh mode for a minimum period of t_ras and can remain in self-refresh mode for an indefinite period of time. After the SDRAM exits self-refresh mode, the SDRAM Controller issues NOP commands for t_xsr before it issues any other command. The t_ras and t_xsr are programmable register values and have default values. These registers can be programmed only once after reset.

When an AHB read/write request to the SDRAM occurs while the SDRAM is in self-refresh mode, the SDRAM Controller generates dummy ready signals to the AHB without accessing external memory; no error response is generated on the AHB bus.

Power-Down

The SDRAM can be put into power-down mode to save power. There are two ways to force the SDR_memctl to put the SDRAM in power-down mode:

- Program bit 2 of SCTLR to 1; should be 0 to bring the SDRAM out of power-down mode.
- Use the power-down input pin; can be driven by an external power management unit; the SDRAM will be in power-down mode as long as this signal stays high

The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit power-down mode

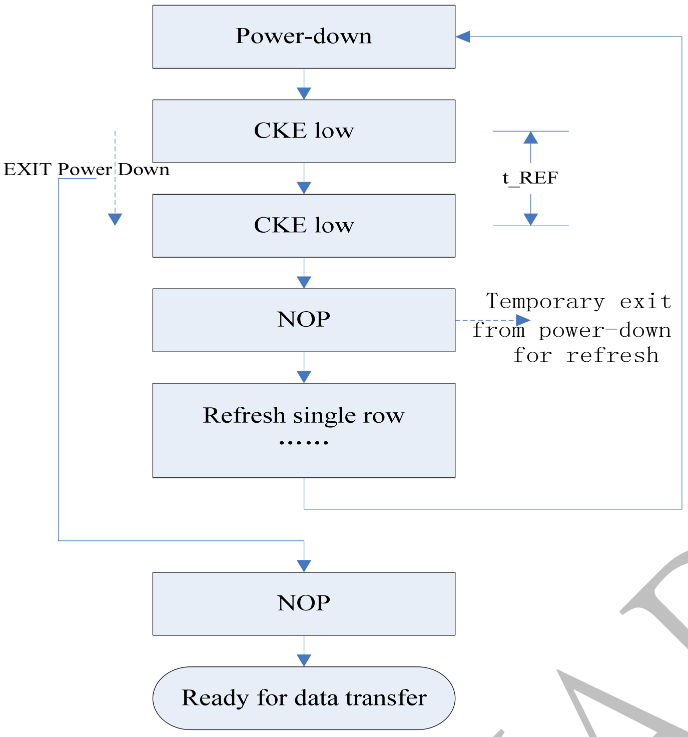


Fig. 4-10 Static Memory/SDRAM Controller power_off mode

When in SDRAM power-down mode, the SDRAM Controller keeps switching the device back and forth between power-down and refresh mode. It remains in power-down for a t_ref period of time, then comes out of power-down and does a single-row refresh; then it again goes into power-down mode.

The SDRAM Controller keeps the SDRAM in this periodical power-down/refresh/power-down sequence until it is commanded to exit power-down mode (by programming bit 2 of SCTLR to 0) When an AHB read/write request to the SDRAM occurs while the SDRAM is in power-down mode,the SDRAM Controller brings the SDRAM out of power-down mode and issues the read/write access to the SDRAM. The SDRAM Controller then puts the SDRAM back to power-down mode after the read/write access.

Chapter 5 Static/Mobile SDRAM Memory Controller

5.1 Design Overview

5.1.1 Overview

The Static Memory/Mobile SDRAM Controller is a memory controller that you can control Synchronous DRAMs – MSDR-SDRAM,– as well as Static memories – SRAMs and FLASHes

5.1.2 Features

AMBA AHB Interface Features

- AMBA AHB bus-compatible
- Supports all types of AMBA bursts
- Supports AHB data widths of 32.
- Supports AHB address width of 32 bits
- Supports busy and early terminations on AHB transactions
- Does not generate split, retry, or error responses on the AMBA bus
- Two-clock-cycle latency from AMBA bus hsel_mem assertion to issue of memory command, depending on optional registering of memory control and data signals
- Supports shared memory address and data buses between Mobile SDRAM and Static memories

MOBILE SDRAM Interface Features

- Glueless connection to all JEDEC-compliant Mobile SDRAM
- Supports up to 16 SDRAM address bits
- Mobile SDRAM data width is 32 bits
- Programmable row and column address bit widths up to:
  - 15-bit column address
  - 16-bit row address
  - 2-bit bank address
- Supports 2K to 64K rows, 256 to 32K columns, and 4 banks
- Supports up to 3 chip selects, with a maximum of 4 GB of address space per chip select
- Mobile SDRAM timing parameters – tRAS, tRCD, tRP, tWR, tWTR, tRCAR, tXSR, and tRC, – can be programmed to values supported by different SDRAM vendors
- Supports auto refresh with programmable refresh intervals
- Supports self-refresh
- Supports Mobile SDRAM power-down mode
- Programmable immediate precharge or delayed precharge modes
- Supports 4 (programmable) open banks for performance; pages can be non-contiguous –Least Recently Used (LRU) algorithm used during page miss replacements

Static Memory Interface Features

- Supports asynchronous SRAMs and page-mode FLASHes
- Supports up to three sets of timing registers
- Configurable address width of up to 32 bits
- Limited synchronous SRAM and FLASH interface support
- Synchronous SRAM and FLASH frequency could be 1, 1/2, 1/3, 1/4, and so on of the AHB frequency

5.2 Architecture

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

5.2.1 Overview

The MSDR_memctl can provide an interface between each of the following memory devices and an AMBA AHB 2.0 bus

- JEDEC-standard Mobile-SDRAM
- Asynchronous SRAM, with or without page-mode
- Asynchronous FLASH, with or without page-mode
- Limited synchronous SRAM and FLASH interface support

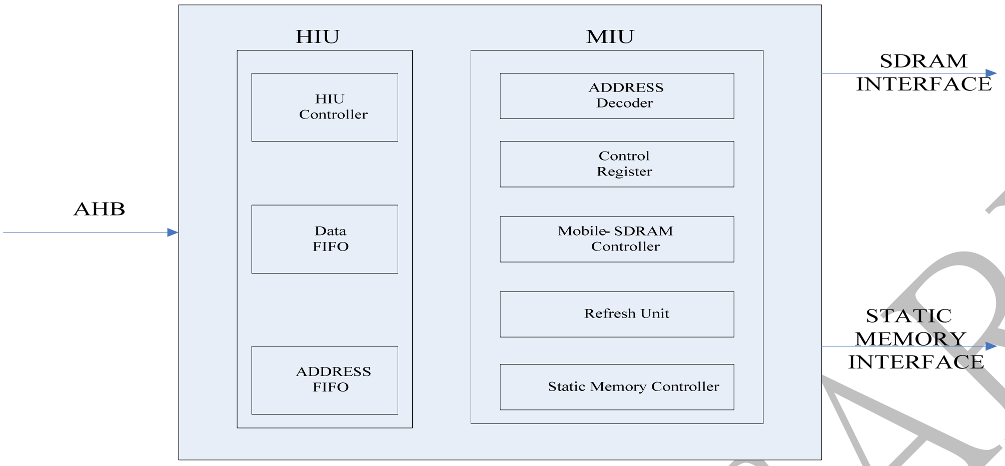


Fig. 5-1 Static Memory/Mobile SDRAM Controller Block Diagram

5.2.2 Block Descriptions

AMBA Host Interface Unit (HIU)

The AMBA Host Interface Unit (HIU) is the interface between the MSDR_memctl and the AMBA Advanced High-performance Bus (AHB). The HIU generates memory read/write requests or control register read/write requests to the MIU block, which correspond to transfers on the AMBA bus; the HIU does not distinguish between an SDRAM request and an SRAM/FLASH request.

The HIU consists of the following sub-blocks:

- Address FIFO – Buffers the request of the AMBA AHB and sends memory/register access requests to the MIU; also contains some control information for a read/write transfer
- Write Data FIFO – Buffers write data to the memory and control registers
- Read Data FIFO – Buffers the read data from the memory
- HIU Control – Controls all the HIU sub-blocks by generating the control logic for read and write transfers

Memory Interface Unit (MIU)

The memory interface unit (MIU) is the interface for both SDRAM and Static memories; it generates appropriate address, data, and control signals corresponding to memory read/write transfers. The MIU contains two sets of modules, which are enabled depending on whether you choose the SDRAM or Static memory.

If you choose the SDRAM controller, the MIU includes the following modules:

- SDRAM controller – Generates the SDRAM control signals
- Refresh unit – Generates the SDRAM refresh request at appropriate intervals
- Address decoder – Generates the row, column, and bank addresses that correspond to the logical address provided by the host interface and Decodes and generates the address to SRAM/FLASH from the AHB address
- Control register – Holds the SDRAM control and configuration registers, and holds the control registers and timing registers for Static memories.
- Static control unit – Generates the SRAM/FLASH control signals

5.3 Registers

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

5.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
MSDR_SCONR	0x00	W	0x1C4F68	SDRAM configuration register
MSDR_STMG0R	0x04	W	0x1E5D696	SDRAM timing register0
MSDR_STMG1R	0x08	W	0x7008	SDRAM timing register1
MSDR_SCTLR	0x0C	W	0x3009	SDRAM control register
MSDR_SREFR	0x10	W	0x410	SDRAM refresh register
MSDR_SCSLR0_LOW	0x14	W	0x0	Chip select register0 (lower 32bits)
MSDR_SCSLR1_LOW	0x18	W	0x5100	Chip select register1 (lower 32bits)
MSDR_SCSLR2_LOW	0x1C	W	0x6000	Chip select register2 (lower 32bits)
MSDR_SMSKR0	0x54	W	0x149	Mask register 0
MSDR_SMSKR1	0x58	W	0x249	Mask register 1
MSDR_SMSKR2	0x5C	W	0xC	Mask register 2
MSDR_CSREMAP0_LOW	0x84	W	0x50000000	Remap register for chip select0 (lower 32 bits)
MSDR_SMTMGR_SET0	0x94	W	0x1154C	Static memory timing register Set0
MSDR_SMTMGR_SET1	0x98	W	0x791950	Static memory timing register Set1
MSDR_SMTMGR_SET2	0x9C	W	0x1C1950	Static memory timing register Set2
MSDR_FLASH_TRPDR	0xA0	W	0xC8	FLASH memory tRPD timing register
MSDR_SMCTLR	0xA4	W	0x1201	Static memory control register
MSDR_EXN_MODE_REG	0xAC	W	0x0	Extended Mode Register

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

5.3.2 Detail Register Description

MSDR_SCONR
Address: Operational Base + offset (0x00)
SDRAM Config Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	RW	0x1	Reserved.
19	RW	0x1	Reserved.
18	RW	0x1	Reserved.
17:15	RW	0x0	Reserved.
14:13	RW	0x1	Specifies SDRAM data width in bits; For all other SDRAMs, bits represent: 00 – 16 bits 01 – 32 bits 10 – 64 bits 11 – 128 bits
12:9	RW	0x7	Number of address bits for column address;

			15 – reserved 7-14 – correspond to 8-15 bits 0-6 – reserved
8:5	RW	0xb	Number of address bits for row address; 10-15 – correspond to 11-16 bits 0-10 – reserved
4:3	RW	0x1	Number of bank address bits; bit Values of 0-3 correspond to 1-4 bits, and therefore select 2-16 banks
2:0	-	-	Reserved.

MSDR_STMG0R
Address: Operational Base + offset (0x04)
SDRAM Timing Register0

bit	Attr	Reset Value	Description
25:22	RW	0x7	Active-to-active command period; values of 0-15 correspond to t_rc of 1-16 clocks
31:27 21:18	RW	0x0 0x9	Exit self-refresh to active or auto-refresh command time; minimum time controller should wait after taking SDRAM out of self-refresh mode before issuing any active or auto-refresh commands; values 0-511 correspond to t_xsr of 1-512 clocks
17:14	RW	0x7	Auto-refresh period; minimum time between two auto-refresh commands; values 0-15 correspond to t_rcar of 1-16 clocks.
13:12	RW	0x1	For writes, delay from last data in to next precharge command; values 0-3 correspond to t_wr of 1-4 clocks
11:9	RW	0x3	Precharge period; values of 0-7 correspond to t_rp of 1-8 clocks
8:6	RW	0x2	Minimum delay between active and read/write commands; values 0-7 correspond to t_rcd values of 1-8 clocks
5:2	RW	0x5	Minimum delay between active and precharge commands; values of 0-15 correspond to T_RAS_MIN of 1-16 clocks
26 1:0	RW	0x0 0x2	Delay in clock cycles between read command and availability of first data 0 – 1 clock 1 – 2 clocks 2 – 3 clocks 3 – 4 clocks 4~7 – Reserved

MSDR_STMG1R
Address: Operational Base + offset (0x08)
SDRAM Timing Register1

bit	Attr	Reset Value	Description
31:22	-	-	Reserved.
21:20	RW	0x0	Reserved.
19:16	RW	0x7	Number of auto-refreshes during initialization; values 0-15 correspond to 1-16 auto-refreshes
15:0	RW	0x8	Number of clock cycles to hold SDRAM inputs stable after power up, before issuing any commands.

MSDR_SCTLR

Address: Operational Base + offset (0x0C)

SDRAM Control Register

bit	Attr	Reset Value	Description
31:21	-	-	Reserved.
20	R	0x0	Status of deep-power-down for mobile SDRAM. 0 – Mobile SDRAM not in deep-power-down 1 – Mobile SDRAM in deep power-down
19	RW	0x0	Reserved.
18	RW	0x0	Commands controller to update Mobile-SDRAM extended-mode register; once mode register update is done, controller automatically clears bit
17	RW	0x0	Reserved.
16:12	RW	0x3	Number of SDRAM internal banks to be open at any time; values of 1-16 correspond to 0-15 banks open.
11	R	0x0	Read only. When “1,” indicates SDRAM is in self-refresh mode.When “self_refresh/deep_power_mode” bit (bit 1 of SCTLR) is set,it may take some time before SDRAM is put into self-refresh mode, depending on whether all rows or one row are refreshed before entering self-refresh mode defined by full_refresh_before_sr bit .Before gating clock in self-refresh mode, ensure this bit is set
9	RW	0x0	Set to 1, forces controller to do update of SDRAM mode register; bit is cleared by controller once it has finished mode register update
8:6	R/W	0x0	Indicates number of registers inserted in read data path for SDRAM in order to correctly latch data; values 0-7 indicate 0-7 registers
5	R/W	0x0	Controls number of refreshes after SDRAM is taken out of self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just 1 row before entering self-refresh mode
4	R/W	0x0	Controls number of refreshes before putting SDRAM into self-refresh mode: 1 – Refresh all rows before entering self-refresh mode 0 – Refresh just one row before entering self-refresh mode
3	R/W	0x1	Determines when row is precharged: 0 – Immediate precharge; row precharged at end of read/write operation 1 – Delayed precharge; row kept open after read/write operations
2	R/W	0x0	Forces put SDRAM in power-down mode; bit 19 determines the type of power-down mode requested
1	R/W	0x0	Forces put SDRAM in self-refresh mode. Bit can be cleared by writing to this bit or by clear_sr_dp pin, generated by external power management unit
0	R/W	0x1	Forces initialize SDRAM; bit reset to 0 by MSDR_memctl once initialization sequence is complete

MSDR_SREFR

Address: Operational Base + offset (0x10)
SDRAM Refresh Interval Register

bit	Attr	Reset Value	Description
31:24	R	0x0	Reserved.
23:16	RW	0x0	Reserved.
15:0	RW	0x410	Number of clock cycles between consecutive refresh cycles;

MSDR_EXN_MODE_REG
Address: Operational Base + offset (0xAC)
Extended Mode Register

bit	Attr	Reset Value	Description
31:13	-	-	Reserved.
12:7	RW	0x0	Always set to zero
6:5	RW	0x0	Driver strength of SDRAM output drivers. 0 - Full 1 - 1/2 2 - 1/4 (optional) 3 - 1/8 (optional)
4:3	RW	0x0	Maximum case temperature 0 – 70 1 – 45 2 – 15 3 – 85
2:0	RW	0x0	Self refresh coverage 0 – 4 banks 1 – 2 banks 2 – 1 bank 5 – 1/2 bank 6 – 1/4 bank Others – reserved

MSDR_SCSLR0
Address: Operational Base + offset (0x14)
chip_select_register0

bit	Attr	Reset Value	Description
31:16	R	0x0000	Upper 16bits of base address for static memory bank0
15:0	-	-	Reserved.

MSDR_SCSLR1
Address: Operational Base + offset(0x18)
chip_select_register1

bit	Attr	Reset Value	Description
31:16	R	0x5100	Upper 16bits of base address for static memory bank1
15:0	-	-	Reserved.

MSDR_SCSLR2
Address: Operational Base + offset (0x1C)
chip_select_register0

bit	Attr	Reset Value	Description
31:16	R	0x6000	Upper 16bits of base address for Mobile SDRAM
15:0	-	-	Reserved.

MSDR_SMSKR0
Address: Operational Base + offset (0x54)
Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	R	0x1	Register determines which timing parameters of memory connect to static memory bank0; primarily used for specifying static memories 0 – register set 0 , set in MSDR_SMTMGR_SET0 1 – register set 1 , set in MSDR_SMTMGR_SET1 2 – register set 2 , set in MSDR_SMTMGR_SET2
7:5	R	0x2	Type of memory connected to static memory bank0: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R	0x9	size of memory connected to static memory bank0; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

MSDR_SMSKR1
Address: Operational Base + offset (0x58)
Address Mask Registers

bit	Attr	Reset Value	Description
31:11	-	-	Reserved.
10:8	R	0x2	Register determines which timing parameters of memory connect to static memory bank1; primarily used for specifying static memories 0 – register set 0 , set in MSDR_SMTMGR_SET0 1 – register set 1 , set in MSDR_SMTMGR_SET1 2 – register set 2 , set in MSDR_SMTMGR_SET2
7:5	R	0x2	Type of memory connected to static memory bank1: 0 – Reserved 1 – SRAM 2 – FLASH Others – Reserved
4:0	R	0x9	size of memory connected to static memory bank1; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10~17 – Reserved

MSDR_SMSKR2
Address: Operational Base + offset (0x5C)

Address Mask Registers

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:5	R	0x0	Type of memory connected to SDRAM 0 – SDRAM Others – Reserved
4:0	R	0xc	size of memory connected to SDRAM; 0 – No memory is connected to the chip select 1– 64KB 2 – 128KB 3 – 256KB 4 – 512KB 5 – 1MB 6 – 2MB 7 – 4MB 8 – 8MB 9 – 16MB 10 – 32MB 11 – 64MB 12 – 128MB 13~17 – Reserved

MSDR_CSREMAP0_LOW
Address: Operational Base + offset (0x84)
REMAP REGISTER0

bit	Attr	Reset Value	Description
31:16	R	0x5000	Represent lower remap register bits for chip select0; compared with corresponding AHB address to generate chip select0; number of compared bits depends on size of memory selected by chip select0 (specified in mask register): 64KB – bits 31:16 compared 128 KB – bits 31:17 compared
15:0	-	-	Reserved.

MSDR_SMTMGR_SET0
Address: Operational Base + offset (0x94)
Static Memory Timing Register - Set0

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set0
27	RW	0x0	Valid if register set0 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device:

			0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x0	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x5	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0xc	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

MSDR_SMTMGR_SET1

Address: Operational Base + offset (0x98)

Static Memory Timing Register – Set1

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set1
27	RW	0x0	Valid if register set1 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0xf	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x1	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold

			time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

MSDR_SMTMGR_SET2
Address: Operational Base + offset (0x9C)
Static Memory Timing Register – Set2

bit	Attr	Reset Value	Description
31:30	-	-	Reserved.
29:28	RW	0x0	Number of registers inserted in the read data path for latching the data correctly, in the case of Static memory associated with register set2
27	RW	0x0	Valid if register set2 is used to control low-frequency synchronous device; instructs the memory controller to sample sm_clken before starting any Static memory operation Synchronous memory device could be same or sub-multiple of AMBA clock
26	RW	0x0	Reserved
25:24	RW	0x0	Page size: 0 – 4-word page 1 – 8-word page 2 – 16-word page 3 – 32-word page
23	RW	0x0	Page-mode device: 0 – device does not support page mode 1 – device supports page mode
22:19	RW	0x3	Page mode read cycle time; values of 0-15 correspond to read cycle time of 1-16 clock cycles
18:16	RW	0x4	Static memory idle cycles between “read to write”, or “write to read”, or “read to read” when chip-select changes for memory data bus turn around time; values of 0-7 correspond to 0-7 idle clock cycles
15:10	RW	0x6	Write pulse width; values of 0-63 correspond to write pulse width of 1-64 clock cycles
9:8	RW	0x1	Write address/data hold time; values of 0-3 correspond to write address/data hold time of 0-3 clock cycles
7:6	RW	0x1	Write address setup time; values of 0-3 correspond to address setup time of 0-3 clock cycles; value of 0 is only valid in case of SSRAM
5:0	RW	0x10	Read cycle time; values of 0-63 correspond to read cycle time of 1-64 clock cycles

MSDR_FLASH_TRPDR
Address: Operational Base + offset (0xA0)
FLASH Timing Register

bit	Attr	Reset Value	Description
31:12	-	-	Reserved.

11:0	RW	0xC8	FLASH reset/power-down high to read/write delay; values correspond to sm_rp_n high to read/write delay minus one

MSDR_SMCTLR
Address: Operational Base + offset (0xA4)
Static Memory Control Register

bit	Attr	Reset Value	Description
31:16	-	-	Reserved.
15:13	RW	0x0	Width of Static memory data bus controlled by Static memory register SET2: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
12:10	RW	0x4	Width of Static memory data bus controlled by Static memory register SET1: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
9:7	RW	0x4	Width of Static memory data bus controlled by Static memory register set 0: 000 – 16 bits 001 – 32 bits 010 – 64 bits 011 – 128 bits 100 – 8 bits
6:4	-	-	Reserved.
3:1	RW	0x0	FLASH write-protection mode; writing 0 forces FLASH memory bootblock to write protect; the three bits correspond to three register sets
0	RW	0x1	FLASH reset/power-down mode; after reset, controller internally performs a power-down for FLASH and then sets this bit to 1 to force FLASH to power-down mode during normal operation: 0 – Forces FLASH to power-down mode 1 – Takes FLASH out of power-down mode

5.4 Functional Description

5.4.1 Operation

Basic Access Operation

1. A page-hit sigle-cycle write:

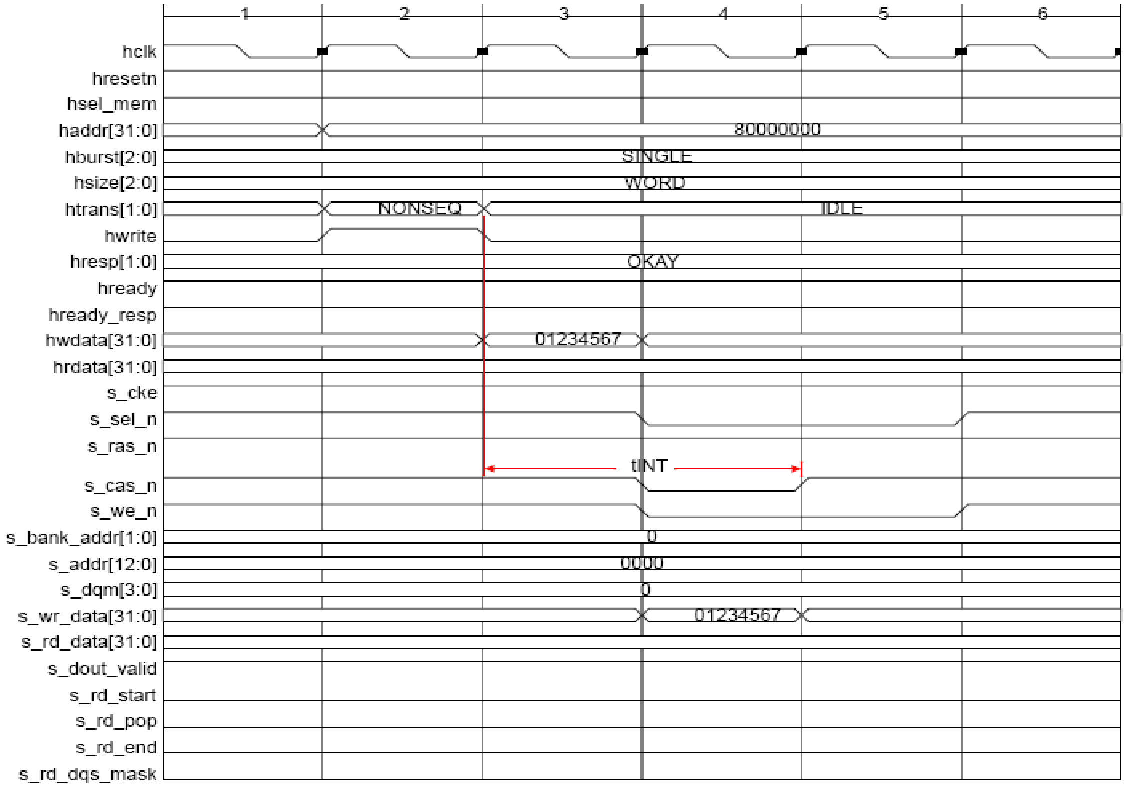


Fig. 5-2 Mobile SDRAM Page-Hit Single Write

2、A page-miss single-cycle write：

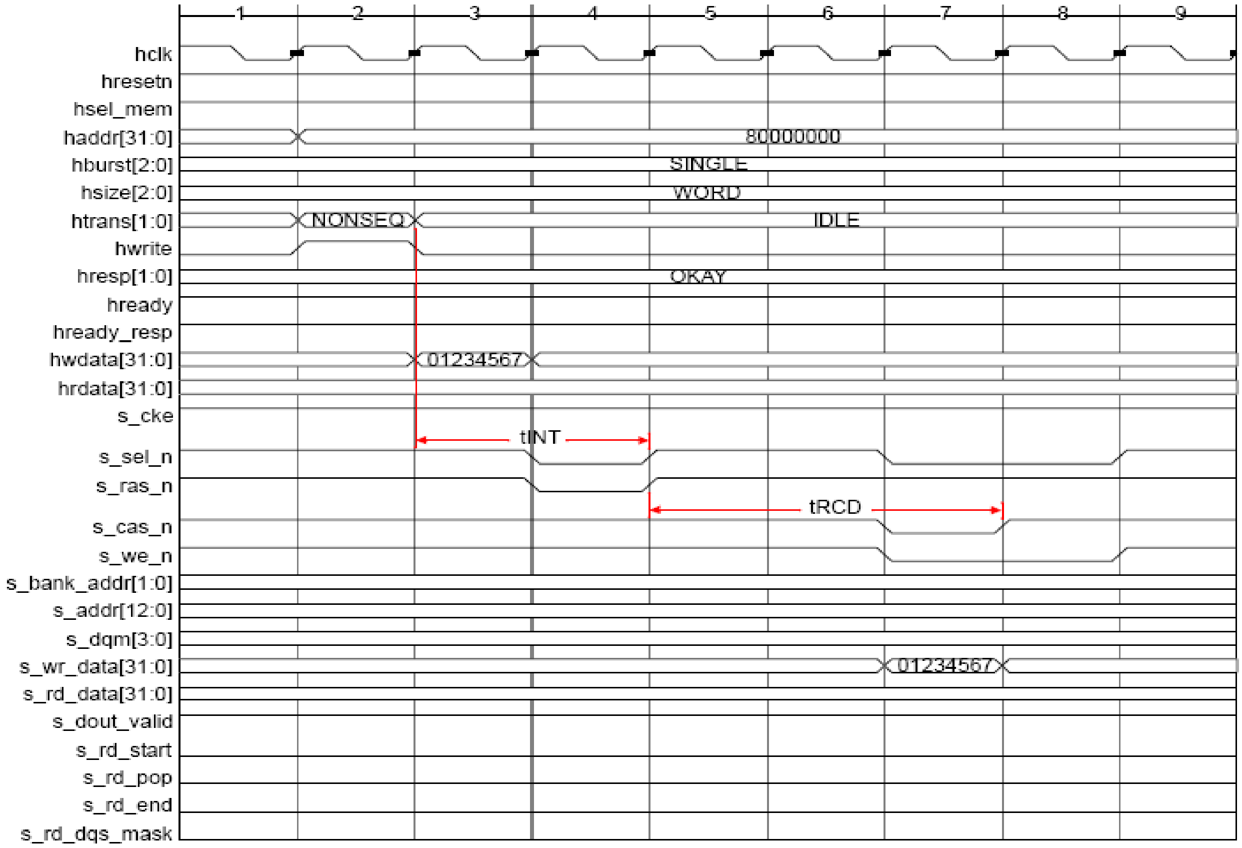


Fig. 5-3 Mobile SDRAM Page-Miss Single Write

3 、A page-hit burst write (hburst == INCR8)

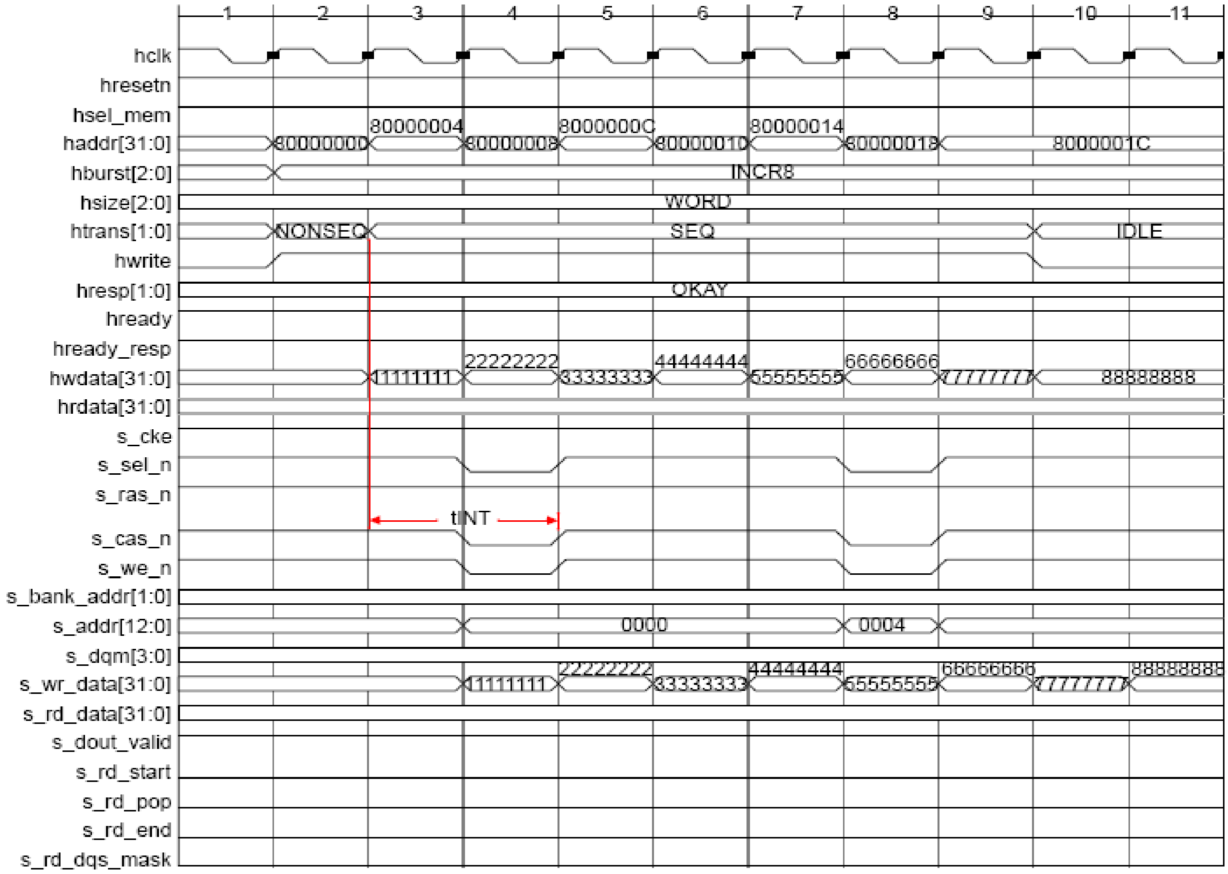


Fig. 5-4 Mobile SDRAM Page-Hit Busrt Write

4、A page-hit single-cycle read:

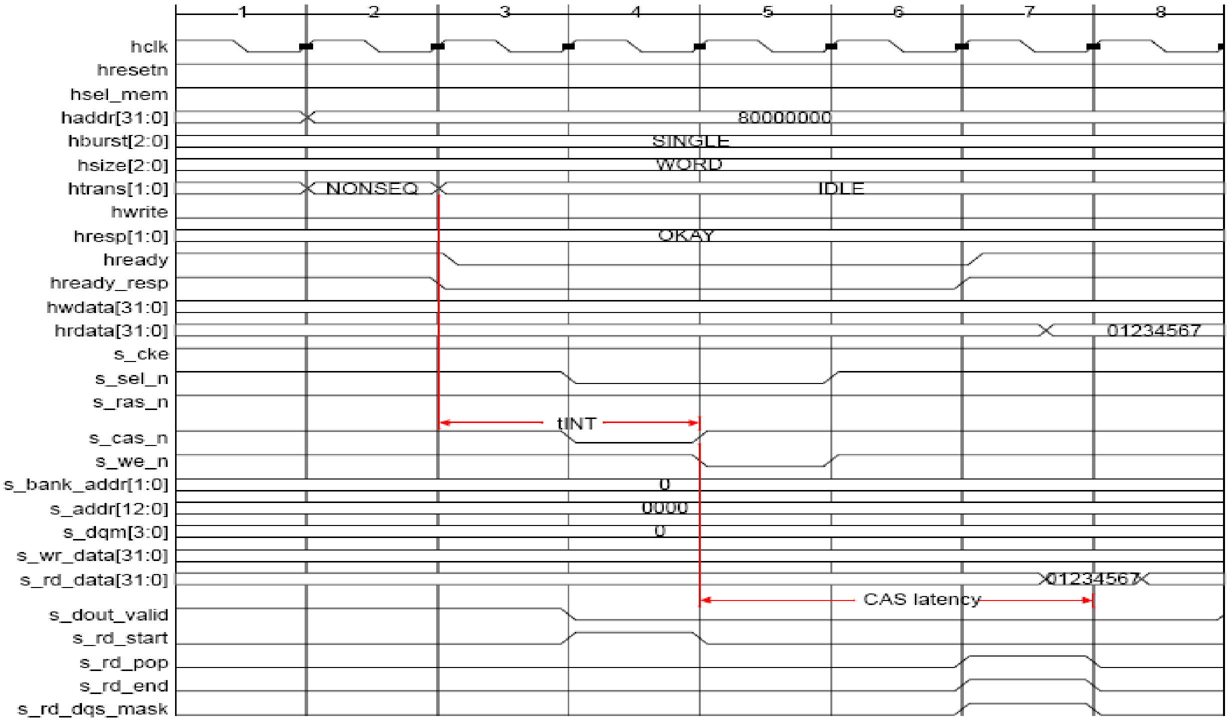


Fig. 5-5 Mobile SDRAM Page-Hit Single Read

5、a page-miss single-cycle read:

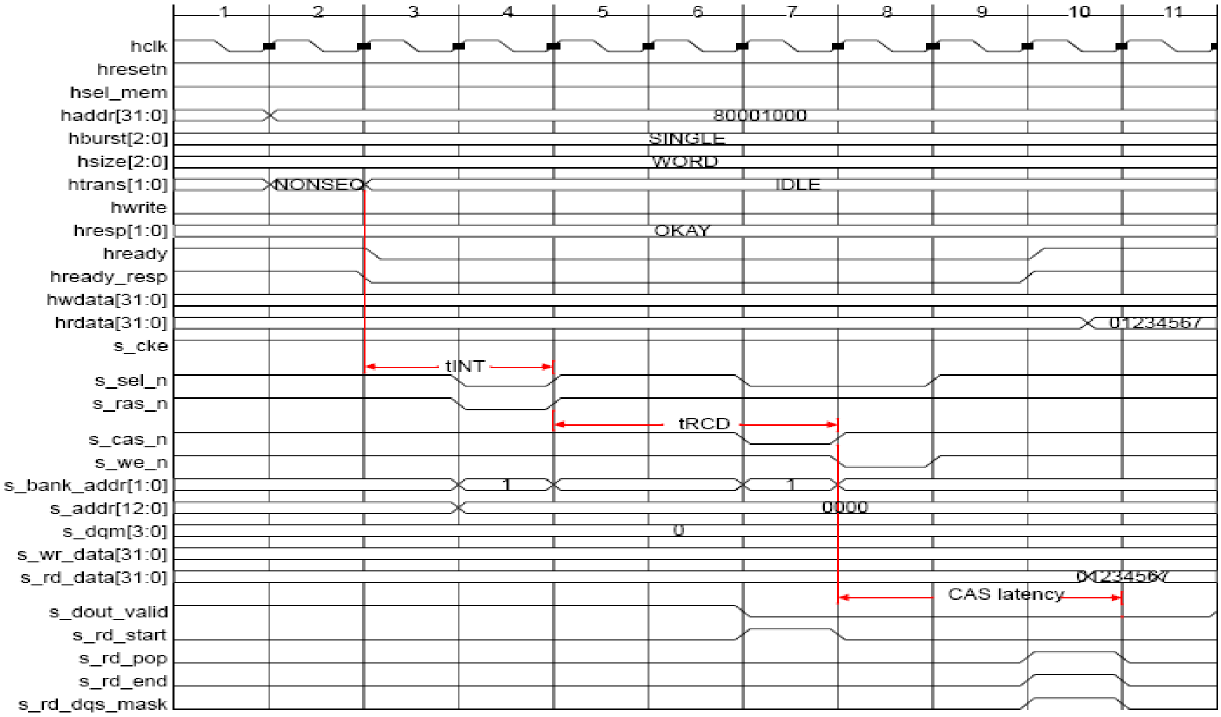


Fig. 5-6 Mobile SDRAM Page-Miss Single Read

6、A page-hit burst read:

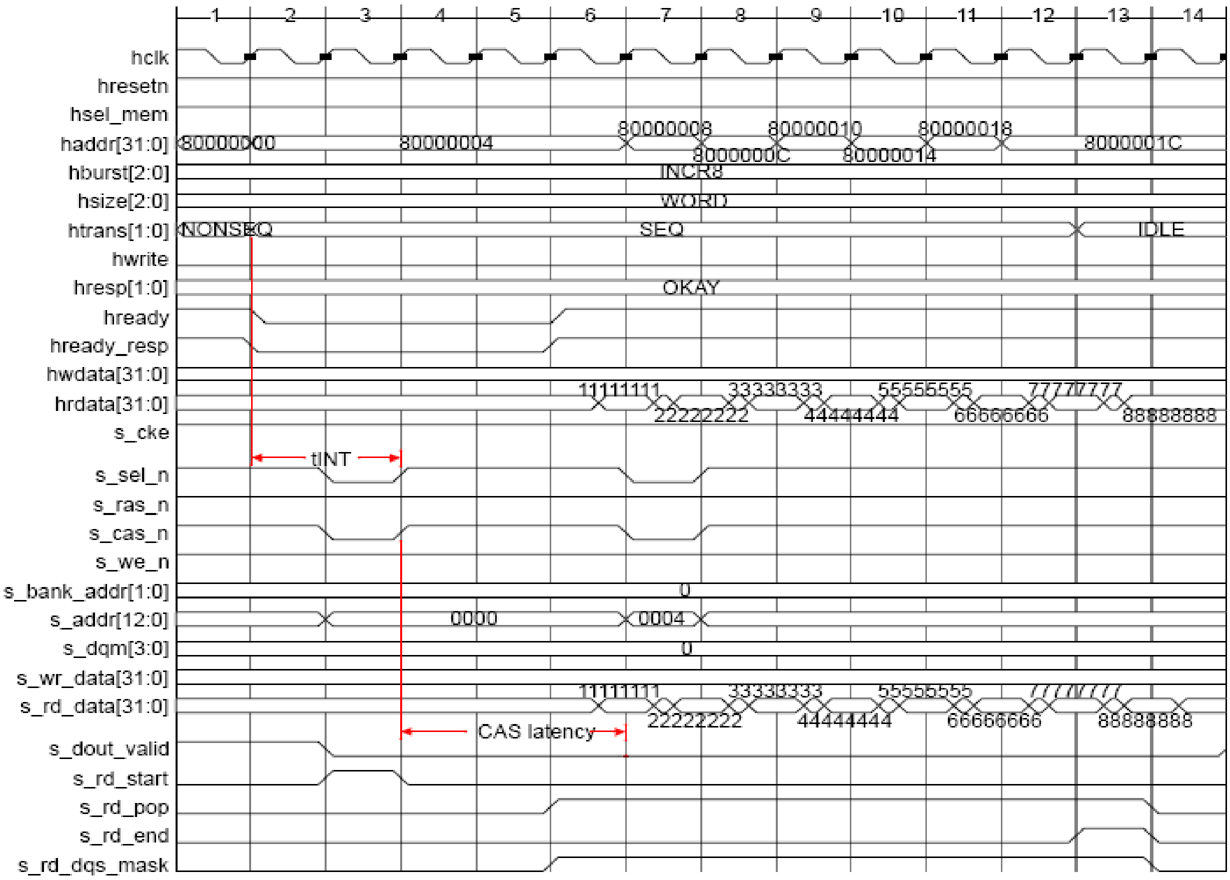


Fig. 5-7 Mobile SDRAM Page-Hit Busrt Read

Power-On Initialization

The SDR-SDRAM controller follows the JEDEC-recommended SDR-SDRAM power-on initialization sequence as follows:

1. Apply power and start clock; maintain a NOP condition at the inputs

2. Maintain stable power, stable clock, and NOP input conditions for a minimum of t_init clock cycles

3. Issue precharge commands for all banks of the device

4. Issue auto-refresh commands, depending on the value num_init_ref in the programmable register

5. Issue a set-mode register command to initialize the mode registe the commands issued to the SDRAM by the controller during the power-on initialization are shown as followed

6. Issue a set-extended-mode register command to initialize the extended-mode register (valid only for Mobile-SDRAM)

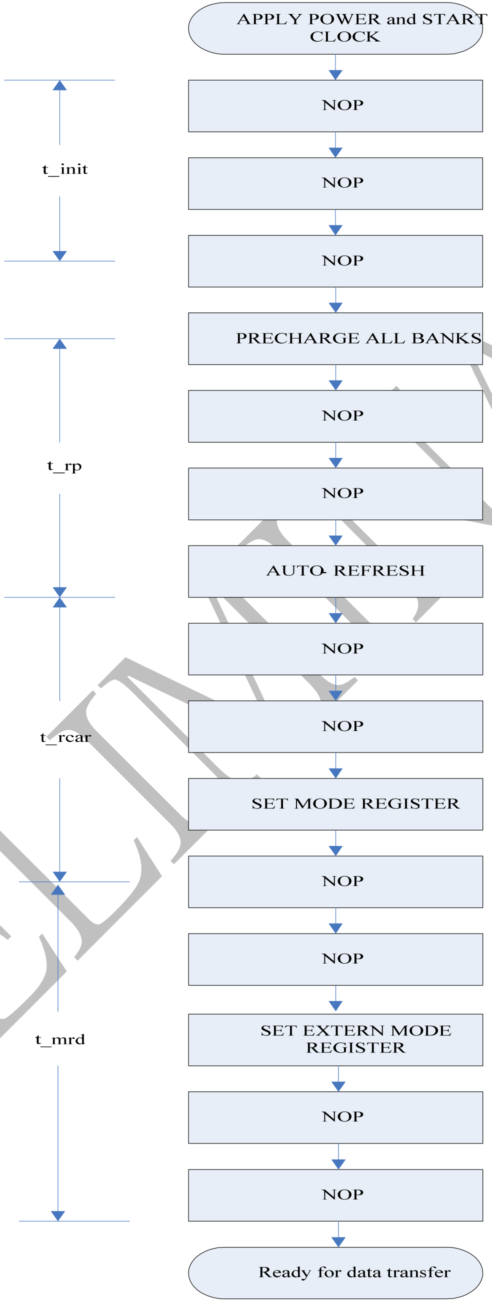


Fig. 5-8 Static Memory/Mobile SDRAM Controller power on sequence

Read / Write Operation

The SDRAM Controller converts all AHB bursts to 4-word bursts on the SDRAM side. The memory bursts are concatenated to achieve continuous data flow for long AHB bursts. You can terminate the memory read/write burst with either a precharge command or terminate command, depending on which precharge mode – immediate precharge or delayed precharge – that you program. You can also terminate the write burst with a subsequent write burst

The SDRAM Controller supports two precharge modes – immediate precharge and delayed precharge. If you program for an immediate precharge mode, then the SDRAM Controller closes the open row after a read or write access. If you program for a delayed precharge mode, then the MSDR_memctl keeps the row open after an access. The MSDR_memctl can keep multiple banks open at the same time, depending on the value of num_open_bank in the programmable register. When the number of open banks reaches the num_open_bank and an access to a new bank comes, the SDRAM Controller will close the oldest bank (the bank opened first) before opening the new bank.

Set-Mode and Extended-Mode Register

The SDRAM Controller automatically sets the SDR-SDRAM mode register and extended-mode register during the power-up initialization. During normal operation, if you want to set the mode register or extended-mode register, you need to set set_mode_reg (bit 9 of SCTLR) or exn_mode_reg_update (bit 18 of SCTLR) in the control register (SCTLR) to 1. After the memory controller finishes the mode register setting, it clears the set_mode_reg or the exn_mode_reg_update bit to 0.

The "burst length" field and the "burst type" field of the SDR-SDRAM-mode register are fixed by the SDRAM Controller to "010" (burst length 4) and "0" (sequential burst), respectively. The SDRAM Controller programs the "CAS latency" field and the "operating mode" field of the mode register according to the values provided by the user in the control and timing registers.

The MSDR_memctl programs the extended-mode register of the Mobile-SDRAM according to the value provided by the user in the EXTN_MODE_REG (address 32'hxxxx_xxAC).

.Auto-Refresh

Auto-refresh commands are issued when the refresh control block issues refresh requests. During normal refresh operations, the SDRAM Controller always refreshes one row at a time. It is important for the user to program the tREF refresh interval register after a reset. If you need to refresh the SDRAM while a burst is active, normally the SDRAM Controller will issue the refresh command after the ongoing burst completes. However, if the ongoing burst is an AHB INCR burst, the SDRAM Controller will stop the burst, issue the refresh command, and then resume the burst.

The SDRAM Controller takes into account the maximum time it takes to complete a worst-case burst. This is the time to complete a read burst corresponding to an INCR16 burst on the AMBA bus, and with an AMBA-to-SDRAM data width ratio of 2:1. It is reasonable to assume 50 cycles for this worst-case burst, with 32 cycles for the data and the remaining 17 cycles for various latencies for the worst case.

The t_ref value can be calculated using the following equation:

t_ref = refresh_period / clock_period

where refresh_period = typically 7.8/15.6 s.

The tREF is the value of a free-running counter that the refresh logic in the SDRAM Controller operates on. When the count expires, the refresh logic gives a refresh request to the SDRAM controller.

Since the 64 ms refresh period is the same for most SDRAMs, the total number of rows in the SDRAM limits the minimum operating frequency for the MSDR_memctl. While calculating the minimum frequency, use the following equation:

tREF > 50*(1/f).

Typically, the refresh cycle is 15.6 s or 7.8 s, depending on the refresh rate; The table is summarized as followed:

Number of rows	tREF	Min Frequency
64K	(64ms-(50/f))/65536	51Mhz
32K	(64ms-(50/f))/32768	26Mhz
16K	(64ms-(50/f))/163904	13Mhz
8K	(64ms-(50/f))/8192	6Mhz
4K	(64ms-(50/f))/4096	3Mhz
2K	(64ms-(50/f))/2048	1.5Mhz

The refresh logic in the MSDR_memctl is inactive when the MSDR_memctl forces the

SDRAM into self-refresh or power-down mode.

Self-Refresh

You can put the SDRAM into self-refresh mode, at which point the SDRAM retains data without external clocking and auto-refresh. The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit the self-refresh mode

You can force the SDRAM Controller to enter self-refresh mode by programming bit 1 of the SDRAM control register (SCTLR) (Address 32'hxxxx_xx0C). The SDRAM Controller forces the SDRAM to come out of self-refresh mode when bit 2 of the SCTLR is set to 0. You can set this bit to 0 by either programming the SDRAM control register or driving the clear_sr_dp pin high. You can use the clear_sr_dp pin when the code resides in the SDRAM, and the SDRAM itself is in self-refresh mode.

Bits 4 and 5 of the SCTLR specify the type of refresh done by the SDRAM Controller just prior to entering self-refresh mode and just after entering self-refresh mode. Programming bit 4 of the SCTLR to 0 forces the SDRAM Controller to refresh only one row before putting the SDRAM into self-refresh mode. The default value of 1 forces the SDRAM Controller to perform auto-refreshes for all rows. Bit 5 does the same, except that it controls the refresh pattern just after coming out of self-refresh mode.

Since it takes time between programming the control register bit to the SDRAM entering self-refresh mode, the SDRAM Controller provides a read-only register bit (bit 11 of the SDRAM control register) to indicate that the SDRAM is already in self-refresh mode. If you want to gate off the clock to the SDRAM Controller when the SDRAM is in self-refresh mode, you should ensure this bit is set to 1 before you stop the clock.

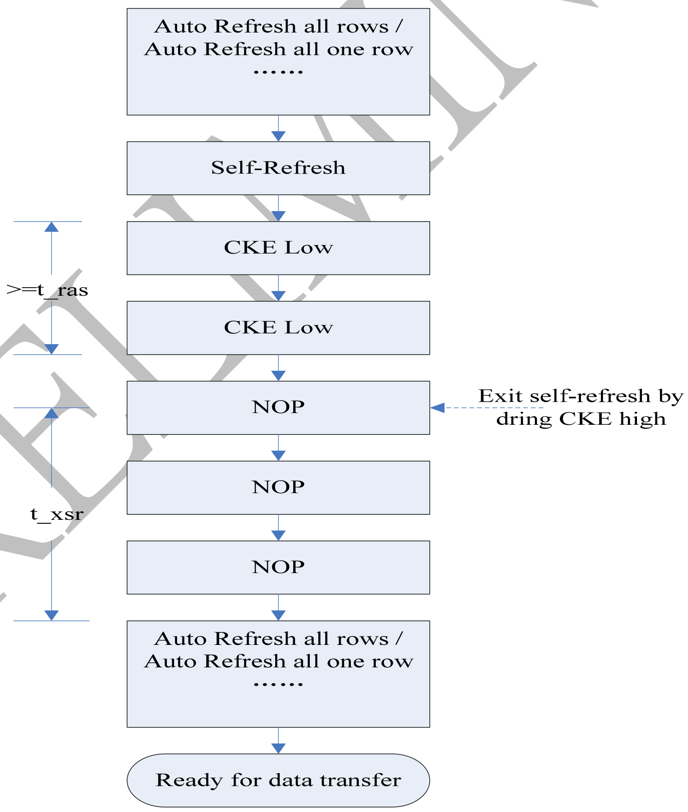


Fig. 5-9 Mobile SDRAM self_reflesh flow

The SDRAM must remain in self-refresh mode for a minimum period of t_ras and can remain in self-refresh mode for an indefinite period of time. After the SDRAM exits self-refresh mode, the SDRAM Controller issues NOP commands for t_xsr before it issues any other command. The t_ras and t_xsr are programmable register values and have default values. These registers can be programmed only once after reset.

When an AHB read/write request to the SDRAM occurs while the SDRAM is in self-refresh mode, the SDRAM Controller generates dummy ready signals to the AHB without accessing external memory; no error response is generated on the AHB bus.

Power-Down

The SDRAM can be put into power-down mode to save power. There are two ways to force the MSDR_memctl to put the SDRAM in power-down mode:

- Program bit 2 of SCTLR to 1; should be 0 to bring the SDRAM out of power-down mode.
- Use the power-down input pin; can be driven by an external power management unit; the SDRAM will be in power-down mode as long as this signal stays high

The Figure as followed illustrates the command sequence issued by the SDRAM controller to initiate, maintain, and exit power-down mode

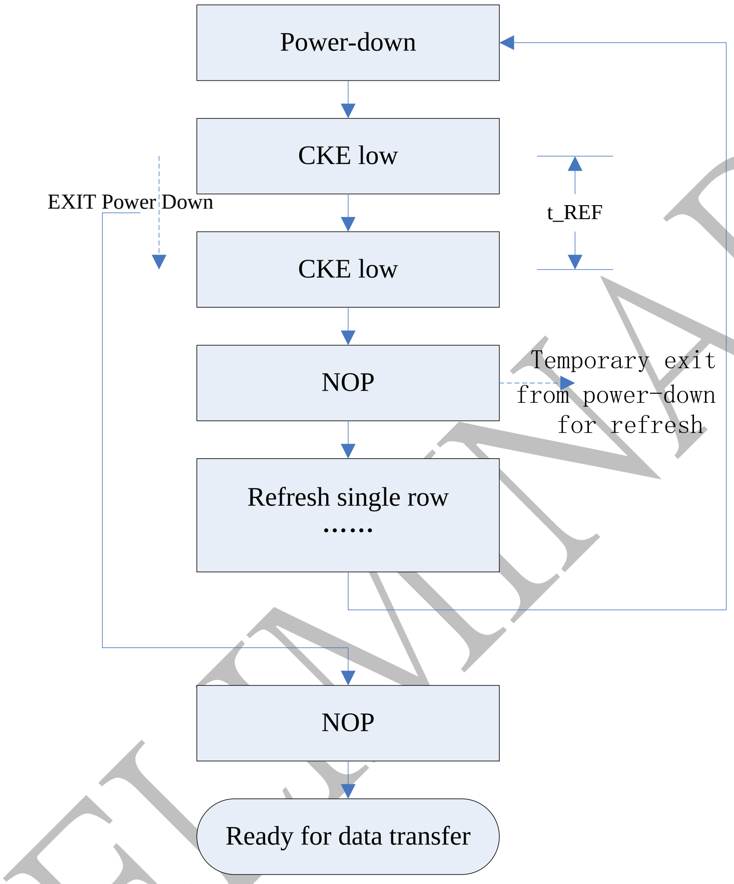


Fig. 5-10 Static Memory/Mobile SDRAM Controller power_off mode

When in SDRAM power-down mode, the SDRAM Controller keeps switching the device back and forth between power-down and refresh mode. It remains in power-down for a t_ref period of time, then comes out of power-down and does a single-row refresh; then it again goes into power-down mode.

The SDRAM Controller keeps the SDRAM in this periodical power-down/refresh/power-down sequence until it is commanded to exit power-down mode (by programming bit 2 of SCTLR to 0) When an AHB read/write request to the SDRAM occurs while the SDRAM is in power-down mode,the SDRAM Controller brings the SDRAM out of power-down mode and issues the read/write access to the SDRAM. The SDRAM Controller then puts the SDRAM back to power-down mode after the read/write access.

Chapter 6 NAND Flash Controller

6.1 Design Overview

6.1.1 Overview

The Nand Flash Controller (NANDC) is used for controlling data transfer to/from nand flash device. Control information is written from a master (CPU) over the AHB bus to the NANDC. It supports transferring data to/from flash in two ways:internal dma and directly bypass. Hardware ECC ,which support 8bits or 14bits bch error correction ,can correct error bits at any position in one codeword.

6.1.2 Features

- AMBA AHB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- Support interrupt interface to interrupt controller.
- 2K sram memory space, divided to 4 sections, used for internal dma transfer.
- 8bits data interface to flash device
- Support 4 flash devices at most.
- Hardware ECC error correction.
- Support FF code auto correction.
- Support internal dma and directly bypass ways to the flash device.
- Data can be transferred to/from sram that is free from internal dma use.

For detailed information about NAND FLASH controller, please refer to **RK28xx NAND Flash Controller.pdf**。

Chapter 7 SD/MMC Host Controller

7.1 Design Overview

7.1.1 Overview

The SD/MMC Host Controller is designed to support Secure Digital memory (SD mem - version 2.00), Secure Digital I/O (SDIO-version 1.10), Multimedia Cards (MMC-version 4.2). There are two SD/MMC Host Controllers connected with ARMD bus, SDMMC0 support SD Card(1/4bit), SDIO, MMC(1/4/8bit), and SDMMC1 support SD Card(1/4bit), SDIO, MMC(1/4bit).

7.1.2 Features

- Supports AMBA AHB interface
- Supports DMA controller for data transfers
- Supports interrupt output
- Supports SD version2.0 except SPI mode
- Supports MMC version4.2 except SPI mode
- Supports SDIO version1.1
- Supports programmable baud rate.
- Provides individual clock control to selectively turn ON or OFF clock to a card
- Supports power management and power switch. Provides individual power control to selectively turn ON or OFF power to a card

7.2 Architecture

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

7.2.1 Block Diagram

The SD/MMC controller consists of the following main functional blocks, which are illustrated in Fig. 6-1.

- Bus Interface Unit (BIU) – Provides AMBA AHB and DMA interfaces for register and data read/writes.
- Card Interface Unit (CIU) – Takes care of the SD_MMC protocols and provides clock management.

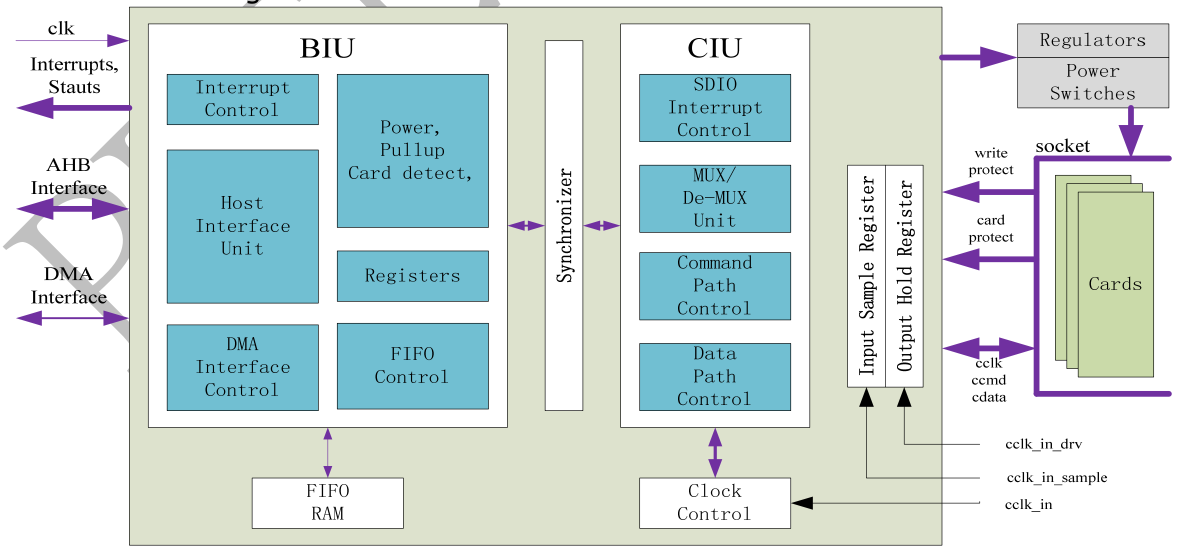


Fig. 7-1 SD/MMC Host Controller Block Diagram

7.2.2 Block Descriptions

The SD/MMC comprises with:

BIU(Bus Interface Unit)

The BIU provides the following functions:

Host Interface

The Host Interface Unit (HIU) is an AHB slave interface, which provides the interface between the SD/MMC controller and the host bus.

DMA Interface

DMA signals interface the SD/MMC controller to an external AHB DMA controller to reduce the software overhead during FIFO data transfers. The DMA request/acknowledge handshake is used for only data transfer. The DMA interface provides a connection to the DW_dmac.

On seeing the DMA request, the DMA controller initiates accesses through the host interface to read or write into the data FIFO. The SD/MMC controller has FIFO transmit/receive watermark registers that you can set, depending on system latency. The DMA interface asserts the request in the following case:

- Read from a card when the data FIFO word count exceeds the Rx-Watermark level
- Write to a card when the FIFO word count is less than or equal to the Tx-Watermark level

When the DMA interface is enabled, you can use normal host read/writes to access the data FIFOs.

Interrupt Control

The interrupt controller unit generates an interrupt that depends on the controller raw interrupt status, the interrupt-mask register, and the global interrupt-enable register bit. Once an interrupt condition is detected, it sets the corresponding interrupt bit in the raw interrupt status register. The raw interrupt status bit stays on until the software clears the bit by writing a 1 to the interrupt bit; a 0 leaves the bit untouched.

The interrupt port is active only when any bit in the raw interrupt status register is active, the corresponding interrupt mask bit is 1, and the global interrupt enable bit is 1.

Notes:

* Before enabling the interrupt, it is always recommended that you write 32'hffff_ffff to the raw interrupt status register in order to clear any pending unserviced interrupts. When clearing interrupts during normal operation, ensure that you clear only the interrupt bits that you serviced.

Table 7-1 SD/MMC Bits in Interrupt Status Register

bit	Interrupt	Description
31:16	SDIO Interrupts	Interrupts from SDIO cards; one bit for each card. Bit[31] corresponds to Card[15].
15	End Bit Error (read)/Write no CRC (EBE)	Error in end-bit during read operation, or no data CRC or negative CRC received during write operation. Note: For MMC CMD19, there may be no CRC status returned by the card. Hence, EBE is set for CMD19. The application should not treat this as an error.
14	Auto Command Done (ACD)	Stop/abort commands automatically sent by card unit and not initiated by host; similar to Command Done (CD) interrupt. Attention – Recommendation: Software typically need not enable this; Data Transfer Over (DTO) interrupt that comes after this interrupt determines whether data transfer has correctly completed.
13	Start Bit Error (SBE)	Error in data start bit when data is read from a card. In 4-bit mode, if all data bits do not have stat bit, then this error is set.
12	Hardware Locked write Error (HLE)	During hardware-lock period, write attempted to one of locked registers.
11	FIFO Underrun/Overrun Error (FRUN)	Host tried to push data when FIFO is full, or host tried to read data when FIFO was empty. Typically this should not happen, except due to error in

		software. Card unit never pushes data into FIFO when FIFO is full, and pop data when FIFO is empty.
10	Data Starvation by Host Timeout (HTO)	To avoid data loss, card clock out (cclk_out) is stopped if FIFO is empty when writing to card, or FIFO is full when reading from card. Whenever card clock is stopped to avoid data loss, data-starvation timeout counter is started with data-timeout value. This interrupt is set if host does not fill data FIFO during write to card, or does not read from FIFO during read from card before timeout period. Even after timeout, card clock stays in stopped state, with CIU state machines waiting. It is responsibility of host to push or pop data into FIFO upon interrupt, which automatically restarts cclk_out and card state machines. Even if host wants to send stop/abort command, it still needs to ensure it has to push or pop FIFO so that clock starts in order for stop/abort command to send on cmd signal along with data that is sent or received on data line.
9	Data Read Timeout(DRTO)	Data timeout occurred. Data Transfer Over (DTO) also set if data timeout occurs.
8	Response Timeout(RTO)	Response timeout occurred. Command Done (CD) also set if response timeout occurs. If command involves data transfer and when response times out, not data transfer is attempted by SD/MMC controller.
7	Data CRC Error (DCRC)	Received Data CRC does not match with locally-generated CRC in CIU.
6	Response CRC Error (RCRC)	Response CRC does not match with locally-generated CRC in CIU.
5	Receive FIFO Data Request (RXDR)	Interrupt set during read operation from card when FIFO level is greater than Receive-Threshold level. Attention – Recommendation: In DMA modes, this interrupt should not be enabled. ISR, in non-DMA mode: pop RX_WMark + 1 data from FIFO
4	Transmit FIFO Data Request (TXDR)	Interrupt set during write operation to card when FIFO level reaches less than or equal to Transmit-Threshold level. Attention – Recommendation: In DMA modes, this interrupt should not be enabled. ISR in non-DMA mode: if (pending_bytes >(FIFO_DEPTH – TX_WMark)) push (FIFO_DEPTH – TX_WMark) data into FIFO else push pending_bytes data into FIFO
3	Data Transfer Over (DTO)	Data transfer completed, even if there is Start Bit Error or CRC error. Attention – Recommendation: In non-DMA mode, when data is read from card, on seeing interrupt, host should read any pending data from FIFO. In DMA mode, DMA controllers guarantee FIFO is flushed before interrupt. Note – DTO bit is set at the end of the last data

		block, even if the device asserts MMC busy after the last data block.
2	Command Done (CD)	Command sent to card and got response from card, even if Response Error or CRC error occurs.
1	Response Error (RE)	Error in received response set if one of following occurs: ● Transmission bit !=0 ● Command index mismatch ● End-bit !=1
0	Card-Detect (CDT)	When one or more cards inserted or removed, this interrupt occurs. Software should read card-detect register (CDETECT, 0x50) to determine current card status. Attention – Recommendation: After power-on and before enabling interrupts, software should read card detect register and store it in memory. When interrupt occurs, it should read card detect register and compare it with value stored in memory to determine which card were removed/inserted. Before exiting ISR, software should update memory with new card-detect value.

Note – The SDIO Interrupts, Receive FIFO Data Request (RXDR), and Transmit FIFO Data Request (TXDR) are set by level-sensitive interrupt sources. Therefore, the interrupt source should be first cleared before you can clear the interrupt bit of the Raw Interrupt register. For example, on seeing the Receive FIFO Data Request(RXDR) interrupt, the FIFO should be emptied so that the "FIFO count greater than the RX-Watermark" condition, which triggers the interrupt, becomes inactive. The reset of the interrupts are triggered by a single clock-pulse-width source.

Register Bank

The register unit is part of the bus interface unit (BIU); it provides read and write access to the registers.

All registers reside in the BIU clock domain. When a command is sent to a card by setting the start_bit, which is bit[31] of the CMD register, all relevant registers needed for the CIU operation are transferred to the CIU block. During this time, the register that are transferred from the BIU to the CIU should not be written. The software should wait for the hardware to clear the start bit before writing to these registers again.

Once a command start is issued by setting the start_bit of the CMD register, the following registers cannot be reprogrammed until the command is accepted by the CIU: CMD/CMDARG/BYTCNT/BLKSIZ/CLKDIV/CLKENA/CLKSRC/TMOUT/CTYPE.

The hardware resets the start_bit once the CIU accepts the command. If a host write to any of these registers is attempted during this lock time, then the write ignored and the hardware lock error bit is set in the raw interrupt status register. Additionally, if the interrupt is enabled and not masked for a hardware lock error, then an interrupt is sent to the host.

When the CIU is in an idle state, it typically takes the following number of clocks for the command handshake, where clk is the BIU clock and cclk_in is the CIU clock: 3(clk) + 3(cclk_in)

Once a command is accepted, you can send another command to the CIU – which has a one-deep command queue – under the following conditions:

- If the previous command was not a data transfer command, the new command is sent to the SD_MMC card once the previous command completes.
- If the previous command is a data transfer command and if wait_prvdata_complete (bit[13]) of the CMD register is set for the new command, the new command is sent to the SD_MMC card only when the data transfer completes.
- If the wait_prvdata_complete is 0, then the new command is sent to the SD_MMC card as soon as the previous command is sent. Typically, you should use this only

to stop or abort a previous data transfer or query the card status in the middle of a data transfer.

FIFO Controller Unit

The FIFO controller interfaces the internal FIFO to the host/DMA interface and the card controller unit.

The FIFO maps to an address offset that is greater than or equal to 0x100. While accessing the FIFO, you can set the address to any value 0x100 or greater; partial access to the FIFO is also supported. If the AHB is 32 bits, you can access the FIFO with two 16-bit accesses. The lower address should be accessed first, and then the higher address, such as 0x100, and then 0x101.

Power/Pullup control and card detection Unit

The register unit has registers that control the power and MMC open-drain pullup. Power to each card can be selectively turned on or off. Additionally, there are two 4-bit card-voltage control signals that can control the voltages of two voltage regulators. The control register has an enable_OD_pullup bit, which is used to turn on the open-drain-mode pullup during MMC initialization.

The card detection unit looks for any changes in the card-detect signals for card insertion or card removal. It filters out the debounces associated with mechanical insertion or removal, and generates one interrupt to the host. You can program the debounce filter value.

On power-on, the controller should read in the card_detect port and store the value in the memory. Upon receiving a card-detect interrupt, it should again read the card_detect port and XOR with the previous card-detect status to find out which card has interrupted.

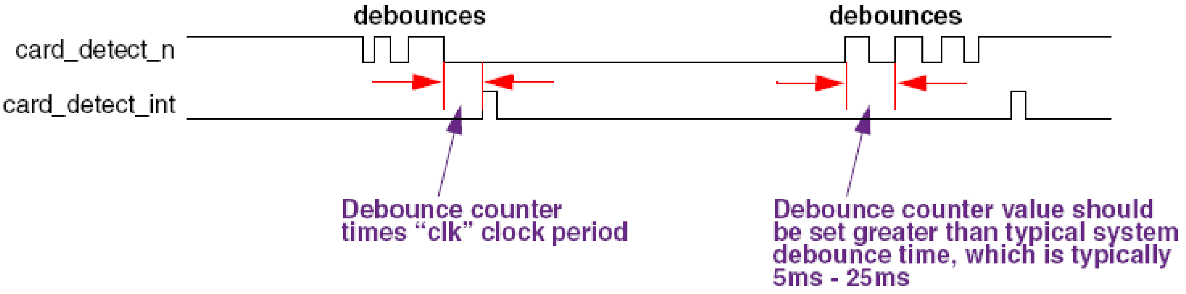

debounces
debounces
card_detect_n
card_detect_int
Debounce counter times "clk" clock period
Debounce counter value should be set greater than typical system debounce time, which is typically 5ms - 25ms

Fig. 7-2 SD/MMC Card Detect timing waveform

CIU(Card Interface Unit)

The Card Interface Unit(CIU) interfaces with the BIU and the SD_MMC card or devices. The host writes command parameters to the SD/MMC controller BIU control registers, and these parameters are then passed to the CIU. Depending on control register values, the CIU generates SD/MMC command and data traffic on a selected card bus according to SD/MMC protocol.

The following software restrictions should be met for proper CIU operation:

- Only one data transfer command can be issued at a time.
- During an open-ended card write operation, if the card clock is stopped because the FIFO is empty, the software must first the data into the FIFO and start the card clock. It can then issue only a stop/abort command to the card.
- During an SDIO card transfer, if the card function is suspended and the software wants to resume the suspended transfer, it must first reset the FIFO and start the resume command as if it were a new data transfer command.
- When issuing card reset commands (CMD0, CMD15 or CMD52_reset) while a card data transfer is in progress, the software must set the stop_abort_cmd bit in the CMD register so that the SD/MMC controller can stop the data transfer after issuing the card reset command.
- When the data end bit error is set in the RINTSTS register, the SD/MMC controller does not guarantee SDIO interrupts. The software should ignore the SDIO interrupts and issue the stop/abort command to the card, so that the card stops sending the read data.

- If the card clock is stopped because the FIFO is full during a card read, the software should read at two FIFO locations to start the card clock.

The CIU block consists of the following primary functional blocks:

Command path

The command path performs the following functions:

- Loads clock parameters
- Loads card command parameters
- Sends commands to card bus
- Receives response from card bus
- Sends responses to BIU
- Drives the P-bit on command line

A new command is issued to the SD/MMC controller by programming the BIU registers and setting the start_cmd bit in the CMD register. The BIU asserts start_cmd, which indicates that a new command is issued to the SD/MMC controller. The command path loads this new command (command. Command argument, timeout) and sends an acknowledge to the BIU by asserting cmd_taken.

Once the new command is loaded, the command path state machine sends a command to the SD_MMC bus-including the internally generated CRC7-and receives a response, if any. The state machine then sends the received response and signals to the BIU that the command is done, and then waits for eight clocks before loading a new command.

1. Load Command Parameters

One of the following commands or responses is loaded in the command path:

- New command from BIU – When start_cmd is asserted, then the start_cmd bit is set in the Command register.
- Internally-generated auto-stop command – When the data path ends, the stop command request is loaded.
- IRQ response with RCA 0x000 – When the command path is waiting for an IRQ response from the MMC card and a "send irq response" request is signaled by the BIU, then the send_irq_response bit is set in the control register.

Loading a new command from the BIU in the command path depends on the following Command register bit settings:

- update_clock_registers_only – If this bit is set in the Command register, the command path updates only the clock enable, clock divider, and clock source registers. If this bit is not set, the command path loads the command, command argument, and timeout registers; it then starts processing the new command.
- wait_prvdata_complete – If this bit is set, the command path loads the new command under one of the following conditions:
    1) Immediately, if the data path is free (that is, there is no data transfer in progress), or if an open-ended data transfer is in progress (byte_count = 0).
    2) After completion of the current data transfer, if a predefined data transfer is in progress.

2. Send Command and Receive Response

Once a new command is loaded in the command path – update_clock_registers_only bit is unset – the command path state machine sends out a command on the SD_MMC bus. The command path state machine is illustrated in Fig. 6-3.

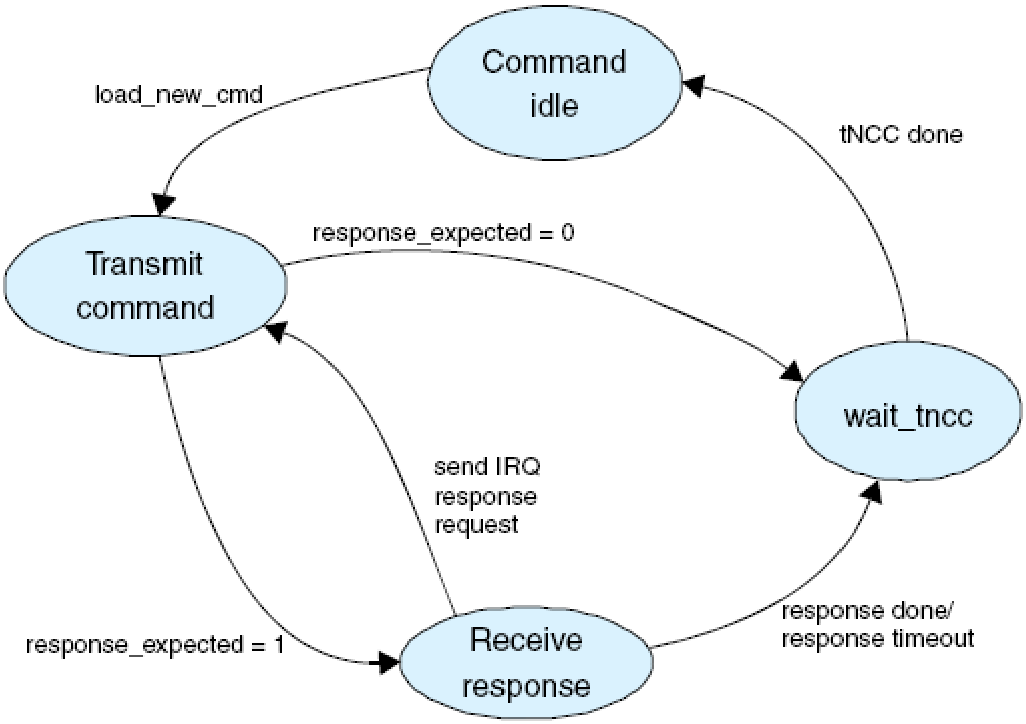


Fig. 7-3 SD/MMC Command Path State Machine

The command path state machine performs the following functions, according to Command register bit values:

- send_initialization – Initialization sequence of 80 clocks is sent before sending the command.
- response_expected – Response is expected for the command. After the command is sent out, the command path state machine receives a 48-bit or 136-bit response and sends it to the BIU. If the start bit of the card response is not received within the number of clocks programmed in the timeout register, then the response timeout and command done bit is set in the Raw Interrupt Status register as a signal to the BIU. If the response-expected bit is not set, the command path sends out a command and signals a response done to the BIU; that is, the command done bit is set in the Raw Interrupt Status register.
- response_length – If this bit is set, a 136-bit response is received; if it is not set, a 48-bit response is received.
- check_response_crc – If this bit is set, the command path compares CRC7 received in the response with the internally-generated CRC7. If the two do not match, the response CRC error is signaled to the BIU; that is, the response CRC error bit is set in the Raw Interrupt Status register.

3. Send Command and Receive Response

If the response_expected bit is set in the Command register, the received response is sent to the BIU. The Response0 register is updated for a short response, and the Response3, Response2, Response1, and Response0 registers are updated on a long response, after which the Command Done bit is set. If the response is for an auto_stop command sent by the CIU, the response is saved in the Response1 register, after which the Auto Command Done bit is set.

Additionally, the command path checks for the following:

- Transmission bit = 0
- Command index matches command index of the sent command
- End bit = 1 in received card response

The command index is not checked for a 136-bit response or if the check_response_crc bit is unset. For a 136-bit response and reserved CRC 48-bit responses, the command index is reserved—that is, 111111.

4. Driving P-bit on CMD line

The command path drives a P-bit = 1 on the CMD line between two commands if a response is not expected. If a response is expected, the P-bit is driven after the response is received and before the start of the next command; this is done by asserting both ccmd_out and ccmd_out_en.

During initialization, the software should set the ccmd_od_pullup_en bit, which indicates an open-drain mode, during which the controller drives only a 0 or high-impedance (Z) on the command bus; a hard 1 is never driven in open-drain mode.

Data Path

The data path block pops the data FIFO and transmits data on cdata_out during a write data transfer, or it receives data on cdata_in and pushes it into the FIFO during a read data transfer. The data path loads new data parameters—that is, data expected, read/write data transfer, stream/block transfer, block size, byte count, card type, timeout registers—whenever a data transfer command is not in progress.

If the data_expected bit is set in the Command register, the new command is a data transfer command and the data path starts one of the following:

- Transmit data if the read/write bit = 1
- Data receive if read/write bit = 0

1. Data Transmit

The data transmit state machine, illustrated in Fig. 6-4, starts data transmission two clocks after a response for the data write command is received; this occurs even if the command path detects a response error or response CRC error. If a response is not received from the card because of a response timeout, data is not transmitted. Depending upon the value of the transfer_mode bit in the Command register, the data transmit state machine puts data on the card data bus in a stream or in block(s).

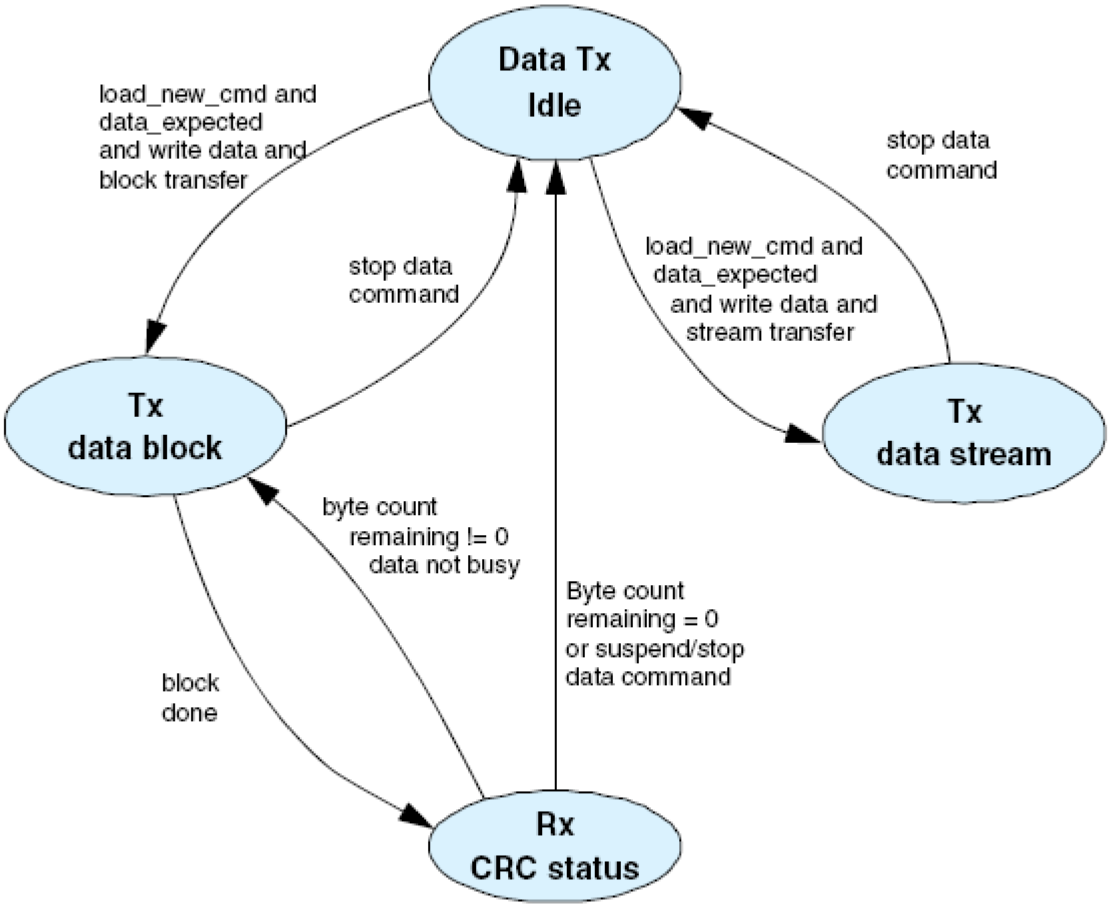

Data Tx Idle
load_new_cmd and data_expected and write data and block transfer
stop data command
stop data command
load_new_cmd and data_expected and write data and stream transfer
Tx data block
Tx data stream
byte count remaining != 0 data not busy
Byte count remaining = 0 or suspend/stop data command
block done
Rx CRC status

Fig. 7-4 SD/MMC Data Transmit State Machine

2. Stream Data transmit

If the transfer_mode bit in the Command register is set to 1, it is a stream-write data transfer. The data path pops the FIFO from the BIU and transmits in a stream to the card data bus. If the FIFO becomes empty, the card clock is stopped and restarted once data is available in the FIFO.

If the byte_count register is programmed to 0, it is an open-ended stream-write data transfer. During this data transfer, the data path continuously transmits data in a stream until the host software issues a stop command. A stream data transfer is terminated when the end bit of the stop command and end bit of the data match over two clocks.

If the byte_count register is programmed with a non-zero value and the send_auto_stop bit is set in the Command register, the stop command is internally generated and loaded in the command path when the end bit of the stop command occurs after the last byte of the stream write transfer matches. This data transfer can also terminate if the host issues a stop command before all the data bytes are transferred to the card bus.

3. Single Block Data

If the transfer_mode bit in the Command register is set to 0 and the byte_count register value is equal to the value of the block_size register, a single-block write-data transfer occurs. The data transmit state machine sends data in a single block, where the number of bytes equals the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set for a 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted for 1, 4, or 8 data lines, respectively.

After a single data block is transmitted, the data transmit state machine receives the CRC status from the card and signals a data transfer to the BIU; this happens when the data-transfer-over bit is set in the RINTSTS register.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register.

Additionally, if the start bit of the CRC status is not received by two clocks after the end of the data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the RINTSTS register.

4. Multiple Block Data

A multiple-block write-data transfer occurs if the transfer_mode bit in the Command register is set to 0 and the value in the byte_count register is not equal to the value of the block_size register. The data transmit state machine sends data in blocks, where the number of bytes in a block equals the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set to 1-bit, 4-bit, or 8-bit data transfer, the data is transmitted on 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and transmitted on 1, 4, or 8 data lines, respectively.

After one data block is transmitted, the data transmit state machine receives the CRC status from the card. If the remaining byte_count becomes 0, the data path signals to the BIU that the data transfer is done; this happens when the data-transfer-over bit is set in the RINTSTS register.

If the remaining data bytes are greater than 0, the data path state machine starts to transmit another data block.

If a negative CRC status is received from the card, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register, and continues further data transmission until all the bytes are transmitted.

Additionally, if the CRC status start bit is not received by two clocks after the end of a data block, a CRC status start bit error is signaled to the BIU by setting the write-no-CRC bit in the RINTSTS register; further data transfer is terminated.

If the send_auto_stop bit is set in the Command register, the stop command is internally generated during the transfer of the last data block, where no extra bytes are transferred to the card. The end bit of the stop command may not exactly match the end bit of the CRC status in the last data block.

If the block size is less than 4, 16, or 32 for card data widths of 1 bit, 4 bits, or 8 bits, respectively, the data transmit state machine terminates the data transfer when all the data is transferred, at which time the internally generated stop command is loaded in the command path.

If the byte_count is 0 – the block size must be greater than 0 – it is an open-ended block transfer. The data transmit state machine for this type of data transfer continues the block-write data transfer until the host software issues a stop or abort command.

5. Data Receive

The data-receive state machine, illustrated in Fig. 6-5, receives data two clock cycles after the end bit of a data read command, even if the command path detects a response error or response CRC error. If a response is not received from the card because a response timeout occurs, the BIU does not receive a signal that the data transfer is complete; this happens if the command sent by the SD/MMC controller is an illegal operation for the card, which keeps the card from starting a read data transfer.

If data is not received before the data timeout, the data path signals a data timeout to the BIU and an end to the data transfer done. Based on the value of the transfer_mode bit in the Command register, the data-receive state machine gets data from the card data bus in a stream or block(s).

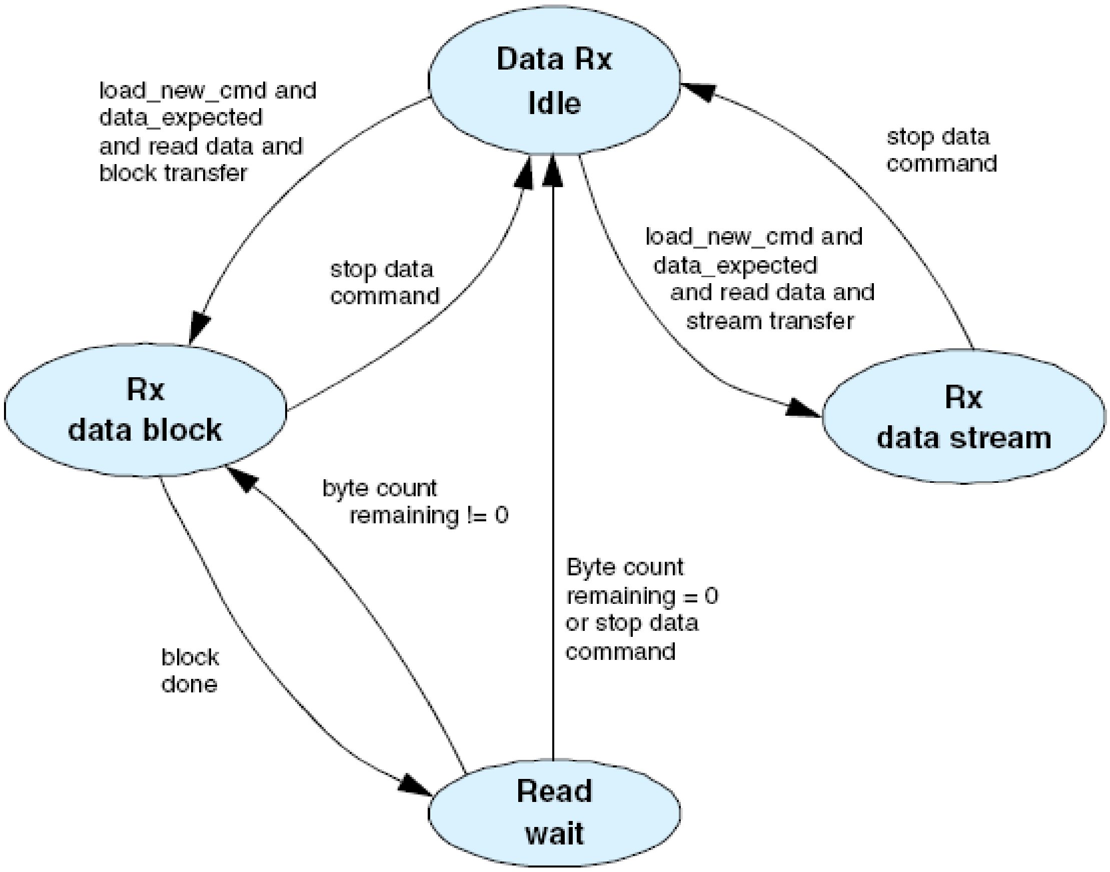


Fig. 7-5 SD/MMC Data Receive State Machine

6. Data Receive

A stream-read data transfer occurs if the transfer_mode bit in the Command register equals 1, at which time the data path receives data from the card and pushes it to the FIFO. If the FIFO becomes full, the card clock stops and restarts once the FIFO is no longer full.

An open-ended stream-read data transfer occurs if the byte_count register equals 0. During this type of data transfer, the data path continuously receives data in a stream until the host software issues a stop command. A stream data transfer terminates two clock cycles after the end bit of the stop command.

If the byte_count register contains a non-zero value and the send_auto_stop bit is set in the Command register, a stop command is internally generated and loaded into the command path, where the end bit of the stop command occurs after the last byte of the stream data transfer is received. This data transfer can terminate if the host issues a stop or abort command before all the data bytes are received from the card.

7. Single-Block Data Receive

A single-block read-data transfer occurs if the transfer_mode bit in the Command register is set to 0 and the value of the byte_count register is equal to the value of the block_size register. When a start bit is received before the data times out, data bytes equal to the block size and CRC16 are received and checked with the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively. If there is a CRC16 mismatch, the data path signals a data CRC error to the BIU. If the received end bit is not 1, the BIU receives an end-bit error.

8. Multiple-Block Data Receive

If the transfer_mode bit in the Command register is set to 0 and the value of the byte_count register is not equal to the value of the block_size register, it is a multiple-block read-data transfer. The data-receive state machine receives data in blocks, where the number of bytes in a block is equal to the block size, including the internally-generated CRC16.

If the CTYPE register bit for the selected card – indicated by the card_num value in the Command register – is set to a 1-bit, 4-bit, or 8-bit data transfer, data is received from 1, 4, or 8 data lines, respectively, and CRC16 is separately generated and checked for 1, 4, or 8 data lines, respectively. After a data block is received, if the remaining byte_count becomes 0, the data path signals a data transfer to the BIU.

If the remaining data bytes are greater than 0, the data path state machine causes another data block to be received. If CRC16 of a received data block does not match the internally-generated CRC16, a data CRC error to the BIU and data reception continue further data transmission until all bytes are transmitted. Additionally, if the end of a received data block is not 1, data on the data path signals terminate the bit error to the CIU and the data-receive state machine terminates data reception, waits for data timeout, and signals to the BIU that the data transfer is complete. If the send_auto_stop bit is set in the Command register, the stop command is internally generated when the last data block is transferred, where no extra bytes are transferred from the card; the end bit of the stop command may not exactly match the end bit of the last data block.

If the requested block size for data transfers to cards is less than 4, 16, or 32 bytes for 1-bit, 4-bit, or 8-bit data transfer modes, respectively, the data-transmit state machine terminates the data transfer when all data is transferred, at which point the internally-generated stop command is loaded in the command path. Data received from the card after that are then ignored by the data path.

If the byte_count is 0—the block size must be greater than 0—it is an open-ended block transfer. For this type of data transfer, the data-receive state machine continues the block-read data transfer until the host software issues a stop or abort command.

SDIO Interrupt Control

Interrupts for SD cards are reported to the BIU by asserting an interrupt signal for two clock cycles. SDIO cards signal an interrupt by asserting cdata_in low during the interrupt period; an interrupt period for the selected card is determined by the interrupt control state machine. An interrupt period is always valid for non-active or non-selected cards, and 1-bit data mode for the selected card. An interrupt period for a wide-bus active or selected card is valid for the following conditions:

- Card is idle
- Non-data transfer command in progress
- Third clock after end bit of data block between two data blocks
- From two clocks after end bit of last data until end bit of next data transfer command

Bear in mind that, in the following situations, the SD/MMC controller does not sample the SDIO interrupt of the selected card when the card data width is 4 bits. Since the SDIO interrupt is level-triggered, it is sampled in a further interrupt period and the host does not lose any SDIO interrupt from the card.

1. Read/Write Resume – The CIU treats the resume command as a normal data transfer command. SDIO interrupts during the resume command are handled similarly to other data commands. According to the SDIO specification, for the normal data command the interrupt period ends after the command end bit of the data command; for the resume command, it ends after the response end bit. In

the case of the resume command, the SD/MMC controller stops the interrupt sampling period after the resume command end bit, instead of stopping after the response end bit of the resume command.

2. Suspend during read transfer – If the read data transfer is suspended by the host, the host sets the abort_read_data bit in the SD/MMC controller to reset the data state machine. In the CIU, the SDIO interrupts are handled such that the interrupt sampling starts after the abort_read_data bit is set by the host. In this case the SD/MMC controller does not sample SDIO interrupts between the period from response of the suspend command to setting the abort_read_data bit, and starts sampling after setting the abort_read_data bit.

Clock Control

The clock control block provides different clock frequencies required for SD_MMC cards. The cclk_in is the source clock (cclk_in >= card max operating frequency)for clock dividers of the clock control block. This source clock (cclk_in) is used to generate different card clock frequencies. Each card clock can have different clock frequencies, since the SD card can be a low-speed SD card or a full-speed SD card. The SD/MMC controller provides one clock signal (cclk_out) per card, which allows each card to operate at different clock frequencies.

The clock frequency of a card depends on the following clock control registers:

- Clock Divider register – Internal clock dividers are used to generate different clock frequencies required for cards; a configuration parameter determines the number of clock dividers (1-4). The division factor for each clock divider can be programmed by writing to the Clock Divider register. The clock divider is an 8-bit value that provides a clock division factor from 1 to 510; a value of 0 represents a clock-divider bypass, a value of 1 represents a divide by 2, a value of 2 represents a divide by 4, and so on.
- Clock Source register – One of the divided clocks from four clock dividers is selected for a card cclk_out by programming the Clock Source register.
- Clock Control register – cclk_out can be enabled or disabled for each card under the following conditions:
  1. clk_enable – cclk_out for a card is enabled if the clk_enable bit for a card in the Clock Control register is programmed (set to 1) or disabled (set to 0).
  2. Low-power mode – Low-power mode of a card can be enabled by setting the low-power mode bit of the Clock Control register to 1. If low-power mode is enabled to save card power, the cclk_out signal is disabled when the card is idle for at least 8 card clock cycles. It is enabled when a new command is loaded and the command path goes to a non-idle state.

Additionally, cclk_out of a selected or active card is disabled when an internal FIFO is full – card read (no more data can be received from card) – or when the FIFO is empty – card write (no data is available for transmission). This helps to avoid FIFO overrun and underrun conditions.

Under the following conditions, the card clock is stopped or disabled, along with the active clk_en, for the selected card:

- Clock can be disabled by writing to Clock Enable register (clk_en bit = 1).
- If low-power mode is selected and card is idle, or not selected for 8 clocks.
- FIFO is full and data path cannot accept more data from the card and data transfer is incomplete –to avoid FIFO overrun.
- FIFO is empty and data path cannot transmit more data to the card and data transfer is incomplete – to avoid FIFO underrun.

Note: Care should be taken by the host firmware while changing the Clock Divider register and Clock Source register values. The card clock must be disabled through the Clock Control register before changing the values of the Clock Divider and Clock Source registers.

SD_MMC Mux/Demux Unit

A separate bus runs between the SD/MMC controller and each card. The demux logic sends the command or data to only the selected card when commands or data are sent from the controller. The unselected cards see those on their command or data path; a card is selected by the card_number value set in the Command register. Similarly, the response or data input from the selected card is multiplexed and sent to the SD/MMC controller.

7.3 Registers

This section describes the registers of the design.

7.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SDMMC_CTRL	0x00	W	0x0	SDMMC Control register
SDMMC_PWREN	0x04	W	0x0	Power-enable register
SDMMC_CLKDIV	0x08	W	0x0	Clock-divider register
SDMMC_CLKSRC	0x0C	W	0x0	Clock-source register
SDMMC_CLKENA	0x10	W	0x0	Clock-enable register
SDMMC_TMOUT	0x14	W	0xFFFFFF40	Time-out register(number of card clock output clocks)
SDMMC_CTYPE	0x18	W	0x0	Card-type register
SDMMC_BLKSIZ	0x1C	W	0x200	Block-size register
SDMMC_BYTCNT	0x20	W	0x200	Byte-count register
SDMMC_INTMASK	0x24	W	0x0	Interrupt-mask register
SDMMC_CMDARG	0x28	W	0x0	Command-argument register
SDMMC_CMD	0x2C	W	0x0	Command-register
SDMMC_RESP0	0x30	W	0x0	Response-0 register
SDMMC_RESP1	0x34	W	0x0	Response-1 register
SDMMC_RESP2	0x38	W	0x0	Response-2 register
SDMMC_RESP3	0x3C	W	0x0	Response-3 register
SDMMC_MINTSTS	0x40	W	0x0	Masked interrupt-status register
SDMMC_RINTSTS	0x44	W	0x0	Raw interrupt-status register
SDMMC_STATUS	0x48	W	0x6	Status register; mainly for debug purposes
SDMMC_FIFOTH	0x4C	W	0x001f0000	FIFO threshold register
SDMMC_CDETECT	0x50	W		Card-detect register
SDMMC_WRTPRT	0x54	W		Write-protect register
SDMMC_TCBCNT	0x5C	W	0x0	Transferred CIU card byte count
SDMMC_TBBCNT	0x60	W	0x0	Transferred host/DMA to/from BIU_FIFO byte count
SDMMC_DEBNCE	0x64	W	0x00ffffff	Card detect debounce register (number of host clocks)

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

7.3.2 Detail Register Description

SDMMC_CTRL
Address: Operational Base + offset (0x00)
SDMMC Control Register

bit	Attr	Reset Value	Description
31:25	-	-	Reserved.
24	RW	0x1	enable_OD_pullup: External open-drain pullup: 0- Disable 1- Enable

			When bit is set, command output always driven in open-drive mode.
23:20	RW	0x0	Card_voltage_b: Card regulator-B voltage setting; output to card_volt_b port.
19:16	RW	0x0	Card_voltage_a: Card regulator-A voltage setting; output to card_volt_a port.
15:9	-	-	Reserved
8	RW	0x0	abort_read_data: 0- No change 1- After suspend command is issued during read-transfer, software polls card to find when suspend happened. Once suspend occurs, software sets bit to reset data state-machine, which is waiting for next block of data. Bit automatically clears once data state-machine resets to idle. Used in SDIO card suspend sequence.
7	RW	0x0	send_irq_response: 0- No change 1- Send auto IRQ response Bit automatically clears once response is sent. To wait for MMC card interrupts, host issues CMD40, and SDMMC Controller waits for interrupt response from MMC card. In meantime, if host wants controller to exit waiting for interrupt state, it can set this bit, at which time controller command state-machine sends CMD40 response on bus and returns to idle state.
6	RW	0x0	read_wait: 0- Clear read wait 1- Assert read wait
5	RW	0x0	dma_enable: 0- Disable DMA transfer mode 1- Enable DMA transfer mode
4	RW	0x0	int_enable: Global interrupt enable/disable bit: 0- Disable interrupts 1- Enable interrupts The int port is 1 only when this bit is 1 and one or more unmasked interrupts are set.
3:2	-	-	Reserved
1	RW	0x0	fifo_reset: 0- No change 1- Reset data FIFO to reset FIFO pointers To reset FIFO, firmware should set bit to 1. This bit is auto-cleared after completion of reset operation.
0	RW	0x0	controller_reset: 0- No change 1- Reset SDMMC controller To reset controller, firmware should set bit to 1. This bit is auto-cleared after two AHB and two cclk_in clock cycles. This resets: ● BIU/CIU interface ● CIU and state machines ● abort_read_data, send_irq_response, and read_wait bits of control register.

			● Start_cmd bit of Command register Dose not affect any registers or DMA interface, or FIFO or host interrupts.

SDMMC_PWREN
Address: Operational Base + offset (0x04)
Power Enable Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved.
0	RW	0x0	Power_enable Once power is turned on, firmware should wait for regulator/switch ramp-up time before trying to initialize card. 0- power off 1- power on

SDMMC_CLKDIV
Address: Operational Base + offset (0x08)
Clock Divider Register

bit	Attr	Reset Value	Description
31:24	RW	0x0	clk_divider3: Clock divider-3 value. Clock division is 2*n. For example, value of 0 means divide by 2*0=0(no division, by pass), a value of 1 means divide by 2*1=2, a value of "ff" means divide by 2*255=510, and so on.
23:16	RW	0x0	clk_divider2: Clock divider-2 value. Clock division is 2*n. For example, value of 0 means divide by 2*0=0(no division, by pass), a value of 1 means divide by 2*1=2, a value of "ff" means divide by 2*255=510, and so on.
15:8	RW	0x0	clk_divider1: Clock divider-1 value. Clock division is 2*n. For example, value of 0 means divide by 2*0=0(no division, by pass), a value of 1 means divide by 2*1=2, a value of "ff" means divide by 2*255=510, and so on.
7:0	RW	0x0	clk_divider0: Clock divider-3 value. Clock division is 2*n. For example, value0 of 0 means divide by 2*0=0(no division, by pass), a value of 1 means divide by 2*1=2, a value of "ff" means divide by 2*255=510, and so on.

SDMMC_CLKSRC
Address: Operational Base + offset (0x0C)
SDMMC Clock source Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	clk_source: Clock divider source for SD/MMC card. Each card has two bits assigned to it. 00- Clock divider 0 01- Clock divider 1 10- Clock divider 2 11- Clock divider 3

SDMMC_CLKENA
Address: Operational Base + offset (0x10)
Clock Enable Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	cclk_low_power: Low-power control for SD/MMC card clock. 0- Non-low-power mode 1- Low-power mode; stop clock when card in IDLE(should be normally set to only MMC and SD memory cards; for SDIO cards, if interrupts must be detected, clock should not be stopped).
15:1	-	-	Reserved.
0	RW	0x0	cclk_enable: Clock_enable control for SD/MMC card clock. 0- Clock disabled 1- Clock enabled

SDMMC_TMOUT
Address: Operational Base + offset (0x14)
SDMMC Timeout Register

Bit	Attr	Reset Value	Description
31:8	W	0xffffff	data_timeout: Value for card Data Read Timeout; same value also used for Data Starvation by Host timeout. Value is in number of card output clocks-cclk out of selected card.
7:0	W	0x40	response_timeout: Response timeout value. Value is in number of card output clocks-cclk_out.

SDMMC_CTYPE
Address: Operational Base + offset (0x18)
Card Type Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	card_width: One bit indicates if card is 8-bit: 0- Non 8-bit mode 1- 8-bit mode
15:1	-	-	Reserved.
0	RW	0x0	card_width: One bit indicates if card is 1-bit or 4-bit: 0- 1-bit mode 1- 4-bit mode

The following examples use values for CTYPE[16]:

- If CTYPE[16]=1, the card at port0 is in 8-bit mode. Note that the CTYPE[0] value is ignored; it is recommended to keep this set to 0.
- If CTYPE[16]=0, the card at port0 is in either 1-bit or 4-bit mode, depending upon the value of CTYPE[0]; that is, if CTYPE[0]=1 -4-bit, CTYPE[0]=0 -1-bit.

SDMMC_BLKSIZ
Address: Operational Base + offset (0x1C)
Block Size Register

bit	Attr	Reset Value	Description
31:16	-	-	Reserved
15:0	RW	0x200	Block size

SDMMC_BYTCNT
Address: Operational Base + offset (0x20)
Byte Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x200	byte_count: Number of bytes to be transferred; should be integer multiple of Block Size for block transfers. For undefined number of byte transfers, byte count should be set to 0. When byte count is set to 0, it is responsibility of host to explicitly send stop/abort command to terminate data transfer.

SDMMC_INTMASK
Address: Operational Base + offset (0x24)
Interrupt Mask Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	sdio_int_mask: Mask SDIO interrupts When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enable an interrupt.
15:0	RW	0x0	int_mask: Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt. bit15 - End-bit error(read)/Write no CRC(EBE) bit14 - Auto command done(ACD) bit13 - Start-bit error(SBE) bit12 - Hardware locked write error(HLE) bit11 - FIFO underrun/overrun error(FRUN) bit10 - Data starvation-by-host timeout(HTO) bit9 - Data read timeout(DRTO) bit8 - Response timeout(RTO) bit7 - Data CRC error(DCRC) bit6 - Response CRC error(RCRC) bit5 - Receive FIFO data request(RXDR) bit4 - Transmit FIFO data request(TXDR) bit3 - Data transfer over(DTO) bit2 - Command done(CD) bit1 - Response error(RE) bit0 - Card detect(CD)

SDMMC_CMDARG
Address: Operational Base + offset (0x28)
Command Argument Register

bit	Attr	Reset Value	Description
31:0	RW	0x0	cmd_arg: Value indicates command argument to be passed to card.

SDMMC_CMD
Address: Operational Base + offset (0x2C)
Command Register

bit	Attr	Reset Value	Description
31	RW	0x0	start_cmd: Start command. Once command is taken by CIU, bit is cleared. When bit is set, host should not attempt to write to any command registers. If write is attempted, hardware lock error is set in raw interrupt register.
30:22	-	-	Reserved
21	RW	0x0	update_clock_registers_only: 0- Normal command sequence 1- Do not send commands, just update clock register value into card clock domain Following register values transferred into card clock domain: CLKDIV, CLKSRC, CLKENA. Changes card clocks(change frequency, trun off or on, and set low-frequency mode); provided in order to change clock frequency or stop clock without having to send command to cards. During normal command sequence, when update_clock_registers_only=0, following control registers are transferred from BIU to CIU: CMD, CMDARG, TMOUT, CTYPE, BLKSIZ, BYTCNT. CIU uses new register values for new command sequence to card. When bit is set, there are no Command Done interrupts because no command is sent to SD_MMC cards.
20:16	-	-	Reserved.
15	RW	0x0	send_initialization: 0- Do not send initialization sequence(80 clocks of 1) before sending this command 1- Send initialization sequence before sending this command. After power on, 80 clocks must be sent to card for initialization before sending any commands to card. Bit should be set while sending first command to card to that controller will initialize clocks before sending command to card.
14	RW	0x0	stop_abort_cmd: 0- Neither stop nor abort command to stop current data transfer in progress. If abort is sent to function-number currently selected or not in data-transfer mode, then bit should be set to 0. 1- Stop or abort command intended to stop current data transfer in progress. When open-ended or predefined data transfer is in progress, and host issues stop or abort command to stop data transfer, bit should be set so that command/data state-machines of CIU can return correctly to idle state.
13	RW	0x0	wait_prvdata_complete: 0- Send command at once, even if previous data transfer has not completed. 1- Wait for previous data transfer completion before sending command. The wait_prvdata_complete=0 option typically used to query status of card during data transfer or to stop current data transfer; card_number should be same as

			in previous command.
12	RW	0x0	send_auto_stop: 0- No stop command sent at end of data transfer 1- Send stop command at end of data transfer When set, SDMMC controller sends stop command to SDMMC cards at end of data transfer. Refer to Table9 on page101 to determine: 0- when send_auto_stop bit should be set, since som data transfers do not need explicit stop commands 1- open-ended transfers that software should explicity send to stop command Don't care if no data expected from card.
11	RW	0x0	transfer_mode: 2- Block data transfer command 3- Stream data transfer command Don't care if no data expected.
10	RW	0x0	read/write: 0- Read from card 1- Write to card Don't care if no data expected from card.
9	RW	0x0	data_expected: 0- No data transfer expected(read/write) 1- Data transfer expected(read/write)
8	RW	0x0	check_response_crc: 0- Do not check response CRC 1- Check response CRC Some of command responses do not return valid CRC bits. Software should disable CRC checks for those commands in order to disable CRC checking by controller.
7	RW	0x0	response_length: 0- Short response expected from card 1- Long response expected from card
6	RW	0x0	response_expect: 0- No response expected from card 1- Response expected from card
5:0	RW	0x0	cmd_index: Command index

SDMMC_RESP0
Address: Operational Base + offset (0x30)
Response Register 0

bit	Attr	Reset Value	Description
31:0	R	0x0	response 0: Bit[31:0] of response

SDMMC_RESP1
Address: Operational Base + offset (0x34)
Response Register 1

bit	Attr	Reset Value	Description
31:0	R	0x0	response 1: Register represents bit[63:32] of long response. When CIU sends auto-stop command, then response is saved in register. Response for previous command sent by host is still preserved in Response 0 register. Additional auto-stop issued only for data transfer commands, and response type is always "short" for

			them. For information on when CIU sends auto-stop commands, refer to “Auto-Stop” on page 100.

SDMMC_RESP2
Address: Operational Base + offset (0x38)
Response Register 2

bit	Attr	Reset Value	Description
31:0	R	0x0	response 2: Bit[95:64] of long response

SDMMC_RESP3
Address: Operational Base + offset(0x3C)
Response Register 3

bit	Attr	Reset Value	Description
31:0	R	0x0	response 3: Bit[127:96] of long response

SDMMC_MINTSTS
Address: Operational Base + offset(0x40)
Masked Interrupt Status Register
MINTSTS = RINTSTS and INTMASK

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	R	0x0	sdio_interrupt: Interrupt from SDIO card. SDIO interrupt for card enabled only if corresponding sdio_int_mask bit is set in Interrupt mask register(mask bit 1 enable interrupt;0 masks interrupt). 0- No SDIO interrupt from card 1- SDIO interrupt from card
15:0	R	0x0	int_status: Interrupt enabled only if corresponding bit in interrupt mask register is set. bit15- End-bit error read/write no CRC (EBE) bit14- Auto command done(ACD) bit13- Start-bit error(SBE) bit12- Hardware locked write error(HLE) bit11- FIFO underrun/overrun error(FRUN) bit10- Data starvation by host timeout(HTO) bit9 – Data read timeout(DRTO) bit8 – Response timeout(RTO) bit7 – Data CRC error(DCRC) bit6 – Response CRC error(RCRC) bit5 – Receive FIFO data request(RXDR) bit4 – Transmit FIFO data request(TXDR) bit3 – Data transfer over(DTO) bit2 – Command done(CD) bit1 – Response error(RE) bit0 – Card detect(CD)

SDMMC_RINTSTS
Address: Operational Base + offset(0x44)
Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:17	-	-	Reserved.
16	RW	0x0	sdio_interrupt: Interrupt from SDIO card. Writes to this bit clear it.

			Value of 1 clears bit and 0 leaves bit intact. 0- No SDIO interrupt from card 1- SDIO interrupt from card Bit is logged regardless of interrupt-mask status.
15:0	RW	0x0	int_status: Writes to bits clear status bit. Value of 1 clears status bit, and value of 0 leaves bit intact. Bits are logged regardless of interrupt mask status. bit15- End-bit error read/write no CRC (EBE) bit14- Auto command done(ACD) bit13- Start-bit error(SBE) bit12- Hardware locked write error(HLE) bit11- FIFO underrun/overrun error(FRUN) bit10- Data starvation by host timeout(HTO) bit9 – Data read timeout(DRTO) bit8 – Response timeout(RTO) bit7 – Data CRC error(DCRC) bit6 – Response CRC error(RCRC) bit5 – Receive FIFO data request(RXDR) bit4 – Transmit FIFO data request(TXDR) bit3 – Data transfer over(DTO) bit2 – Command done(CD) bit1 – Response error(RE) bit0 – Card detect(CD)

SDMMC_STATUS
Address: Operational Base + offset(0x48)
Status Register

bit	Attr	Reset Value	Description
31	R	0x0	dma_req: DMA request signal state
30	R	0x0	dma_ack: DMA acknowledge signal state
29:17	R	0x0	fifo_count: FIFO count – Number of filled locations in FIFO
16:11	R	0x0	response_index: Index of previous response, including any auto-stop sent by core
10	R	0x0	data_state_mc_busy: Data transmit or receive state-machine is busy
9	R	0x0	data_busy: Inverted version of raw selected card_data[0] 0- card data not busy 1- card data busy
8	R	0x0	data_3_status: Raw selected card_data[3]; checks whether card is present 0- card not present 1- card present
7:4	R	0x0	command fsm states: 0- Idle 1- Send init sequence 2- Tx cmd start bit 3- Tx cmd tx bit 4- Tx cmd index + arg 5- Tx cmd crc7 6- Tx cmd end bit

			7- Rx resp start bit 8- Rx resp IRQ response 9- Rx resp tx bit 10- Rx resp cmd idx 11- Rx resp data 12- Rx resp crc7 13- Rx resp end bit 14- Cmd path wait NCC 15- Wait; CMD-to response turnaround
3	R	0x0	fifo_full: FIFO is full status
2	R	0x1	fifo_empty: FIFO is empty status
1	R	0x1	fifo_tx_watermark: FIFO reached Transmit watermark level; not qualified with data transfer.
0	R	0x0	fifo_rx_watermark: FIFO reached Receive watermark level; not qualified with data transfer.

SDMMC_FIFOTH
Address: Operational Base + offset(0x4C)
FIFO Threshold Watermark Register

bit	Attr	Reset Value	Description
31	-	-	Reserved
30:28	RW	0x0	DW_DMA_Multiple_Transaction_Size: Burst size of multiple transaction; should be programmed same as DW_DMA controller multiple-transaction-size SRC/DEST_MSIZE. 000-1 transfers 001-4 010-8 011-16 100-32 101-64 110-128 111-256 Value should be sub-multiple of (RX_WMark+1) and (32-TX_WMark) Recommended: MSize=16, TX_WMask=16, RX_WMask=15
27:16	RW	0x1f	RX_WMark: FIFO threshold watermark level when receiving data to card. When FIFO data count reaches greater than this number, DMA/FIFO request is raised. During end of packet, request is generated regardless of threshold programming in order to complete any remaining data. In non-DMA mode, when receiver FIFO threshold(RXDR) interrupt is endable, then interrupt is generated instead of DMA request. During end of packet, interrupt is not generated if threshold programming is larger than any remaining data. It is responsibility of host to read remaining bytes on seeing Data Transfer Done interrupt. In DMA mode, at end of packet, even if remaining bytes are less than threshold, DMA request does single

			transfers to flush out any remaining bytes before Data Transfer Done interrupt is set. Limitation: RX_WMark<=30 Recommended:15;(means greater than 15)
15:12	-	-	Reserved
11:0	RW	0x0	TX_WMark: FIFO threshold watermark level when transmitting data to card. When FIFO data count is less than or equal to this number, DMA/FIFO request is raised. If interrupt is enabled, then interrupt occurs. During end of packet, request or interrupt is generated. Regardless of threshold programming. In non-DMA mode, when transmit FIFO threshold(TXDR) interrupt is enable, then interrupt is generated instead of DMA request. During end of packet, on last interrupt, host is responsible for filling FIFO with only required remaining bytes(not before FIFO is full or after CIU completes data transfers, because FIFO may not be empty). In DMA mode, at end of packet, if last transfer is less than burst size, DMA controller does single cycles until required bytes are transferred. Limitation: TX_WMark>=1; Recommended:16(means less than or equal to 16)

SDMMC_CDETECT
Address: Operational Base + offset(0x50)
Card Detect Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x1	card_detect_n: Value on card_detect input port 0 represents presence of card.

SDMMC_WRTPRT
Address: Operational Base + offset(0x54)
Write Protect Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	write_protect: Value on card_write_prt input port 1 represents write protection.

SDMMC_TCBCNT
Address: Operational Base + offset(0x5C)
Transferred CIU Card Byte Count Register

bit	Attr	Reset Value	Description
31:0	R	0x0	trans_card_byte_count Number of bytes transferred by CIU unit to card.

SDMMC_TBBCNT
Address: Operational Base + offset(0x60)
Transferred Host to BIU-FIFO Byte Count Register

bit	Attr	Reset Value	Description
31:0	R	0x0	trans_fifo_byte_count: Number of bytes transferred between Host/DMA

			memory and BIU FIFO.

SDMMC_DEBNCE
Address: Operational Base + offset(0x64)
Debounce Count Register

bit	Attr	Reset Value	Description
31:24	-	-	Reserved
23:0	RW	24'hff_ffff	Number of host clocks used by debounce filter logic; typical debounce time is 5-25ms.

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

7.4 Functional Description

7.4.1 Operation

Software/Hardware Restrictions

Before issuing a new data transfer command, the software should ensure that the card is not busy due to any previous data transfer command. Before changing the card clock frequency, the software must ensure that there are no data or command transfers in progress.

To avoid glitches in the card clock outputs (cclk_out), the software should use the following steps when changing the card clock frequency:

1. Before disable the clocks, ensure that the card is not busy due to any previous data command. To determine this, check for 0 in bit9 of STATUS register.
2. Update the Clock Enable register to disable all clocks. To ensure completion of any previous command before this update, send a command to the CIU to update the clock registers by setting:
   - start_cmd bit
   - "update clock registers only" bits
   - "wait_previous data complete" bit

   Wait for the CIU to take the command by polling for 0 on the start_cmd bit.
3. Set the start_cmd bit to update the Clock Divider and/or Clock Source registers, and send a command to the CIU in order to update the clock registers; wait for the CIU to take the command.
4. Set start_cmd to update the Clock Enable register in order to enable the required clocks and send a command to the CIU to update the clock registers; wait for the CIU to take the command.

In non-DMA mode, while reading from a card, the Data Transfer Over (RINTSTS[3]) interrupt occurs as soon as the data transfer from the card is over. There still could be some data left in the FIFO, and the RX_WMark interrupt may or may not occur, depending on the remaining bytes in the FIFO. Software should read any remaining bytes upon seeing the Data Transfer Over (DTO) interrupt. While using the external DMA interface for reading from a card, the DTO interrupt occurs only after all the data is flushed to memory by the DMA interface unit.

While writing to a card in external DMA mode, if an undefined-length transfer is selected by setting the Byte Count Register to 0, the DMA logic will likely request more data than it will send to the card, since it has no way of knowing at which point the software will stop the transfer. The DMA request stops as soon as the DTO is set by the CIU.

If the software issues a controller_reset command by setting control register bit[0] to 1, all the CIU state machines are reset; the FIFO is not cleared. The DMA sends all remaining bytes to the host. In addition to a card-reset, if a FIFO reset is also issued, then:

- Any pending DMA transfer on the bus completes correctly
- DMA data read is ignored
- Write data is unknown(x)

Additionally, if dma_reset is also issued, any pending DMA transfer is abruptly terminated. When the DW-DMA is used, the DMA controller channel should also be reset and reprogrammed.

If any of the previous data commands do not properly terminate, then the software should issue the FIFO reset in order to remove any residual data, if any, in the FIFO. After asserting the FIFO reset, you should wait until this bit is cleared.

One data-transfer requirement between the FIFO and host is that the number of transfers should be a multiple of the FIFO data width (32bits). For example, you want to write only 15 bytes to an SDMMC card (BYTCNT), the host should write 16 bytes to the FIFO or program the DMA to do 16-byte transfers. The software can still program the Byte Count register to only 15, at which point only 15 bytes will be transferred to the card. Similarly, when 15 bytes are read from a card, the host should still read all 16 bytes from the FIFO.

It is recommended that you not change the FIFO threshold register in the middle of data transfers.

7.4.2 Programming sequence

Initialization

Fig. 6-6 illustrates the initialization flow.

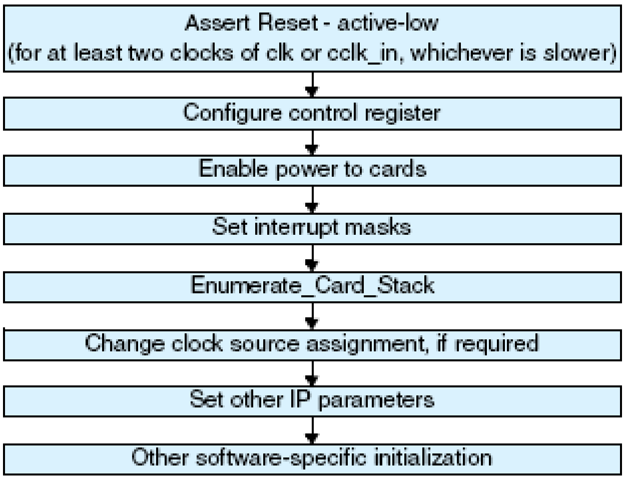

Assert Reset - active-low
(for at least two clocks of clk or cclk_in, whichever is slower)
Configure control register
Enable power to cards
Set interrupt masks
Enumerate_Card_Stack
Change clock source assignment, if required
Set other IP parameters
Other software-specific initialization

Fig. 7-6 SD/MMC Initialization Sequence

Once the power and clocks are stable, reset_n should be asserted(active-low) for at least two clocks of clk or cclk_in, whichever is slower. The reset initializes the registers, ports, FIFO-pointers, DMA interface controls, and state-machines in the design. After power-on reset, the software should do the following:

1. Configure control register – For MMC mode, enable the open-drain pullup by setting enable_OD_pullup(bit24) in the control register.
2. Enable power to cards – Before enabling the power, confirm that the voltage setting to the voltage regulators is correct. Enable power to the connected cards by setting the corresponding bit to 1 in the Power Enable register. Wait for the power ramp-up time.
3. Set masks for interrupts by clearing appropriate bits in the Interrupt Mask register. Set the global int_enable bit of the Control register. It is recommended that you write 0xffff_ffff to the Raw Interrupt register in order to clear any pending interrupts before setting the int_enable bit.
4. Enumerate card stack – Each card is enumerated according to card type; for details, refer to "Enumerated Card Stack". For enumeration, you should restrict the clock frequency to 400KHz.
5. Changing clock source assignment – set the card frequency using the clock-divider and clock-source registers; for details, refer to "Clock Programming". MMC cards operate at a maximum of 20MHz (at maximum of 52MHz in high-speed mode). SD mode operates at a maximum of 25MHz (at maximum of 50MHz in high-speed mode).
6. Set other parameters, which normally do not need to be changed with every command, with a typical value such as timeout values in cclk_out according to SDMMC

specifications.

- ResponseTimeOut = 0x64
- DataTimeOut = highest of one of the following:
  (10*((TAAC*Fop)+(100*NSAC))
  Host FIFO read/write latency from FIFO empty/full
- Set the debounce value to 25ms(default:0x0fffff) in host clock cycle units in the DEBNCE register.
- FIFO threshold value in bytes in the FIFOTH register. Typically, the threshold value can be set to half the FIFO depth; that is:
  RX_WMark=15;
  TX_WMark=16

7. According to MMC standards, the open-drain pullup resistor is required during only the enumeration phase. Therefore for MMC mode, disable the open-drain pullup by clearing the enable_OD_pullup (bit24) in the Control register.

Enumerated Card Stack

The card stack does the following:

- Enumerates all connected cards
- Sets the RCA for the connected cards
- Reads card-specific information
- Stores card-specific information locally

Enumeration depends on the operating mode of the SDMMC Host Controller; the card type is first identified and the appropriate card enumeration routine is called.

1. Check if the card is connected.
2. Clear the card type register to set the card width as a single bit. For the given card number, clear the corresponding bits in the card_type register. Clear the register bit for a 1-bit, 4-bit, or 8-bit bus width. For example, for card number=1, clear bit 0 and bit 16 of the card_type register.
3. Set clock frequency to Fod=400KHz, maximum – Program clock divider0 (bits 0-7 in the CLKDIV register) value to one-half of the cclk_in frequency divided by 400KHz. For example, if cclk_in is 20MHz, then the value is 20,000/(2*400)=25.
4. Identify the card type; that is, SD, MMC, or SDIO.
   a. Send CMD5 first. If a response is received, then the card is SDIO
   b. If not, send CMD8 with the following Argument
      Bit[31:12] = 20'h0 //reserved bits
      Bit[11:8] = 4'b0001 //VHS value
      Bit[7:0] = 8'b10101010 //Preferred Check Pattern by SD2.0
   c. If Response is received the card supports High Capacity SD2.0 then send ACMD41 with the following Argument
      Bit[31] = 1'b0; //Reserved bits
      Bit[30] = 1'b1; //High Capacity Status
      Bit[29:24] = 6'h0; //Reserved bits
      Bit[23:0] = Supported Voltage Range
   d. If Response is received for ACMD41 then the card is SD. Otherwise the card is MMC.
   e. If response is not received for initial CMD8 then card does not support High Capacity SD2.0, then issue CMD0 followed by ACMD41 with the following Argument
      Bit[31] = 1'b0; //Reserved bits
      Bit[30] = 1'b0; //High Capacity Status
      Bit[29:24] = 6'h0; //Reserved bits
      Bit[23:0] = Supported Voltage Range

5. Enumerate the card according to the card type.
6. Use a clock source with a frequency = Fod (that is, 400KHz) and use the following enumeration command sequence:
   - SD card – Send CMD0, CMD8, ACMD41, CMD2, CMD3.
   - SDIO – Send CMD5, CMD3.
   - MMC – Send CMD0, CMD1, CMD2, CMD3.

Power Control

You can implement power control using the following registers, along with external circuitry:

- Control register bits card_voltage_a and card_voltage_b – Status of these bits is reflected at the IO pins. The bits can be used to generate or control the supply voltage that the memory cards require.
- Power enable register – Control power to individual cards.

Programming these two register depends on the implemented external circuitry. While turning on or off the power enable, you should confirm that power supply settings are correct. Power to all cards usually should be disable while switching off the power.

Clock Programming

The SDMMC controller supports four clock sources, each of which can be programmed with a different frequency; software can select the clock source for each card. The clock to an individual card can be enabled or disabled. Registers that support this are:

- CLKDIV – Programs individual clock source frequency.
- CLKSRC – Assign clock source for each card.
- CLKENA – Enables or disables clock for individual card and enables low-power mode, which automatically stops the clock to a card when the card is idle for more than 8 clocks.

The SDMMC Controller loads each of these registers only when the start_cmd bit and the Update_clk_regs_only bit in the CMD register are set. When a command is successfully loaded, the SDMMC Controller clears this bit, unless the SDMMC Controller already has another command in the queue, at which point it gives an HLE(Hardware Locked Error).

Software should look for the start_cmd and the Update_clk_regs_only bits, and should also set the wait_prvdata_complete bit to ensure that clock parameters do not change during data transfer. Note that even though start_cmd is set for updating clock registers, the SDMMC Controller does not raise a command_done signal upon command completion.

The following shows how to program these registers:

1. Confirm that no card is engaged in any transaction; if there is a transaction, wait until it finishes.
2. Stop all clocks by writing xxxx0000 to the CLKENA register. Set the start_cmd, Update_clk_regs_only, and wait_prvdata_complete bits in the CMD register. Wait until start_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.
3. Program the CLKDIV and CLKSRC registers, as required. Set the start_cmd, Update_clk_regs_only, and wait_prvdata_complete bits in the CMD register. Wait until start_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.
4. Re-enable all clocks by programming the CLKENA register. Set the start_cmd, Update_clk_regs_only, and wait_prvdata_complete bits in the CMD register. Wait until start_cmd is cleared or an HLE is set; in case of an HLE, repeat the command.

No-Data Command With or Without Response Sequence

To send any non-data command, the software needs to program the CMD register @0x2C and the CMDARG register @0x28 with appropriate parameters. Using these two registers, the SD/MMC controller forms the command and sends it to the command bus. The SD/MMC controller reflects the errors in the command response through the error bits of the RINTSTS register.

When a response is received – either erroneous or valid – the SD/MMC controller sets the command_done bit in the RINTSTS register. A short response is copied in Response Register0, while a long response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3 register bit 31 represents the MSB, and the Response0 register bit 0 represents the LSB of a long response.

For basic commands or non-data commands, follow these steps:

1. Program the Command register @0x28 with the appropriate command argument parameter.
2. Program the Command register @0x2C with the settings in Table 6-2.

Table 7-2 SD/MMC Command Register Settings for No-Data Command

Parameter	Value	Description
Default		
start_cmd	1	-
Update_clk_regs_only	0	No clock parameters update command
data_expected	0	No data command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
cmd_index	command-index	-
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
wait_prvdata_complete	1	Before sending command on command line, host should wait for completion of any data command in process, if any (recommended to always set this bit, unless the current command is to query status or stop data transfer when transfer is in progress)
check_response_crc	1	If host should crosscheck CRC of response received

3. Wait for command acceptance by host. The following happens when the command is loaded into the SD/MMC controller:

- SD/MMC controller accepts the command for execution and clears the start_cmd bit in the CMD register, unless one command is in process, at which point the SD/MMC controller can load and keep the second command in the buffer.
- If the SD/MMC controller is unable to load the command – that is, a command is already in progress, a second command is in the buffer, and a third command is attempted – then it generates an HLE (hardware-locked error).

4. Check if there is an HLE.

5. Wait for command execution to complete. After receiving either a response from a card or response timeout, the SD/MMC controller sets the command_done bit in the RINTSTS register. Software can either poll for this bit or respond to a generated interrupt.

6. Check if response_timeout error, response_CRC error, or response error is set. This can be done either by responding to an interrupt raised by these errors or by polling bits 1, 6, and 8 from the RINTSTS register @0x44. If no response error is received, then the response is valid. If required, the software can copy the response from the response registers @0x30-0x3C.

Software should not modify clock parameters while a command is being executed.

Data Transfer Commands

Data transfer commands transfer data between the memory card and the SD/MMC controller. To send a data command, the SD/MMC controller needs a command argument, total data size, and block size. Software can receive or send data through the FIFO. Before a data transfer command, software should confirm that the card is not busy and is in a transfer state, which can be done using the CMD13 and CMD7 commands, respectively.

For the data transfer commands, it is important that the same bus width that is programmed in the card should be set in the card type register @0x18.

The SD/MMC controller generates an interrupt for different conditions during data transfer, which are reflected in the RINTSTS register @0x44 as:

1. Data_Transfer_Over (bit 3) – When data transfer is over or terminated. If there is a response timeout error, then the SD/MMC Controller does not attempt any data transfer and the "Data Transfer Over" bit is never set.
2. Transmit_FIFO_Data_request (bit 4) – FIFO threshold for transmitting data was reached; software is expected to write data, if available, in FIFO.
3. Receive_FIFO_Data_request (bit 5) – FIFO threshold for receiving data was reached; software is expected to read data from FIFO.
4. Data starvation by Host timeout (bit 10) – FIFO is empty during transmission or is full during reception. Unless software writes data for empty condition or reads data for full condition, the SD/MMC controller cannot continue with data transfer. The clock to the card has been stopped.
5. Data read timeout error (bit 9) – Card has not sent data within the timeout period.
6. Data CRC error (bit 7) – CRC error occurred during data reception.
7. Start bit error (bit 13) – Start bit was not received during data reception.
8. End bit error (bit 15) – End bit was not received during data reception or for a write operation; a CRC error is indicated by the card.

Conditions 6, 7, and 8 indicate that the received data may have errors. If there was a response timeout, then no data transfer occurred.

Single-Block or Multiple-Block Read

Steps involved in a single-block or multiple-block read are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C. The SD/MMC controller expects data from the card in blocks of size BLKSIZ each.
3. Program the CMDARG register @0x28 with the data address of the beginning of a data read.

   Program the Command register with the parameters listed in Table 6-3. For SD and MMC cards, use CMD17 for a single-block read and CMD18 for a multiple-block read. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

Table 7-3 SD/MMC Command Register Setting for Single-Block or Multiple-Block Read

Parameter	Value	Description
Default		
start_cmd	1	-
Update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0 or 1	Set according to Table xx
transfer_mode	0	Block transfer
read_write	0	Read from card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
cmd_index	command -index	-
wait_prvdata_complete	1	0- Sends command immediately 1- Sends command after previous data transfer ends
check_response_crc	1	0- SD/MMC controller should not check response CRC 1- SD/MMC controller should check response

		CRC

After writing to the CMD register, the SD/MMC controller starts executing the command; when the command is sent to the bus, the command_done interrupt is generated.

4. Software should look for data error interrupts; that is, bits 7, 9, 13, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending a STOP command.
5. Software should look for Receive_FIFO_Data_request and/or data starvation by host timeout conditions. In both cases, the software should read data from the FIFO and make space in the FIFO for receiving more data.
6. When a Data_Transfer_Over interrupt is received, the software should read the remaining data from the FIFO.

Single-Block or Multiple-Block Write

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the BYTCNT register @0x20.
2. Write the block size in bytes in the BLKSIZ register @0x1C; the SD/MMC controller sends data in blocks of size BLKSIZ each.
3. Program CMDARG register @0x28 with the data address to which data should be written.
4. Write data in the FIFO; it is usually best to start filling data the full depth of the FIFO.
5. Program the Command register with the parameters listed in Table 6-4. For SD and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SDIO cards, use CMD53 for both single-block and multiple-block transfers.

Table 7-4 SD/MMC Command Register Settings for Single-Block or Multiple-Block Write

Parameter	Value	Description
Default		
start_cmd	1	-
Update_clk_regs_only	0	No clock parameters update command
card number	0	Actual card number(one controller only connect one card, the num is No.0)
send_initialization	0	Can be 1, but only for card reset commands, such as CMD0
stop_abort_cmd	0	Can be 1 for commands to stop data transfer, such as CMD12
send_auto_stop	0 or 1	Set according to Table xx
transfer_mode	0	Block transfer
read_write	1	Write to card
data_expected	1	Data command
response_length	0	Can be 1 for R2(long) response
response_expect	1	Can be 0 for commands with no response; for example, CMD0, CMD4, CMD15, and so on
User-selectable		
cmd_index	command-index	-
wait_prvdata_complete	1	2- Sends command immediately 3- Sends command after previous data transfer ends
check_response_crc	1	2- SD/MMC controller should not check response CRC 3- SD/MMC controller should check response CRC

After writing to the CMD register, SD/MMC controller starts executing a command; when the command is sent to the bus, a command_done interrupt is generated.

6. Software should look for data error interrupts; that is, for bits 7, 9, and 15 of the RINTSTS register. If required, software can terminate the data transfer by sending the STOP command.
7. Software should look for Transmit_FIFO_Data_request and/or timeout conditions from data starvation by the host. In both cases, the software should write data into the FIFO.
8. When a Data_Transfer_Over interrupt is received, the data command is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the SD/MMC controller should send the STOP command, if necessary. Completion of the AUTO-STOP command is reflected by the Auto_command_done interrupt – bit 14 of the RINTSTS register. A response to AUTO_STOP is stored in RESP1 @0x34.

Stream Read

A stream read is like the block read mentioned in "Single-Block or Multiple-Block Read", except for the following bits in the Command register:

transfer_mode = 1; //Stream transfer
cmd_index = CMD20;

A stream transfer is allowed for only a single-bit bus width.

Stream Write

A stream write is exactly like the block write mentioned in "Single-Block or Multiple-Block Write", except for the following bits in the Command register:

transfer_mode = 1;//Stream transfer
cmd_index = CMD11;

In a stream transfer, if the byte count is 0, then the software must send the STOP command. If the byte count is not 0, then when a given number of bytes completes a transfer, the SD/MMC controller sends the STOP command. Completion of this AUTO_STOP command is reflected by the Auto_command_done interrupt. A response to an AUTO_STOP is stored in the RESP1 register @0x34.

A stream transfer is allowed for only a single-bit bus width.

Sending Stop or Abort in Middle of Transfer

The STOP command can terminate a data transfer between a memory card and the SD/MMC controller, while the ABORT command can terminate an I/O data transfer for only the SDIO_IOONLY and SDIO_COMBO cards.

- Send STOP command – Can be sent on the command line while a data transfer is in progress; this command can be sent at any time during a data transfer. For information on sending this command, refer to "No-Data Command With or Without Response Sequence".

  You can also use an additional setting for this command in order to set the Command register bits (5-0) to CMD12 and set bit 14 (stop_abort_cmd) to 1. If stop_abort_cmd is not set to 1, the SD/MMC controller does not know that the user stopped a data transfer. Reset bit 13 of the Command register (wait_prvdata_complete) to 0 in order to make the SD/MMC controller send the command at once, even though there is a data transfer in progress.
- Send ABORT command – Can be used with only an SDIO_IOONLY or SDIO_COMBO card. To abort the function that is transferring data, program the function number in AS*x* bits (CCCR register of card, address 0x06, bits (0-2) using CMD52.

  This is a non-data command. For information on sending this command, refer to "No-Data Command With or Without Response Sequence".

The command format for CMD52 is illustrated in Fig. 6-7:

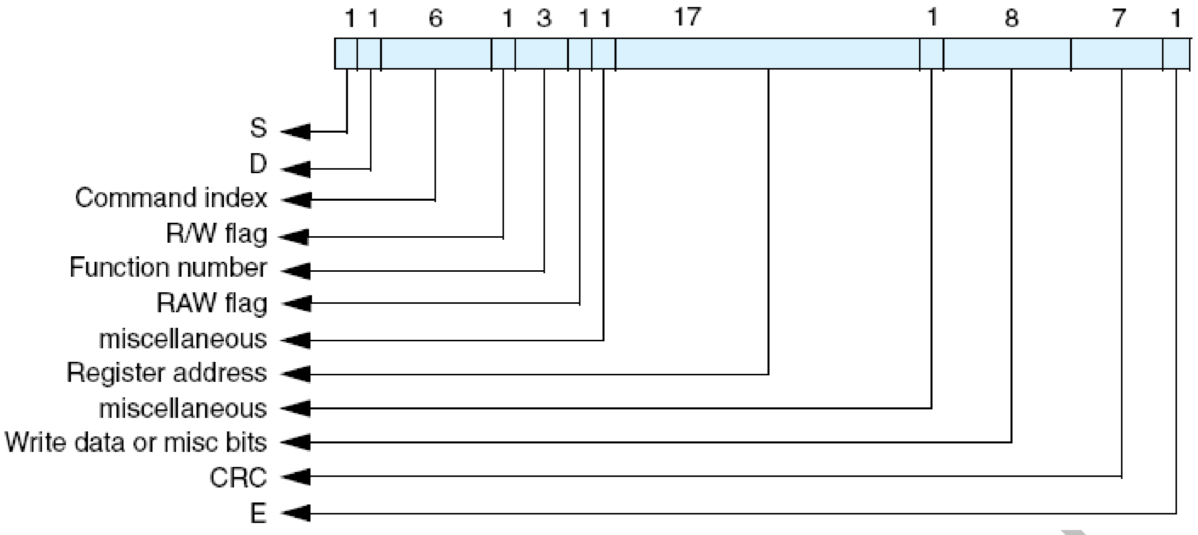


Fig. 7-7 SD/MMC Command format for CMD52

a. Program the CMDARG register @0x28 with the appropriate command argument parameters listed in Table 6-5.

Table 7-5 SD/MMC Parameters for CMDARG Registers

CMDARG Bits	Contents	Value
31	R/W flag	1
30-28	Function Number	0, for CCCR access
27	RAW flag	1, if needed to read after write
26	Don’t care	-
25-9	Register address	0x06
8	Don’t care	-
7-0	Write Data	Function number to be aborted

b. Program the Command register using the command index as CMD52. Similar to the STOP command described, set bit 14 of the Command register (stop_abort_cmd) to 1, which must be done in order to inform the SD/MMC controller that the user aborted the data transfer. Reset bit 13 (wait_prvdata_complete) of the Command register to 0 in order to make the SD/MMC controller send the command at once, even though a data transfer is in progress.
c. Wait for command_transfer_over.
d. Check response (R5) for errors.

Suspend or Resume Sequence

In an SDIO card, the data transfer between an I/O function and the SD/MMC controller can be temporarily halted using the SUSPEND command; this may be required in order to perform a high-priority data transfer with another function. When desired, the data transfer can be resumed using the RESUME command.

The following functions can be implemented by programming the appropriate bits in the CCCR register (Function 0) of the SDIO card. To read from or write to the CCCR register, use the CMD52 command.

1. SUSPEND data transfer – Non-data command.
   a. Check if the SDIO card supports the SUSPEND/RESUME protocol; this can be done through the SBS bit in the CCCR register @0x08 of the card.
   b. Check if the data transfer for the required function number is in process; the function number that is currently active is reflected in bits 0-3 of the CCCR register @0x0D. Note that if the BS bit (address 0xc::bit 0) is 1, then only the function number given by the FS*x* bits is valid.
   c. To suspend the transfer, set BR (bit 2) of the CCCR register @0x0C.
   d. Poll for clear status of bits BR (bit 1) and BS (bit 0) of the CCCR @0x0C. The BS

(Bus Status) bit is 1 when the currently-selected function is using the data bus; the BR (Bus Release) bit remains 1 until the bus release is complete. When the BR and BS bits are 0, the data transfer from the selected function has been suspended.

   e. During a read-data transfer, the SD/MMC controller can be waiting for the data from the card. If the data transfer is a read from a card, then the SD/MMC controller must be informed after the successful completion of the SUSPEND command. The SD/MMC controller then resets the data state machine and comes out of the wait state. To accomplish this, set abort_read_data (bit 8) in the Control register.
   f. Wait for data completion. Get pending bytes to transfer by reading the TCBCNT register @0x5C.

2. RESUME data transfer – This is a data command.
   a. Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.
   b. If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.
   c. Check that a function to be resumed is ready for data transfer; this can be confirmed by reading the RFx flag in CCCR @0x0F. If RF = 1, then the function is ready for data transfer.
   d. To resume transfer, use CMD52 to write the function number at FS*x* bits (0-3) in the CCCR register @0x0D. Form the command argument for CMD52 and write it in CMDARG @0x28; bit values are listed in Table 6-6.

Table 7-6 SD/MMC CMDARG Bit Values

CMDARG Bits	Contents	Value
31	R/W flag	1
30-28	Function Number	0, for CCCR access
27	RAW flag	1, read after write
26	Don’t care	-
25-9	Register address	0x0D
8	Don’t care	-
7-0	Write Data	Function number to be resumed

   e. Write the block size in the BLKSIZ register @0x1C; data will be transferred in units of this block size.
   f. Write the byte count in the BYTCNT register @0x20. This is the total size of the data; that is, the remaining bytes to be transferred. It is the responsibility of the software to handle the data.
   g. Program Command registers; similar to a block transfer. For details, refer to “Single-Block or Multiple-Block Read” and “Single-Block or Multiple-Block Write”.
   h. When the Command register is programmed, the command is sent and the function resumes data transfer. Read the DF flag (Resume Data Flag). If it is 1, then the function has data for the transfer and will begin a data transfer as soon as the function or memory is resumed. If it is 0, then the function has no data for the transfer.
   i. If the DF flag is 0, then in case of a read, the SD/MMC Controller waits for data. After the data timeout period, it gives a data timeout error.

Read_Wait Sequence

Read_wait is used with only the SDIO card and can temporarily stall the data transfer-either from function or memory—and allow the host to send commands to any function within the SDIO device. The host can stall this transfer for as long as required. The SD/MMC Controller provides the facility to signal this stall transfer to the card. The steps for doing this are:

1. Check if the card supports the read_wait facility; read SRW (bit 2) of the CCCR register @0x08. If this bit is 1, then all functions in the card support the read_wait

facility. Use CMD52 to read this bit.

2. If the card supports the read_wait signal, then assert it by setting the read_wait (bit 6) in the CTRL register @0x00.
3. Clear the read_wait bit in the CTRL register.

Controller/DMA/FIFO Reset Usage

Communication with the card involves the following:

- Controller – Controls all functions of the SD/MMC controller.
- FIFO – Holds data to be sent or received.
- DMA – If DMA transfer mode is enabled, then transfers data between system memory and the FIFO.
- Controller reset – Resets the controller by setting the controller_reset bit (bit 0) in the CTRL register; this resets the CIU and state machines, and also resets the BIU-to-CIU interface. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- FIFO reset – Resets the FIFO by setting the fifo_reset bit (bit 1) in the CTRL register; this resets the FIFO pointers and counters of the FIFO. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.
- DMA reset – Resets the internal DMA controller logic by setting the dma_reset bit (bit 2) in the CTRL register, which abruptly terminates any DMA transfer in process. Since this reset bit is self-clearing, after issuing the reset, wait until this bit is cleared.

The following are recommended methods for issuing reset commands:

- Non-DMA transfer mode – Simultaneously sets controller_reset and fifo_reset; clears the RAWINTS register @0x44 using another write in order to clear any resultant interrupt.
- DMA mode – Sets controller_reset and fifo_reset; waits until dma_req goes inactive (the Status register indicates the value of this signal). Resets the FIFO again. Clears the interrupts by clearing the RAWINTS register @0x44 using another write in order to clear any resultant interrupt. You also need to reset and reprogram the channel(s) of the DW_dmac controller that are interfaced to the SD/MMC Controller.

  In external DMA transfer mode, even when the FIFO pointers are reset, if there is a DMA transfer in progress, it could push or pop data to or from the FIFO; the DMA itself completes correctly. In order to clear the FIFO, the software should issue an additional FIFO reset and clear any FIFO underrun or overrun errors in the RAWINTS register caused by the DMA transfers after the FIFO was reset.

Error Handling

The SD/MMC controller implements error checking; errors are reflected in the RAWINTS register @0x44 and can be communicated to the software through an interrupt, or the software can poll for these bits. Upon power-on, interrupts are disabled (int_enable in the CTRL register is 0), and all the interrupts are masked (bits 0-31 of the INTMASK register; default is 0).

Error handling:

- Response and data timeout errors – For response timeout, software can retry the command. For data timeout, the SD/MMC controller has not received the data start bit – either for the first block or the intermediate block – within the timeout period, so software can either retry the whole data transfer again or retry from a specified block onwards. By reading the contents of the TCBCNT later, the software can decide how many bytes remain to be copied.
- Response errors – Set when an error is received during response reception. In this case, the response that copied in the response registers is invalid. Software can retry the command.
- Data errors – Set when error in data reception are observed; for example, data CRC, start bit not found, end bit not found, and so on. These errors could be set for any block-first block, intermediate block, or last block. On receipt of an error, the software can issue a STOP or ABORT command and retry the command for either whole data or partial data.

- Hardware locked error – Set when the SD/MMC controller cannot load a command issued by software. When software sets the start_cmd bit in the CMD register, the SD/MMC controller tries to load the command. If the command buffer is already filled with a command, this error is raised. The software then has to reload the command.
- FIFO underrun/overrun error – If the FIFO is full and software tries to write data in the FIFO, then an overrun error is set. Conversely, if the FIFO is empty and the software tries to read data from the FIFO, an underrun error is set. Before reading or writing data in the FIFO, the software should read the fifo_empty or fifo_full bits in the Status register.
- Data starvation by host timeout – Raised when the SD/MMC controller is waiting for software intervention to transfer the data to or from the FIFO, but the software does not transfer within the stipulated timeout period. Under this condition and when a read transfer is in process, the software should read data from the FIFO and create space for further data reception. When a transmit operation is in process, the software should fill data in the FIFO in order to start transferring data to the card.
- CRC Error on Command – If a CRC error is detected for a command, the CE-ATA device does not send a response, and a response timeout is expected from the SD/MMC controller. The ATA layer is notified that an MMC transport layer error occurred.

Note:

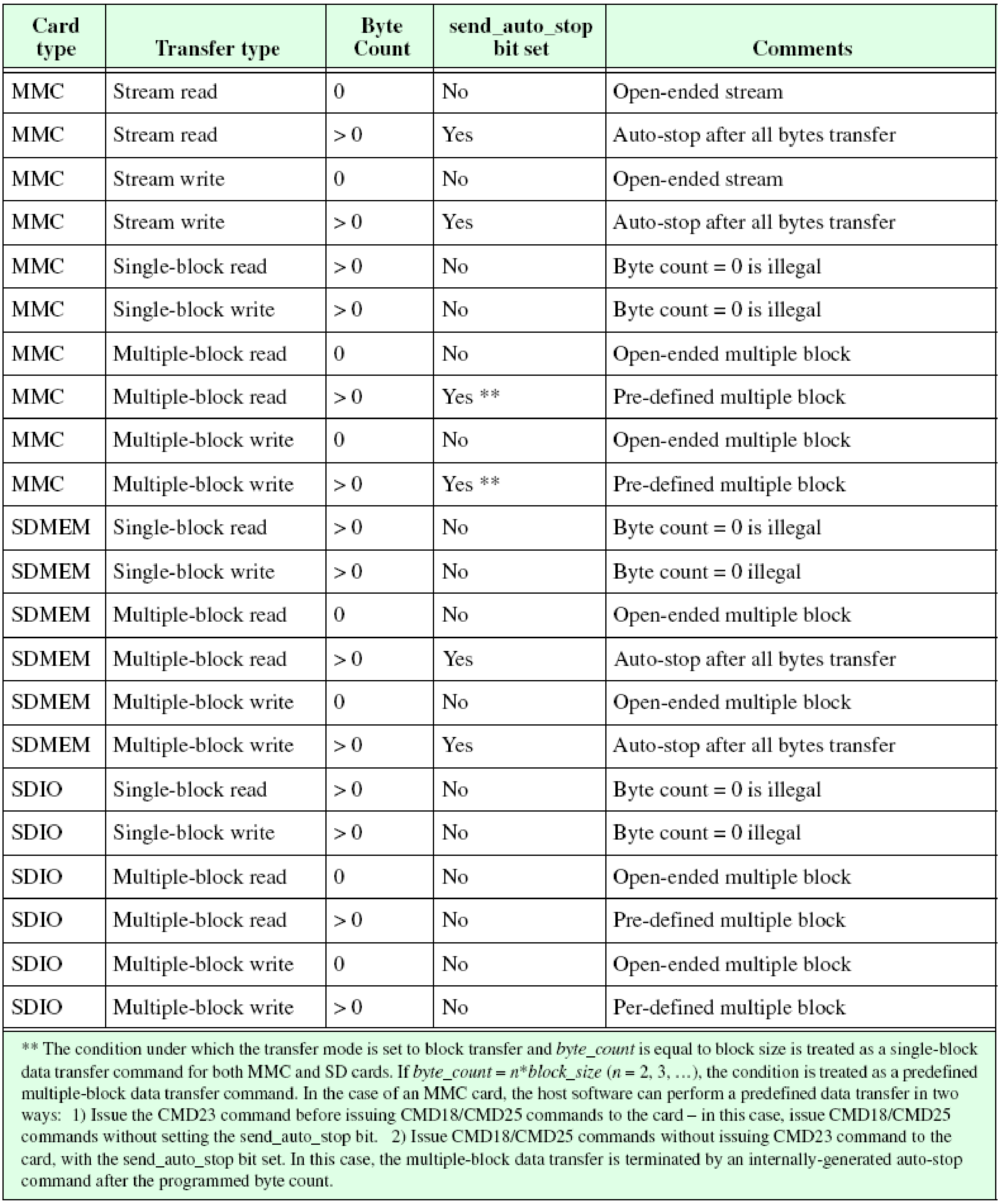
During a multiple-block data transfer, if a negative CRC status is received from the device, the data path signals a data CRC error to the BIU by setting the data CRC error bit in the RINTSTS register. It then continues further data transmission until all the bytes are transmitted.

Auto-Stop

The SD/MMC controller internally generates a stop command and is loaded in the command path when the send_auto_stop bit is set in the Command register. The auto-stop command helps to send an exact number of data bytes using a stream read or write for the MMC, and a multiple-block read or write for SD memory transfer for SD cards.

The software should set the send_auto_stop bit according to details listed in Table 6-7.

Table 7-7 SD/MMC Auto-Stop Generation condition list



Card type	Transfer type	Byte Count	send_auto_stop bit set	Comments
MMC	Stream read	0	No	Open-ended stream
MMC	Stream read	> 0	Yes	Auto-stop after all bytes transfer
MMC	Stream write	0	No	Open-ended stream
MMC	Stream write	> 0	Yes	Auto-stop after all bytes transfer
MMC	Single-block read	> 0	No	Byte count = 0 is illegal
MMC	Single-block write	> 0	No	Byte count = 0 is illegal
MMC	Multiple-block read	0	No	Open-ended multiple block
MMC	Multiple-block read	> 0	Yes **	Pre-defined multiple block
MMC	Multiple-block write	0	No	Open-ended multiple block
MMC	Multiple-block write	> 0	Yes **	Pre-defined multiple block
SDMEM	Single-block read	> 0	No	Byte count = 0 is illegal
SDMEM	Single-block write	> 0	No	Byte count = 0 illegal
SDMEM	Multiple-block read	0	No	Open-ended multiple block
SDMEM	Multiple-block read	> 0	Yes	Auto-stop after all bytes transfer
SDMEM	Multiple-block write	0	No	Open-ended multiple block
SDMEM	Multiple-block write	> 0	Yes	Auto-stop after all bytes transfer
SDIO	Single-block read	> 0	No	Byte count = 0 is illegal
SDIO	Single-block write	> 0	No	Byte count = 0 illegal
SDIO	Multiple-block read	0	No	Open-ended multiple block
SDIO	Multiple-block read	> 0	No	Pre-defined multiple block
SDIO	Multiple-block write	0	No	Open-ended multiple block
SDIO	Multiple-block write	> 0	No	Per-defined multiple block

** The condition under which the transfer mode is set to block transfer and *byte_count* is equal to block size is treated as a single-block data transfer command for both MMC and SD cards. If *byte_count* = *n***block_size* (*n* = 2, 3, …), the condition is treated as a predefined multiple-block data transfer command. In the case of an MMC card, the host software can perform a predefined data transfer in two ways: 1) Issue the CMD23 command before issuing CMD18/CMD25 commands to the card – in this case, issue CMD18/CMD25 commands without setting the send_auto_stop bit. 2) Issue CMD18/CMD25 commands without issuing CMD23 command to the card, with the send_auto_stop bit set. In this case, the multiple-block data transfer is terminated by an internally-generated auto-stop command after the programmed byte count.

The following list conditions for the auto-stop command.

- Stream read for MMC card with byte count greater than 0 – The SD/MMC controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent out when the last byte of data is read from the card and no extra data byte is received. If the byte count is less than 6 (48 bits), a few extra data bytes are received from the card before the end bit of the stop command is sent.
- Stream write for MMC card with byte count greater than 0 – The SD/MMC controller generates an internal stop command and loads it into the command path so that the end bit of the stop command is sent when the last byte of data is transmitted on the card bus and no extra data byte is transmitted. If the byte count is less than 6 (48 bits), the data path transmits the data last in order to meet the above condition.
- Multiple-block read memory for SD card with byte count greater than 0 – If the block

size is less than 4 (single-bit data bus), 16 (4-bit data bus), or 32 (8-bit data bus), the auto-stop command is loaded in the command path after all the bytes are read. Otherwise, the top command is loaded in the command path so that the end bit of the stop command is sent after the last data block is received.

- Multiple-block write memory for SD card with byte count greater than 0 – If the block size is less than 3 (single-bit data bus), 12 (4-bit data bus), or 24 (8-bit data bus), the auto-stop command is loaded in the command path after all data blocks are transmitted. Otherwise, the stop command is loaded in the command path so that the end bit of the stop command is sent after the end bit of the CRC status is received.
- Precaution for host software during auto-stop – Whenever an auto-stop command is issued, the host software should not issue a new command to the SD/MMC controller until the auto-stop is sent by the SD/MMC controller and the data transfer is complete. If the host issues a new command during a data transfer with the auto-stop in progress, an auto-stop command may be sent after the new command is sent and its response is received; this can delay sending the stop command, which transfers extra data bytes. For a stream write, extra data bytes are erroneous data that can corrupt the card data. If the host wants to terminate the data transfer before the data transfer is complete, it can issue a stop or abort command, in which case the SD/MMC controller does not generate an auto-stop command.

Chapter 8 Video Input Processor(VIP)

8.1 Design Overview

8.1.1 Overview

The Video Input Processor receives the data from Camera or CCIR656 encoder, and transfers the data into system main memory by AHB bus.

8.1.2 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, 8M pixel
- Support YUYV/UYVY format input
- In Sensor Mode, support Vsync and Href High active or Low active configurable

Notes: vsync porality control is programmable by bit 7 of register CPU_APB_REG5, refer to Chapter 34 (General Register File in CPU System) for detailed information

8.2 Architecture

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

8.2.1 Block Diagram

The VIP comprises with:

- AHB Slave – Host configure the VIP_Reg via the AHB Slave
- AHB Master – VIP transmit the data to chip memory via the AHB Master
- VIP_REG – The register bank store the status and configuration information
- YUV Interface – translate the input video data into the requisite data format.
- DMA Control – Manage the memory buffer
- Buffer control and line buffer – store the translated video data.

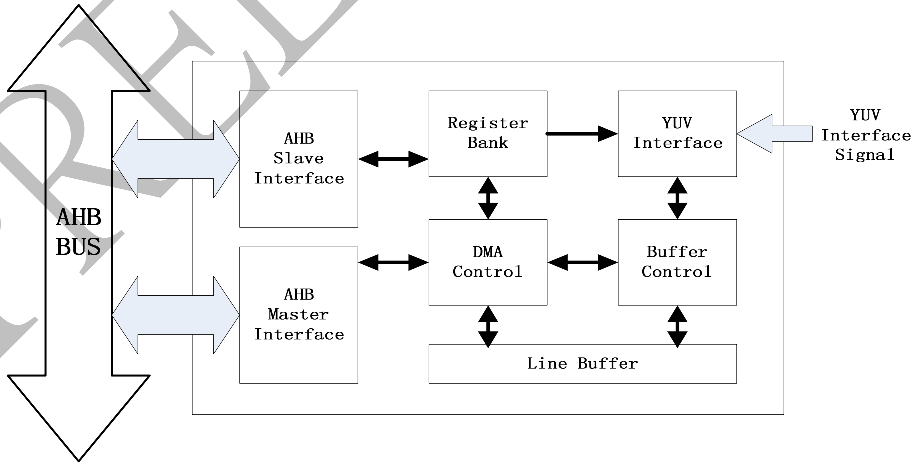


Fig. 8-1 VIP Block Diagram

8.3 Registers

This section describes the registers of the design.

8.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
VIP_AHBR_CTRL	0x00	W	0x1	AHB write control register

VIP_INT_MASK	0x04	W	0x0	Interrupt Mask register
VIP_INT_STS	0x08	W	0x0	Interrupt status register
VIP_STS	0x0C	W	0x0	VIP Status register
VIP_CTRL	0x10	W	0x0	VIP control register
VIP_CAPTURE_F1SA_Y	0x14	W	0xFFFFFFFF	Capture data frame 1 start address for Y
VIP_CAPTURE_F1SA_UV	0x18	W	0xFFFFFFFF	Capture data frame 1 start address for UV
VIP_CAPTURE_F1SA_Cr	0x1C	W	0xFFFFFFFF	Capture data frame 1 start address for Cr
VIP_CAPTURE_F2SA_Y	0x20	W	0xFFFFFFFF	Capture data frame 2 start address for Y
VIP_CAPTURE_F2SA_UV	0x24	W	0xFFFFFFFF	Capture data frame 2 start address for UV
VIP_CAPTURE_F2SA_Cr	0x28	W	0xFFFFFFFF	Capture data frame 2 start address for Cr
VIP_FB_SR	0x2C	W	0xB	Frame buffer status register for capturing raw data
VIP_FS	0x30	W	0x02D001E6	Frame data size register
VIP_CROP	0x38	W	0x0	Cropping start upper left point to other little resolution
VIP_CRM	0x3C	W	0x0	Y/U/V color modification
VIP_RESET	0x40	W	0x0	Capture engine reset
VIP_L_SFT	0x44	W	0x0	Line shifter from first line

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

8.3.2 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
31:3	-	-	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

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

8.4 Functional Description

This chapter is used to illustrate the operational behavior of how VIP module works. VIP module receive the sensor or ccir656 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.

8.4.1 Operation

Software/Hardware Reset

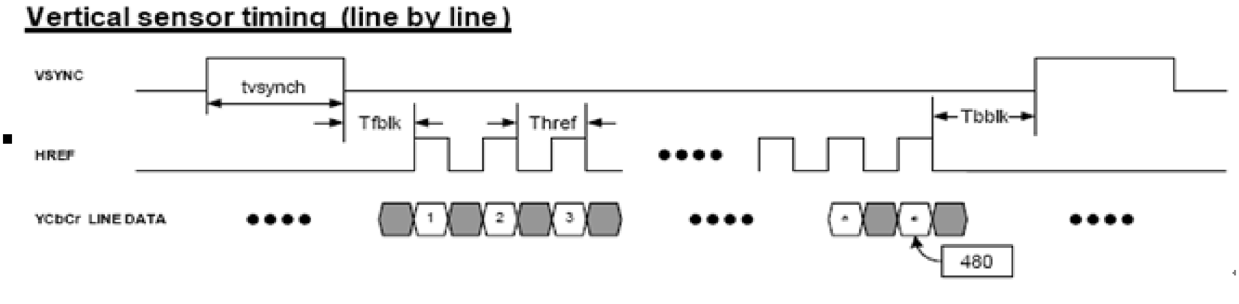
When RESETN pin is set to low, it will cause everything including both VIP and AHB modules to be reset to default state immediately.

While configuring the VIP_RESET(offset 0x40) register with 0x76543210, VIP will be software reset after cycles of AHB.

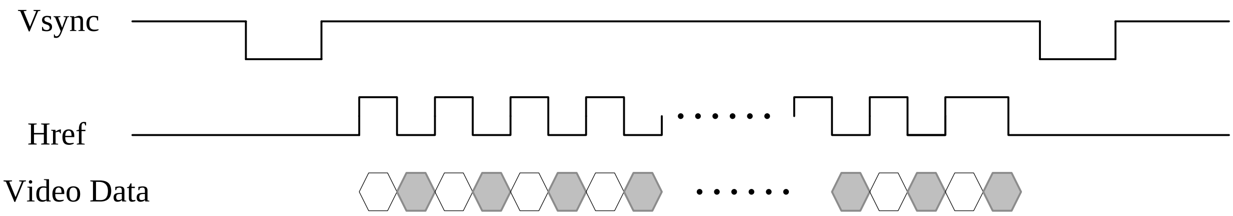
Input data format

The VIP module support the YUV422 and CCIR656 data input.

- Vsync Low active as below:



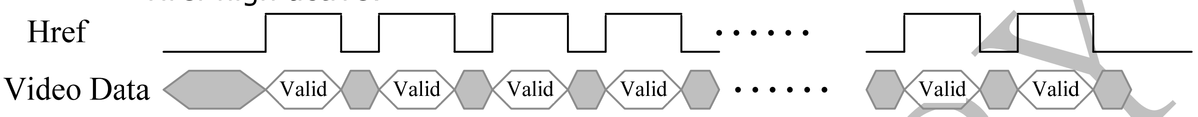
- Vsync High active :



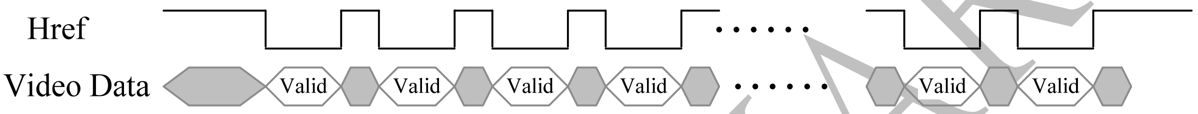
Notes:

** Set the Vsync Valid porality. Refer to Chapter 34 (General Register File in CPU System), bit[7] in the Register of CPU_APB_REG5 for detailed descriptions*

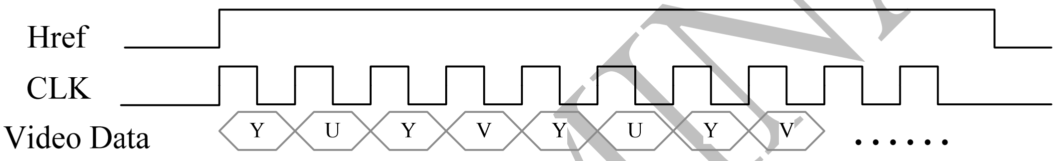
- Href high active:



- Href Low active



- Y first



- U first

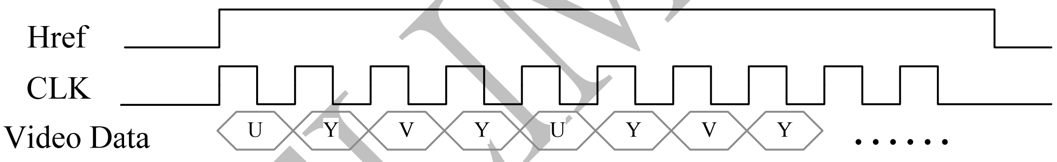


Fig. 8-2 VIP Input Signal Timing

Initial Configuration

After HW/SW reset, host must initially configure VIP by control registers via AHB bus. The configurations include external device, frame data start address, frame size, output format… etc.

It is recommended that you can configure the VIP register as the follow the sequence:

1. Set the VIP_AHBR_CTRL to config the AHB Burst Length(INCR, INCR8 or INCR16)
2. Set the VIP_INT_MASK to config the VIP interrupt mask
3. Set the VIP_CAPTURE_F1SA_Y, VIP_CAPTURE_F1SA_UV, VIP_CAPTURE_F2SA_Y, VIP_CAPTURE_F2SA_UV to config the Y/UV start address of the frame1 and frame 2.
4. Set the VIP_CTRL, to config the parameter of the VIP, for example, ccir656 format、vip work mode(ping-pong/continue/one frame)、input data order(y or uv first)…etc. Don't set the VIP_enable bit (bit[0]) to 1 at this time.
5. Set the VIP_FS to config the frame size
6. Set the VIP_FB_SR to clear the frame buffer status
7. In the end, start the vip by setting the VIP_enable bit(bit[0] in VIP_CTRL) to 1.

VIP module can work in three modes: one frame stop mode、ping-pong mode、continuous mode.

One frame stop mode (only frame1 can be used in this mode)

In this mode, only need to set VIP_CAPTURE_F1SA_<Y,UV> as start memory address of the continuous memory. Before trigger VIP to start capture, frame1 buffer status in VIP_FB_SR register(bit 0) must be clear to 1'b0. Then configure register VIP_CTRL(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_UV separately.

Ping-Pong mode

In this mode, need to set VIP_CAPTURE_F1SA_<Y, UV> and VIP_CAPTURE_F2SA_<Y, UV>. Before trigger VIP to start captue, frame1 & frame2 buffer status in VIP_FB_SR register(bit1:0) must be clear to 2'b00. Then configure VIP_CTRL 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 new F1 address) without any stop and so on for the following frames. But if host did not update the frame buffer address, the VIP will cover the pre-frame data stored in the memory with the following frame data. And if host did not clear the frame status, the VIP will stop after both 2 frame buffer (F1&F2) are at data ready state(bit[1:0] of VIP_FB_SR register is 2'b11).

Continuous mode

In this mode, need to set VIP_CAPTURE_F1SA_<Y, UV> and VIP_CAPTURE_F2SA_<Y, UV>. Before trigger VIP to start captue, frame1 & frame2 buffer status in VIP_FB_SR register(bit1:0) must be clear to 2'b00. Then configure VIP_CTRL to start ping-pong mode. After one frame(F1) captured, VIP will start to capture the next frame(F2) automatically. If you setup register VIP_INT_MASK to be value 0x1 you will get a complete interrupt from VIP after every frame end. 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 9 LCD Controller

9.1 Design Overview

9.1.1 Overview

LCD Controller has an ahb slave and master module that connect to standard ahb bus. The LCD Controller is a bridge used between host(Memory) and LCD Panel.There is ahb master interface for the LCD Controller that can increase the data transfer rate between Host(memory) and LCD Panel. It is designed to cover a wider area of application such as cellular phone, MP3, MP4,PDA,etc

9.1.2 Features

- Support AHB Slave Interface
- Support AHB Master Interface
- Support one SCALE window and one no SCALE window.
- YUV422/YUV420/RGB565/RGB888 Input are Supported in SCALE window
- RGB565/RGB888 Input and 4 AREAS are Supported in NO SCALE window
- Support Virtual Display.
- Build in scaler engine from 1/8 to 8
- Support 16 Level grade alpha blending and transparent operation.
- Support Blank/Black Function
- Support LCD Pannel resolution up to WVGA(1280x720).
- Compatible with MCU Pannel.
- Support MCU PANNEL Bypass Mode and SCALE Mode
- Compatible with RGB Delta/no-Delta Pannel
- Compatible with RGB Series/Parallel Output
- Support Interlace and Progressive Output
- Support CCIR656 interface

9.2 Architecture

9.2.1 Overview

FUNCTION DIAGRAM

The Function diagram is shown in Fig.8-1, there are 3 Module in LCD Controller, AHB_INF, DSP_PRS and DSP_CTRL.

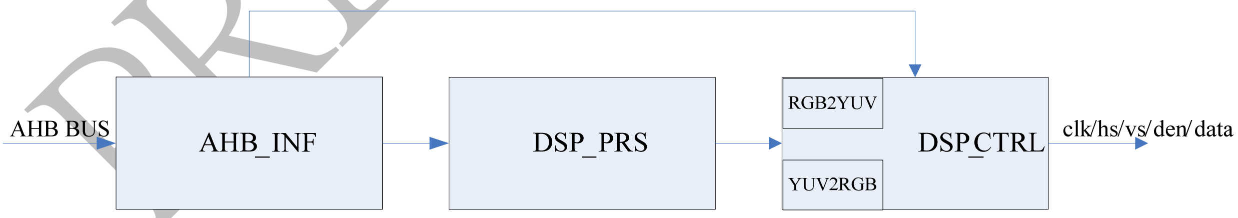


Fig. 9-1 LCD Controller Function Diagram

9.2.2 Block Description

- AHB_INF is the interface with Host, It has AHB master/slave Interface.
- DSP_PRS will achieve scale up/scale down Function.
- DSP_CTRL will display control module for RGB PANNEL or MCU PANNEL.

9.3 Register Definition

9.3.1 Register Summary

Name	Offset	Size	Reset	Description

			Value	
SYS_CONFIG	0x0000	W	0x0	SYSTEM configure register
WIN0_VIR	0x0004	W	0x0	WIN0 VIRTUAL DISPLAY Width
WIN0_YRGB_MST	0x0008	W	0x0	Win0 YRGB memory start address
WIN0_CBR_MST	0x000c	W	0x0	Win0 Cbr memory start address
INT_LUT	0x0010	W	0xff4000	Interrupt lookup table
WIN0_ACT_INFO	0x0014	W	0x0	Win0 active window widht and height
WIN1_VIR0	0x0018	W	0x0	Win1 AREA Virtual display width
WIN1_VIR1	0x001c	W	0x0	Win1 AREA Virtual display width
WIN1_AREA0_MST	0x0020	W	0x0	Win1 AREA0 memory start address
WIN1_AREA1_MST	0x0024	W	0x0	Win1 AREA1 memory start address
WIN1_AREA2_MST	0x0028	W	0x0	Win1 AREA2 memory start address
WIN1_AREA3_MST	0x002c	W	0x0	Win1 AREA3 memory start address
DSP_HTOTAL	0x0030	W	0x17F	Dsiplay pannel horizontal total
DSP_HS_END	0x0034	W	0x17	Display Pannel hsync start point
DSP_HACT_ST	0x0038	W	0x2B	Display Pannle horizontal enable start point
DSP_HACT_END	0x003c	W	0x16A	Display Pannel horizontal enable end point
DSP_VTOTAL	0x0040	W	0x107	Dsplay Pannel verital total.
DSP_VS_END	0x0044	W	0x2	Display Pannel vertical sync end point, start point is always from0
DSP_VACT_ST	0x0048	W	0xD	Display Pannel veritcal enable start line
DSP_VACT_END	0x004c	W	0xFC	Display Pannel vertical enable end line
DSP_VS_ST_F1	0x0050	W	0x0	Display Pannle filed1 start line, only in Interlace Mode.
DSP_VS_END_F1	0x0054	W	0x0	Display Pannel field1 end line, only in Interlce Mode.
DSP_VACT_ST_F1	0x0058	W	0x0	Display Pannel field 1 vertical enable start line, only in Interlce Mode.
DSP_VEND_ST_F1	0x005c	W	0x0	Dsiplay Pannel filed 1 vertical enable end line, Only in Interlace Mode.
DSP_WIN0_ST	0x0060	W	0x0	Display win0 horizontal and veritical start point on the pannel
DSP_WIN0_INFO	0x0064	W	0x0	Display win0 width and height on the pannel
SD_FACTOR	0x0068	W	0x0	Horizontal and vertical scale down factor setting

SP_FACTOR	0x006c	W	0x0	Horizontal and vertical scale up factor setting
DSP_WIN1_AREA0_ST	0x0070	W	0x0	Win1 AREA0 horizontal and vertical start point on Pannel
DSP_WIN1_AREA1_ST	0x0074	W	0x0	Win1 AREA0 horizontal width and vertical height on Pannel
DSP_WIN1_AREA2_ST	0x0078	W	0x0	Win1 AREA1 horizontal and vertical start point on Pannel
DSP_WIN1_AREA3_ST	0x007c	W	0x0	Win1 AREA1 horizontal width and vertical height on Pannel
DSP_WIN1_AREA0_INFO	0x0080	W	0x0	Win1 AREA2 horizontal and vertical start point on Pannel
DSP_WIN1_AREA1_INFO	0x0084	W	0x0	Win1 AREA2 horizontal width and vertical height on Pannel
DSP_WIN1_AREA2_INFO	0x0088	W	0x0	Win1 AREA3 horizontal and vertical start point on Pannel
DSP_WIN1_AREA3_INFO	0x008c	W	0x0	Win1 AREA3 horizontal width and vertical height on Pannel
REG_CFG_DONE	0x0090	W	0x0	REGISTER CONFIG FINISH REGISTER.
DSP_CTRL_REG0	0x0094	W	0x0	Display control register
DSP_CTRL_REG1	0x0098	W	0x80000000	Display Control register
WIN1_WATERMARK	0x009c	W	0x0	Win1 display watermark
MCU_TIMING_CTRL	0x00a0	W	0x0	MCU display timing control register
RAM_CEN_CTRL	0x00a4	W	0x3	RAM CEN Control Register
WIN0_YRGB_WPORT	0x0100	W	--	WIN0 YRGB DATA Write only PORT
WIN0_CBR_WPORT	0x0200	W	--	WIIN0 Cbr Data Write only port
WIN1_AREA0_WPORT	0x0300	W	--	WIN1 AREA0 Data Write Only Port
WIN1_AREA1_WPORT	0x0400	W	--	WIN1 AREA1Data Write Only Port
WIN1_AREA2_WPORT	0x0500	W	--	WIN1 AREA2 Data Write Only Port
WIN1_AREA3_WPORT	0x0600	W	--	WIN1 AREA3 Data Write Only Port
MCU_BYPASS_WPORT	0x0700	W	--	MCU BYPASS MODE, DATA Write Only Port

9.3.2 Detailed Register Description

SYS_CONFIG

Address: Operational Base + offset(0x0000)
LCDC System Configure Register

Bit	Attr	Reset Value	Description

31	W/R	0x0	MCU PANNEL SELECT
30	W/R	0x0	MCU PANNEL BYPASS MODE
29	W/R	0x0	MCU PANNEL RS Control
28:26	W/R	0x000	AHB Master Burst Control
25:23	W/R	0x000	AHB Master Size control
22	W/R	0x0	AHB Master Write Control
21:17	W/R	0x00000	AHB Master INCR Control
16	W/R	0x0	AHB Master field polarity
15	W/R	0x0	AHB Master jump read mode
14	W/R	0x0	Win1 AREA3 enable control
13	W/R	0x0	Win1 AREA2 enable control
12	W/R	0x0	Win1 AREA1 enable control
11	W/R	0x0	Win1 AREA0 enable control
10	Reserved	-----	
9	W/R	0x0	Win0 enable control
8	W/R	0x0	Win1 AREA3 Display Format
7	W/R	0x0	Win1 AREA2 Display Format
6	W/R	0x0	Win1 AREA1 Display Format
5	W/R	0x0	Win1 AREA0 Display Format
4:2	W/R	0x000	Win0 Display Input Format 3'b000 : RGB888 3'b001 : RGB565 3'b010 : YUV422 3'b011 : YUV4200 3'b100 : YUV4201 3'b101 : YUV420M 3'b110 : YUV444
1	W/R	0x0	Win1 AHB Master transfer control
0	W/R	0x0	Win0 AHB Master transfer control

WIN0_VIR

Address: Operational Base + offset(0x0004)

Win0 Virtual display width control register

Bit	Attr	Reset Value	Description
31:24	Reserved	----	
23	R/W	0x0	LCDC output tristate control
22:21	R/W	0x0	LCDC Vertical Deflicker Filter
20:19	R/W	0x0	LCDC Horizontal Deflicker Filter
18:16	Reserved	----	
15:0	W/R	0x0	Win0 Virtual Display Width

WIN0_YRGB_MST

Address: Operational Base + offset(0x0008)

Bit	Attr	Reset Value	Description
31:0	W/R	0x00	AHB Master YRGB data Start Point in Memory

WIN0_CBR_MST

Address: Operational Base + offset(0x000c)

Bit	Attr	Reset Value	Description
31:0	W/R	0x00	AHB Master Cbr data Start Point in

			Memory

INT_LUT

Address: Operational Base + offset(0x0010)

Interrupt source occurs look up table

Bit	Attr	Reset Value	Description
31	W/R	0x0	Frame start interrupt clear
30	W/R	0x0	Horizontal start interrupt clear
29	W/R	0x0	WIN0 YRGB EmptyInterrupt clear
28	W/R	0x0	WIN0 Cbr Empty Interrupt clear
27	W/R	0x0	WIN1 AREA0 Empty Interrupt clear
26	W/R	0x0	WIIN1 AREA1 Empty Interrupt clear
25	W/R	0x0	WIN1 AREA2 Empty Interrupt clear
24	W/R	0x0	WIN1 AREA3 Empty Interrupt clear
23	W/R	0x1	Frame start interrupt mask
22	W/R	0x1	Horizontal start interrupt mask
21	W/R	0x1	WIN0 YRGB EmptyInterrupt mask
20	W/R	0x1	WIN0 Cbr Empty Interrupt mask
19	W/R	0x1	WIN1 AREA0 Empty Interrupt mask
18	W/R	0x1	WIN1 AREA1 Empty Interrupt mask
17	W/R	0x1	WIN1 AREA2 Empty Interrupt mask
16	W/R	0x1	WIN1 AREA3 Empty Interrupt mask
15	W	0x0	AHB Master Error Interrupt clear
14	W/R	0x1	AHB Master Error Interrupt Mask
13:9	Reserved	----	
8	R	0x0	AHB Master Error Interrupt
7	R	0x0	Frame start interrupt
6	R	0x0	Horizontal start interrupt
5	R	0x0	WIN0 YRGB EmptyInterrupt
4	R	0x0	WIN0 Cbr Empty Interrupt
3	R	0x0	WIN1 AREA0 Empty Interrupt
2	R	0x0	WIIN1 AREA1 Empty Interrupt
1	R	0x0	WIN1 AREA2 Empty Interrupt
0	R	0x0	WIN1 AREA3 Empty Interrupt

WIIN0_ACT_INFO

Address: Operational Base + offset(0x0014)

WIN0 Active window width and height.

Bit	Attr	Reset Value	Description
31:16	W/R	0x00	WIN0 Active window height
15:0	W/R	0x00	WIN0 Active window width

WIIN1_VIR0

Address: Operational Base + offset(0x0018)
WIN1 AREA0 and AREA1 Virtual display width.

Bit	Attr	Reset Value	Description
31:16	W/R	0x00	WIN1 AREA1 Virtual display width
15:0	W/R	0x00	WIN1 AREA0 Virtual display width

WIIN1_VIR1

Address: Operational Base + offset(0x001c)
WIN1 AREA2 and AREA3 Virtual display width.

Bit	Attr	Reset Value	Description
31:16	W/R	0x00	WIN1 AREA3 Virtual display width
15:0	W/R	0x00	WIN1 AREA2 Virtual display width

WIN1_AREA0_MST

Address: Operational Base + offset(0x0020)
WIN1 AREA0 Data start address in Memory

Bit	Attr	Reset Value	Description
31:0	W/R	0x00	AHB Master WIN1 AREA0 data Start Point in Memory

WIN1_AREA1_MST

Address: Operational Base + offset(0x0024)
WIN1 AREA1 Data start address in Memory

Bit	Attr	Reset Value	Description
31:0	W/R	0x00	AHB Master WIN1 AREA1 data Start Point in Memory

WIN1_AREA2_MST

Address: Operational Base + offset(0x0028)
WIN1 AREA2 Data start address in Memory

Bit	Attr	Reset Value	Description
31:0	W/R	0x00	AHB Master WIN1 AREA2 data Start Point in Memory

WIN1_AREA3_MST

Address: Operational Base + offset(0x002c)
WIN1 AREA3 Data start address in Memory

Bit	Attr	Reset Value	Description
31:0	W/R	0x00	AHB Master WIN1 AREA3 data Start Point in Memory

DSP_HTOTAL

Address: Operational Base + Offset(0x0030)
Pannel display Horizontal Total Register.

Bit	Attr	Reset Value	Description
31:11	Reserved	---	
10:0	W/R	0x17f	Pannel display horizontal total.

DSP_HS_END

Address: Operational Base + Offset(0x0034)
Pannel Display Horizontal Sycn end Register, the Horizontal Sync start is always from 0..

Bit	Attr	Reset Value	Description
31:11	Reserved	---	
10:0	W/R	0x17	Pannel display horizontal sync end register.

DSP_HACT_ST
Address: Operational Base + Offset(0x0038)
Pannel display Horizontal active start Register.

Bit	Attr	Reset Value	Description
31:11	Reserved	---	
10:0	W/R	0x2B	Pannel display horizontal active start register.

DSP_HACT_END
Address: Operational Base + Offset(0x003c)
Pannel display Horizontal active end Register.

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x16A	Pannel display horizontal active end register.

DSP_VTOTAL
Address: Operational Base + Offset(0x0040)
Pannel display Vertical line Total Register.

Bit	Attr	Reset Value	Description
31:11	Reserved	---	
10:0	W/R	0x107	Pannel display vertical line total.

DSP_VS_END
Address: Operational Base + Offset(0x0044)
Pannel display vertical Sycn end Register, the vertical Sync start is always from 0.

Bit	Attr	Reset Value	Description
31:11	Reserved	---	
10:0	W/R	0x002	Pannel display veritcal sync end register. When in the Interlace Mode, this register is used as the Pannel display field0 virtical sync end point

DSP_VACT_ST
Address: Operational Base + Offset(0x0048)
Pannel display Virtical active start Register.

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x00D	Pannel display virtical active start register. When in the Interlace Mode, this register is used as the Pannel display field0 vertical acitive start point

DSP_VACT_END
Address: Operational Base + Offset(0x004c)
Pannel display Virtical active end Register.

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x0FC	Pannel display virtical active end register. When in the Interlace Mode, This register is used as the Pannel display field0 vertical active end point

DSP_VS_ST_F1

Address: Operational Base + Offset(0x0050)

Pannel display Vertical sync start point of field1. It is used only in the Interlace Mode

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x00	Pannel display vertical sync start point of Filed1. only used in Interlace Mode

DSP_VS_END_F1

Address: Operational Base + Offset(0x0054)

Pannel display vertical Sycn end of field1, only used in Interlace Mode.

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x00	Pannel display veritcal sync end register of field1. only used in the Interlace Mode,

DSP_VACT_ST_F1

Address: Operational Base + Offset(0x0058)

Pannel display Virtical active start Register of field1. only used in Interlace Mode

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x00	Pannel display virtical active start register of field1. Only used in the Interlace Mode.

DSP_VACT_END_F1

Address: Operational Base + Offset(0x005c)

Pannel display Virtical active end Register of field1. only used in Interlace Mode.

Bit	Attr	Reset Value	Description
31:11	Reserved	--	
10:0	W/R	0x00	Pannel display virtical active end register of field1. only used in the Interlace Mode.

DSP_WIN0_ST

Address: Operational Base + Offset(0x0060)

Win0 horizontal/vertical start point displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win0 veritcal display start point on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win0 horizontal display start point on the Pannel.

DSP_WIN0_INFO

Address: Operational Base + Offset(0x0064)

Win0 width/height displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win0 height display on the Pannel.
15:11	Reserved	----	
10:0	W/R	0x00	Win0 width display on the Pannel.

SD_FACTOR

Address: Operational Base + Offset(0x0068)
Win0 horizontal/vertical scale down factor.

Bit	Attr	Reset Value	Description
31:16	W/R	0x00	Win0 vertical scale down factor, vsd_fct = ceil((dsp_win0/act_win0) * 2^15). When there is no scale down operation on it, it should be 0.
15:0	W/R	0x00	Win0 horizontal scale down factor. hsd_fct = ceil((dsp_win0/act_win0) * 2^15). When there is no scale down operation on it, It should be 0.

SP_FACTOR
Address: Operational Base + Offset(0x006c)
Win0 horizontal/vertical scale up factor.

Bit	Attr	Reset Value	Description
31:16	W/R	0x00	Win0 vertical scale up factor, vsp_fct = ceil((act_win0/dsp_win0) * 2^15). When there is no scale up operation on it, it should be 0.
15:0	W/R	0x00	Win0 horizontal scale up factor. hsp_fct = ceil((act_win0/dsp_win0) * 2^15). When there is no scale up operation on it, It should be 0.

DSP_WIN1_AREA0_ST
Address: Operational Base + Offset(0x0070)
Win1 AREA0 horizontal/vertical start point displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA0 veritcal display start point on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA0 horizontal display start point on the Pannel.

DSP_WIN1_AREA1_ST
Address: Operational Base + Offset(0x0074)
Win1 AREA1 horizontal/vertical start point displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA1 veritcal display start point on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA1 horizontal display start point on the Pannel.

DSP_WIN1_AREA2_ST
Address: Operational Base + Offset(0x0078)
Win1 AREA2 horizontal/vertical start point displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA2 veritcal display start point on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA2 horizontal display start point on the Pannel.

DSP_WIN1_AREA3_ST
Address: Operational Base + Offset(0x007c)
Win1 AREA0 horizontal/vertical start point displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA3 veritcal display start point on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA3 horizontal display start point on the Pannel.

DSP_WIN1_AREA0_INFO
Address: Operational Base + Offset(0x0080)
Win1 AREA0 width/height displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA0 height display on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA0 width display on the Pannel.

DSP_WIN1_AREA1_INFO
Address: Operational Base + Offset(0x0084)
Win1 AREA1 width/height displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA1 height display on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA1 width display on the Pannel.

DSP_WIN1_AREA2_INFO
Address: Operational Base + Offset(0x0088)
Win1 AREA2 width/height displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA2 height display on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA2 width display on the Pannel.

DSP_WIN1_AREA3_INFO
Address: Operational Base + Offset(0x008c)
Win1 AREA3 width/height displayed on the Pannel.

Bit	Attr	Reset Value	Description
31:27	Reserved	----	
26:16	W/R	0x00	Win1 AREA3 height display on the Pannel
15:11	Reserved	----	
10:0	W/R	0x00	Win1 AREA3 width display on the Pannel.

REG_CFG_DONE
Address: Operational Base + offset(0x0090)
Register Configure Done

Bit	Attr	Reset Value	Description
31:0	W	0x00	When all the register config finish, this

			register must be written to enable the configured register

DSP_CTRL_REG0
Address: Operational Base + Offset(0x0094)
Pannel display Control Register0

Bit	Attr	Reset Value	Description
31:28	W/R	0x0	WIN1 AREA3 Transparent factor
27:24	W/R	0x0	WIN1 AREA2 Transparent factor
23:20	W/R	0x0	WIN1 AREA1 Transparent factor
19:16	W/R	0x0	WIN1 AREA0 Transparent factor
15	W/R	0x0	WIN1 AREA3 Transparent enable
14	W/R	0x0	WIN1 AREA2 Transparent enable
13	W/R	0x0	WIN1 AREA1 Transparent enable
12	W/R	0x0	WIN1 AREA0 Transparent enable
11	W/R	0x0	DSP OUTUT PIN dclk polarity invert
10	W/R	0x0	DSP OUTPUT PIN den polarity invert
9	W/R	0x0	DSP OUTPUT PIN VSYNC polarity invert
8	W/R	0x0	DSP OUTPUT PIN HSYNC polarity invert
7	W/R	0x0	Top Layer Control Register 1:win0 is on the top of win1 0:win1 is on the top of win0
6:4	W/R	0x000	Combine with dsp_rgb666_en, { DSP_CTRL_REG0[6:4], MCU_TIMING_CTRL[26]}: Display Format Control Register 4’b0000: Parallel RGB888 output 4’b0001: Parallel RGB666 output 3’b0010: Parallel RGB565 output 3’b0100: Serial 16bit RGB888x 3’b0110: CCIR656 output 3’b1000: Serial RGB888 output without dumy cycle 3’b1100: serial RGB888 output with dumy cycle. Others: Reserved
3	W/R	0x0	Display Red and Green Data Swap Register
2	W/R	0x0	Display Red and Blue Data Swap Register
1	W/R	0x0	Interlace Display Control
0	W/R	0x0	Display Black Control. When this bit enable, the hs/vs/den is output, but data is black data

DSP_CTRL_REG1
Address: Operational Base + Offset(0x0098)
Pannel display Control Register0

Bit	Attr	Reset Value	Description
31	W/R	0x1	Display Blank Control. When this bit enable, the hs/vs/den is not output.
30	W/R	0x0	Dumy swap Control, When This bit enable, Dumy data swap with normal data
29	W/R	0x0	DROP LINE scale down Mode
28	W/R	0x0	DISPLAY DELTA SWAP ENABLE
27	W/R	0x0	Transparent factor from RAM Control. It

			is enable in the WIN1 AREA3 RGB888 Input Format
26	W/R	0x0	Transparent factor from RAM Control. It is enable in the WIN1 AREA2 RGB888 Input Format
25	W/R	0x0	Transparent factor from RAM Control. It is enable in the WIN1 AREA1 RGB888 Input Format
24	W/R	0x0	Transparent factor from RAM Control. It is enable in the WIN1 AREA0 RGB888 Input Format
23:16	W/R	0x00	Black Function Blue color
15:8	W/R	0x00	Black Function Green color
7:0	W/R	0x00	Black Function Red color

WIN1_WATERMARK

Address: Operational Base + Offset(0x009c)

WIN1 Watermark Control Register

Bit	Attr	Reset Value	Description
31	W/R	0x0	WIN1 AREA3 ahb loop swap control
30	W/R	0x0	WIN1 AREA2 ahb loop swap control
29	W/R	0x0	WIN1 AREA1 ahb loop swap control
28	W/R	0x0	WIN1 AREA0 ahb loop swap control
27	W/R	0x0	WIN1 AREA3 ahb 8bit swap control
26	W/R	0x0	WIN1 AREA3 ahb 16bit swap control
25	W/R	0x0	WIN1 AREA2 ahb 8bit swap control
24	W/R	0x0	WIN1 AREA2 ahb 16bit swap control
23	W/R	0x0	WIN1 AREA1 ahb 8bit swap control
22	W/R	0x0	WIN1 AREA1 ahb 16bit swap control
21	W/R	0x0	WIN1 AREA0 ahb 8bit swap control
20	W/R	0x0	WIN1 AREA0 ahb 16bit swap control
19	W/R	0x0	WIN0 Cbr ahb 8bit swap Control
18	W/R	0x0	WIN0 Cbr ahb 16bit swap Control
17	W/R	0x0	WIN0 YRGB ahb 8bit swap control
16	W/R	0x0	WIN0 YRGB ahb 16bit swap control
15	W/R	0x0	WIN0 Cbr ahb loop swap control
14	W/R	0x0	WIN0 YRGB ahb loop swap control
13	W/R	0x0	WIN0 YRGB ahb middle 8bit swap control
12	W/R	0x0	In WIN0 RGB565 mode, Red and Blue swap
11	W/R	0x0	In WIN1 AREA0 RGB565 mode, Red and Blue swap.
10	W/R	0x0	In WIN1 AREA1 RGB565 mode, Red and Blue swap.
9	W/R	0x0	In WIN1 AREA2 RGB565 mode, Red and Blue swap.
8	W/R	0x0	In WIN1 AREA3 RGB565 mode, Red and Blue swap.
7:0	W/R	0x00	WIN1 AREA0 ~ AREA3 Watermark

MCU_TIMING_CTRL

Address: Operational Base + Offset(0x00a0)

MCU TIMING Control register

Bit	Attr	Reset Value	Description
31	W/R	0x00	MCU HOLD Mode Frame Start Signal

30	W/R	0x00	MCU HOLD Mode Select
29	R	0x00	MCU HOLD Control
28	W/R	0x00	YUV2RGB Color Space BYPASS Controll
27	W/R	0x00	JPEG Color Space Control
26	W/R	0x0	Combine with dsp_rgb666_en, { DSP_CTRL_REG0[6:4], MCU_TIMING_CTRL[26]}: Display Format Control Register 4’b0000: Parallel RGB888 output 4’b0001: Parallel RGB666 output 3’b0010: Parallel RGB565 output 3’b0100: Serial 16bit RGB888x 3’b0110: CCIR656 output 3’b1000: Serial RGB888 output without dumy cycle 3’b1100: serial RGB888 output with dumy cycle. Others: Reserved
25	W/R	0x0	DISPLAY OUTPUT BLUE and GREEN SWAP
24:20	R/W	0x00	MCU_RW Signal end point
19:15	R/W	0x00	MCU_RW signal start point
14:10	R/W	0x00	MCU_CS signal end point
9:5	R/W	0x00	MCU_CS signal start point
4:0	R/W	0x00	MCU TIMING total point

RAM_CEN_CTRL

Address: Operational Base + Offset(0x00a4)

RAM CEN Control register

Bit	Attr	Reset Value	Description
31:8	R/W	0x0	Key Color
7	R	0x0	WIN1 AREA3 Color key enable. When Write 0098_BIT27 && 0094_BIT15 = 1, It is enable
6	R	0x0	WIN1 AREA2 Color key enable. When Write 0098_BIT26 && 0094_BIT14 = 1, It is enable
5	R	0x0	WIN1 AREA1 Color key enable. When Write 0098_BIT25 && 0094_BIT13 = 1, It is enable
4	R	0x0	WIN1 AREA0 Color key enable. When Write 0098_BIT24 && 0094_BIT12 = 1, It is enable
3	W/R	0x0	MCU BYPASS REMAP EANBLE
2	W/R	0x0	Deflick_en, the deflick filter enable control, It will act with the HSD_FACTOR/VSD_FACTOR register
1	R/W	0x1	RAM1_BUF1 CEN Control
0	R/W	0x1	RAM1_BUF0 CEN Control

WIN0_YRGB_WPORT

Address: Operational Base + Offset(0x0100)

WIN0 YRGB Data write port

Bit	Attr	Reset Value	Description
31:0	W	---	Win0 RGB Data or Y Data Write Port

WIN0_Cbr_WPORT

Address: Operational Base + Offset(0x0200)
WIN0 Cbr Data write port

Bit	Attr	Reset Value	Description
31:0	W	---	Win0 Cbr Data Write Port

WIN1_AREA0_WPORT
Address: Operational Base + Offset(0x0300)
WIN1 AREA0 Data write port

Bit	Attr	Reset Value	Description
31:0	W	---	Win1 AREA0 Data Write Port

WIN1_AREA1_WPORT
Address: Operational Base + Offset(0x0400)
WIN1 AREA1 Data write port

Bit	Attr	Reset Value	Description
31:0	W	---	Win1 AREA1 Data Write Port

WIN1_AREA2_WPORT
Address: Operational Base + Offset(0x05xx)
WIN1 AREA2 Data write port

Bit	Attr	Reset Value	Description
31:0	W	---	Win1 AREA2 Data Write Port

WIN1_AREA3_WPORT
Address: Operational Base + Offset(0x0600)
WIN1 AREA3 Data write port

Bit	Attr	Reset Value	Description
31:0	W	---	Win1 AREA3 Data Write Port

MCU_BYPASS_WPORT
Address: Operational Base + Offset(0x0700)
MCU BYPASS Data Write Port

Bit	Attr	Reset Value	Description
31:0	W	---	When MCU is in BYPASS Mode, BYPASS Data is Writen by this Port

9.4 Function Description

9.4.1 BUS Operation

The LCD Controller should first Configure the Bus Register, Then Configure the display Register, Disable the Blank Register, Enable the display_done Register, and when the Interrupt occurs, send data to RAM. Win0 and Win1 data is through different port to the RAM, The YRGB and Cb/Cr Data of win0 is through different port also.

Some register such as DSP_WIN0_ST, DSP_WIN0_INFO will be active by writen REG_CFG_DONE.

9.4.2 Bus Operation Configure

If data transfer from AHB Master, AHB slave should configure the AHB Master transfer register and Memory Start Register.

For the win0 Virtual display, the Parameter Setting is shown in Fig.8-2 .

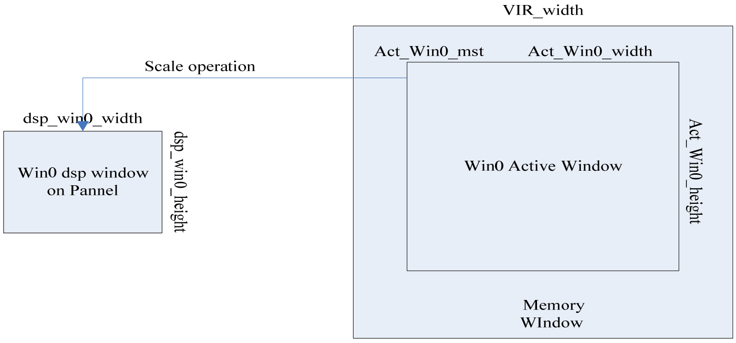


Fig. 9-2 LCDC virtual display Parameter Setting diagram

For Win1 Virtual display, the Parameter Setting is shown in Fig. 8-3.

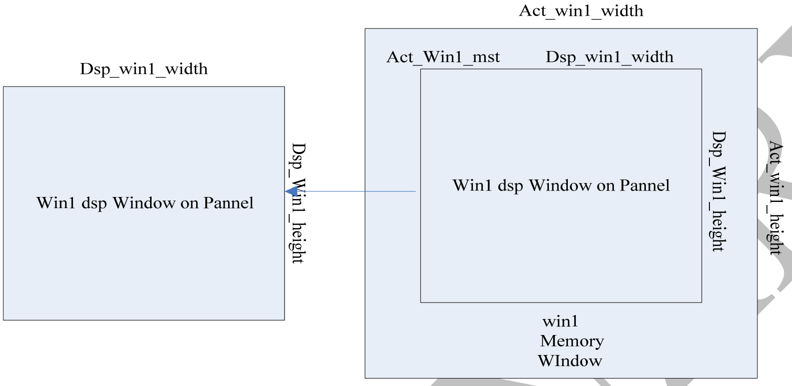


Fig. 9-3 LCDC Win1 Virtual Display Parameter Setting diagram

9.4.3 Display Operation Configure

Win0 and win1 display on the Pannel is shown in Fig. 8-4:

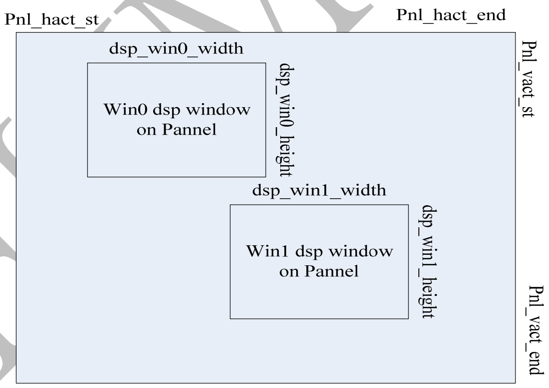


Fig. 9-4 LCDC win0 and win1 display on panel diagram

Win0 and Win1 must be in the Pannel region:

- Pnl_hact_st < dsp_win0_xst < Pnl_hact_end
- Pnl_hact_st < dsp_win0_xst + dsp_win0_width < Pnl_hact_end
- Pnl_vact_st < dsp_win0_yst < Pnl_vact_end
- Pnl_vact_st < dsp_win0_yst + dsp_win0_height < Pnl_vact_end
- Pnl_hact_st < dsp_win1_AREA0~3_xst < Pnl_hact_end
- Pnl_hact_st < dsp_win1_AREA0~3_xst + dsp_win1_AREA0~3_width < Pnl_hact_end
- Pnl_vact_st < dsp_win1_AREA0~3_yst < Pnl_vact_end
- Pnl_vact_st < dsp_win1_AREA0~3_yst + dsp_win1_AREA0~3_height < Pnl_vact_end

9.4.4 DISPLAY Timing

The Horizontal SYNC and DATA Enable Display Timing is shown in Fig. 8-5.
HSYNC is the Horizontal sync signal .
DEN is the DATA Eanble signal
DATA is the DATA signal

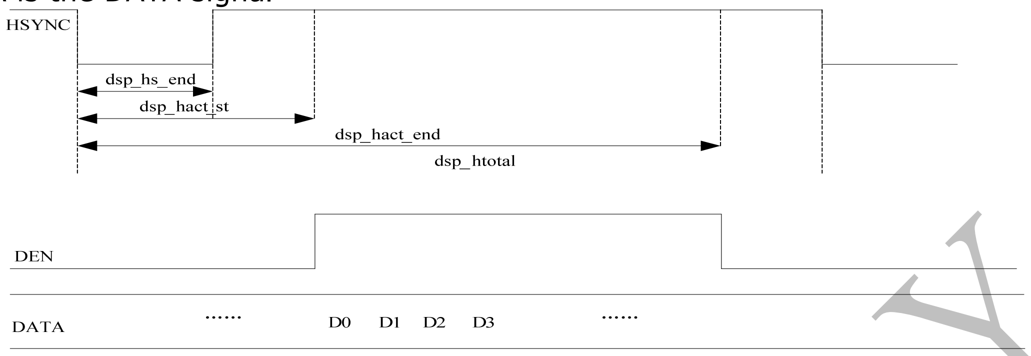


Fig. 9-5 LCDC Horizontal Display Timing waveform

The Vertical Progressive Display Timing is shown in Fig. 8-6 .

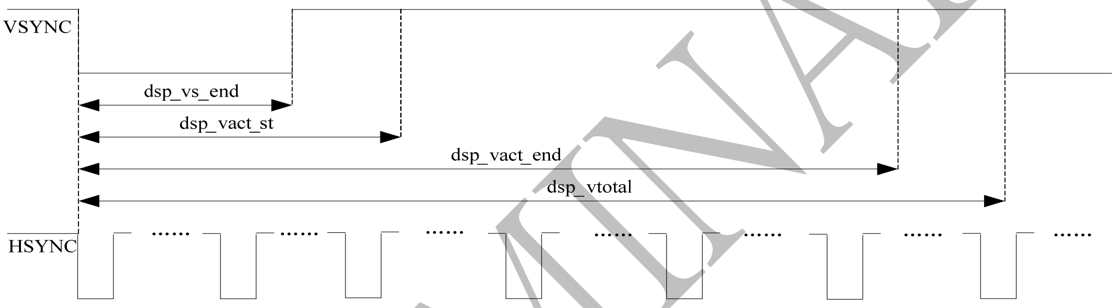


Fig. 9-6 LCDC Vertical Progressive Display Timing Waveform

The second field of Vertical Interlace Display Timing is shown in Fig.8-7 and the first field of vertical Interlace Display Timing is the same as shown in Fig.8-6 .

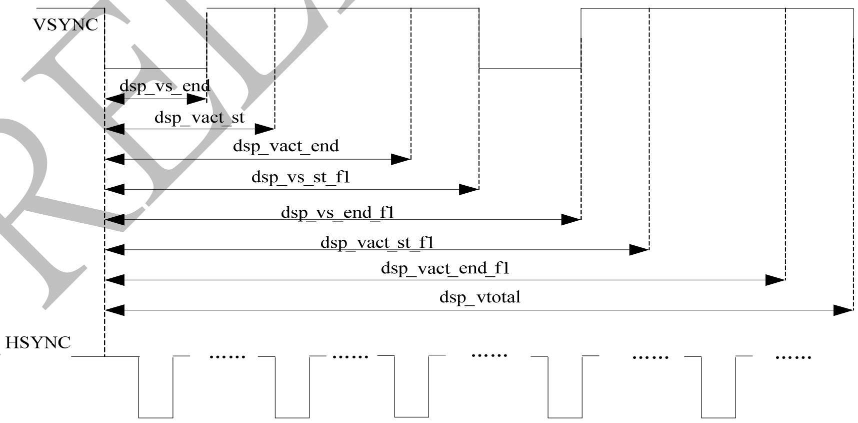


Fig. 9-7 LCDC Vertical Interlace Display Timing Waveform

If the Display Timing of MCU mode is shown blelow

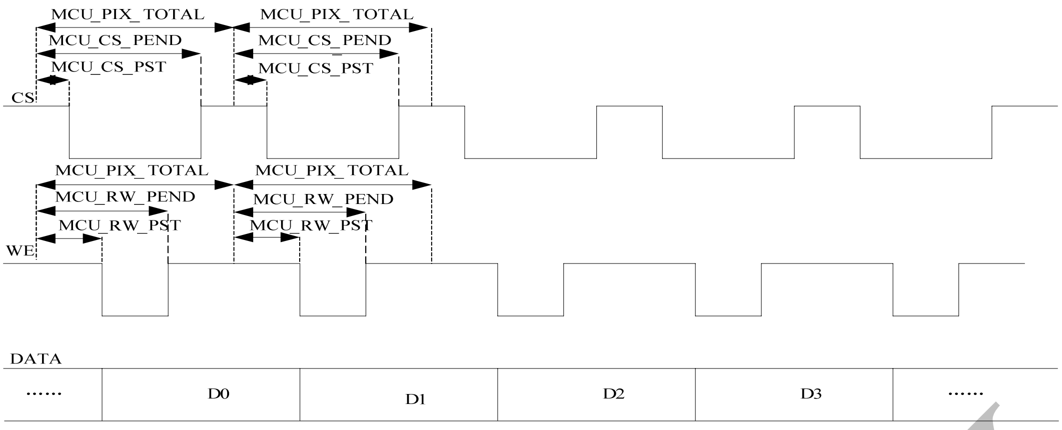


Fig. 9-8 LCDC MCU Mode Display Timing Waveform

When the MCU Panel in the dsp_hold Mode, the timing will not send out after one frame end. When config the dsp_hold Start signal, the timing will send again.

9.4.5 LCDC Input DATA Format

LCD Controller supports six input format, RGB888, RGB565, YUV444, YUV422, YUV4200, YUV4201, YUV420M. The detail information is shown in Fig. 8-9 .

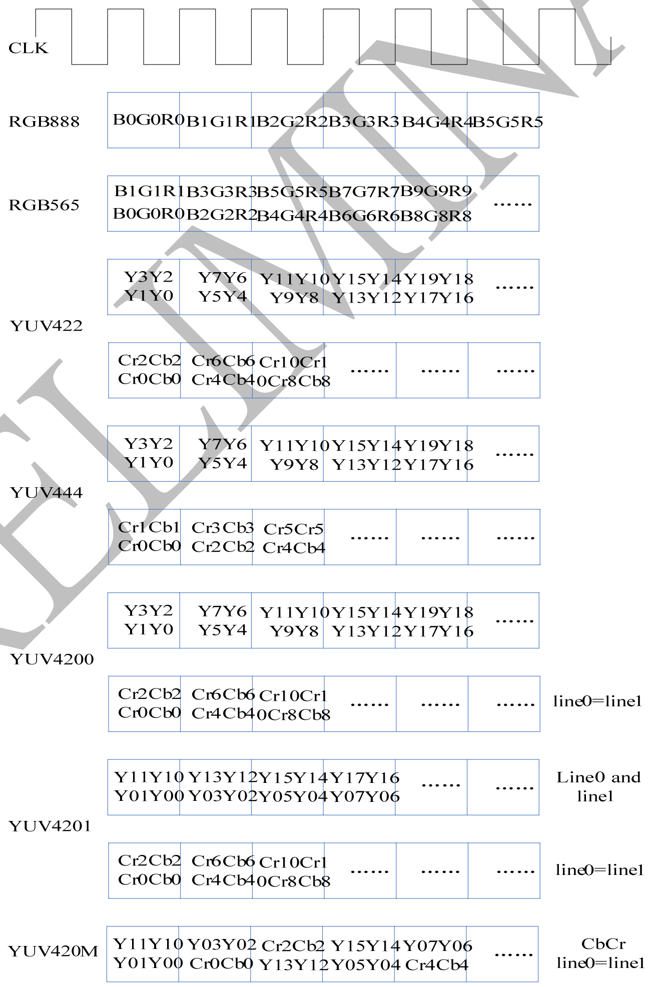

CLK
RGB888
B0G0R0 B1G1R1 B2G2R2 B3G3R3 B4G4R4 B5G5R5
RGB565
B1G1R1 B3G3R3 B5G5R5 B7G7R7 B9G9R9 ……
B0G0R0 B2G2R2 B4G4R4 B6G6R6 B8G8R8
YUV422
Y3Y2 Y1Y0 | Y7Y6 Y5Y4 | Y11Y10 Y9Y8 | Y15Y14 Y13Y12 | Y19Y18 Y17Y16 | ……
Cr2Cb2 Cr0Cb0 | Cr6Cb6 Cr4Cb4 | Cr10Cr1 0Cr8Cb8 | …… | …… | ……
YUV444
Y3Y2 Y1Y0 | Y7Y6 Y5Y4 | Y11Y10 Y9Y8 | Y15Y14 Y13Y12 | Y19Y18 Y17Y16 | ……
Cr1Cb1 Cr0Cb0 | Cr3Cb3 Cr2Cb2 | Cr5Cr5 Cr4Cb4 | …… | …… | ……
YUV4200
Y3Y2 Y1Y0 | Y7Y6 Y5Y4 | Y11Y10 Y9Y8 | Y15Y14 Y13Y12 | Y19Y18 Y17Y16 | ……
Cr2Cb2 Cr0Cb0 | Cr6Cb6 Cr4Cb4 | Cr10Cr1 0Cr8Cb8 | …… | …… | …… line0=line1
YUV4201
Y11Y10 Y01Y00 | Y13Y12 Y03Y02 | Y15Y14 Y05Y04 | Y17Y16 Y07Y06 | …… | …… Line0 and line1
Cr2Cb2 Cr0Cb0 | Cr6Cb6 Cr4Cb4 | Cr10Cr1 0Cr8Cb8 | …… | …… | …… line0=line1
YUV420M
Y11Y10 Y01Y00 | Y03Y02 Cr0Cb0 | Cr2Cb2 Y13Y12 | Y15Y14 Y05Y04 | Y07Y06 Cr4Cb4 | …… CbCr line0=line1

Fig. 9-9 LCDC Input Data Format Diagram

If the input format is RGB888, the 32bit bus is xBGR, R is the lowest byte, and x is the highest byte.

If the input format is RGB565, the R0 is the lowest byte, and the B1 is highest byte.

If the input format is YUV422, the Y3 is the highest byte and the Y0 is the lowest byte in channel Y. The Cr2 is the highest byte and the Cb0 is the lowest byte in channel C.

If the input format is YUV444, the Y3 is highest byte and the Y0 is the lowest byte in channel Y.The Cr1 is the highest byte and the Cb0 is the lowest byte in channel C.

If the input format is YUV4200, Y3 is the highest byte and Y0 is the lowest byte in channel Y. Cr2 is the highest byte and Cb0 is the lowest byte in channel C. the first line Cbr and the second line Cbr is shared in YUV4200 format.

If the input format is YUV4201, Y11 is the highest byte and Y00 is the lowest byte in channel Y. Y11 is Y value of the second pixel of the second line , and Y00 is Y value of the first pixel of the first line. Cr2 is the highest byte and Cb0 is the lowest byte in channel C. the first line Cbr and the second line Cbr is shared in YUV4201 format.

It is the same as YUV4200 .

If the input format is YUV420M, Y and C is send in the same Channel, Y11 is the highest byte and Y00 is the lowest byte. Y11 is Y value of the second pixel of the second line , and Y00 is Y value of the first pixel of the first line. Cr2 is the highest byte and Cb0 is the lowest byte. the first line Cbr and the second line Cbr is shared.

8.4.6 LCDC Output DATA Format

There are seven output format that lcd controller supported. It is shown in Fig. 8-10

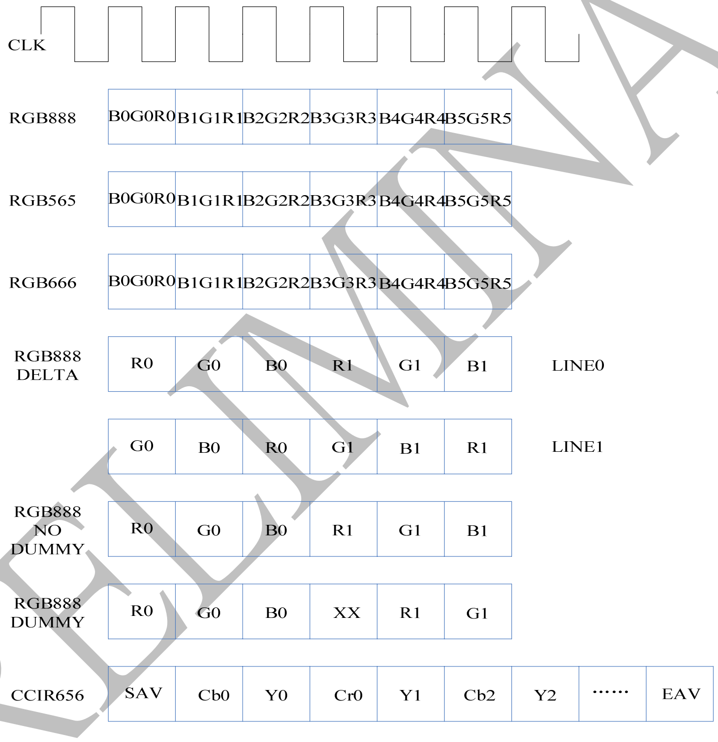

CLK
RGB888
B0G0R0 B1G1R1 B2G2R2 B3G3R3 B4G4R4 B5G5R5
RGB565
B0G0R0 B1G1R1 B2G2R2 B3G3R3 B4G4R4 B5G5R5
RGB666
B0G0R0 B1G1R1 B2G2R2 B3G3R3 B4G4R4 B5G5R5
RGB888 DELTA
R0 G0 B0 R1 G1 B1
LINE0
G0 B0 R0 G1 B1 R1
LINE1
RGB888 NO DUMMY
R0 G0 B0 R1 G1 B1
RGB888 DUMMY
R0 G0 B0 XX R1 G1
CCIR656
SAV Cb0 Y0 Cr0 Y1 Cb2 Y2 …… EAV

Fig. 9-10 LCDC Output Data Format Diagram

- RGB888: it is the 24bit RGB output, B is in the highest byte and R is in the lowest byte.
- RGB565: it is the low 16bit RGB output, B is in the highest bit and R is in the lowest bit
- RGB666: it is the low 18bit RGB output, B is the hightest bit and R is the lowest bit.
- RGB888 DELTA: It is low 8 bit RGB output. the first line is different from second line. R outputs first, G outputs followed and B outputs latter in the first line. G outputs first and B outputs followed and R output latter in second line.
- RGB888 NO DUMMY: it is low 8bit RGB output. R outputs first, G outputs followed and B outputs latter.
- RGB888 DUMMY: it is low 8 bit RGB output. R outputs first, G outputs followed, B outputs latter and XX output latest.
- CCIR656: it is YUV outputs, sav is the valid start of one line the eav is valid end of one line. The valid Cb0 outputs first, Y0 outputs followed, Cr0 outputs latter and Y1

outputs latest.

9.4.7 LCDC Pin Table Description

When use the different Pannel, LCD controler will use the different Pin. The relation of Pannel and Pin is shown in Table 32-2.

Chapter 10 DW_DMA

10.1 Design Overview

10.1.1 Overview

The DW_DMA controller provides an interface to translate data between AHB and AHB. 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 ARMD AHB bus devices and EXP AHB bus data transfer.

10.1.2 Features

- Provide AHB-to-AHB bus protocol translation
- Support AHB-to-AHB DMA or Single AHB DMA
- Three 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
- Channel 0 has 64 bytes FIFO, channel 1 has 32 bytes FIFO, Channel 2 has 16 bytes FIFO.
- Channel 2 do not support Scatter/Gather and LLP transfter, and it is Maximum block size in source transfer widths is 2047.

10.2 Architecture

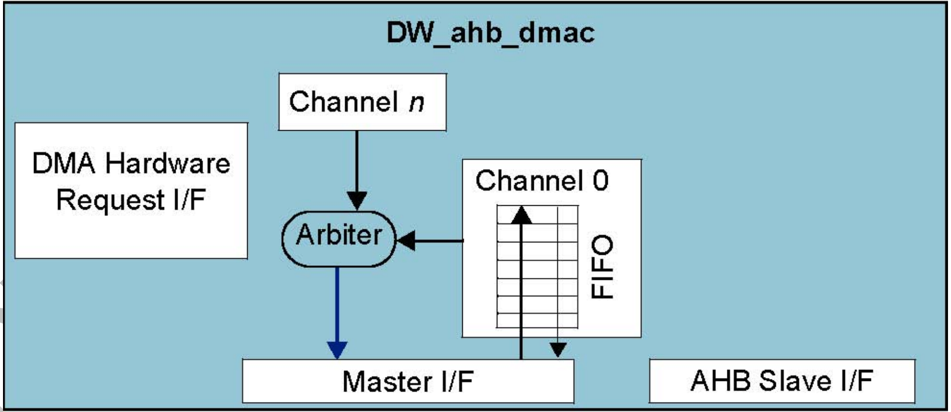


Fig. 10-1 DW_DMA Architecture

10.3 Registers

10.3.1 Registers Summary

Name	Address Offset	Width	R/W	Description
Channel Registers, x = channels number , (0 ~2)				
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
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
Miscellaneous Registers				

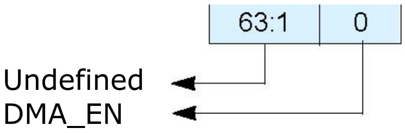
DmaCfgReg	0x398	64	R/W	DMA Configuration Register Reset Value: 0x0
ChEnReg	0x3a0	64	R/W	DMA Channel Enable Register Reset Value: 0x0

10.3.2 Configuration and Channel Enable Registers

DmaCfgReg

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

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



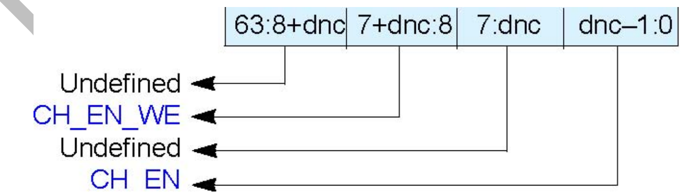
Bits	Name	R/W	Reset	Description
63:1	Undefined	N/A	0x0	Reserved
0	DMA_EN	R/W	0x0	**DW_DMA Enable bit.** 0 = DW_DMA Disabled 1 = DW_DMA Enabled.

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

This is the DW_DMA 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_DMA 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.

10.3.3 Channel Registers

The channel registers consist of the following, where x = 0 to 2:

- **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
- **LLPx** – Linked list pointer register for channel x
- **SARx** – Source address register for channel x
- **SGRx** – Source gather 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.

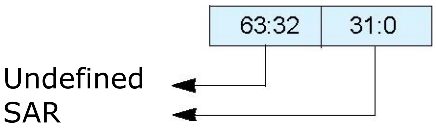
SARx

- Name: Source Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:
  SAR0 – 0x000
  SAR1 – 0x058
  SAR2 – 0x0b0
- 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.

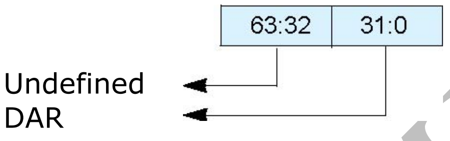
DARx

- Name: Destination Address Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:
  DAR0 – 0x008
  DAR1 – 0x060
  DAR2 – 0x0b8
- 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_DMA 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_DMA 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:

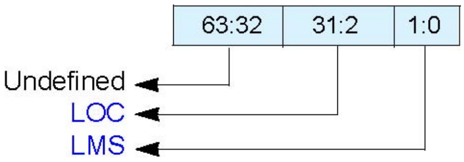
- When on the DAR address, OR
- Contiguous multi-block transfers on destination side, OR
- DST_TR_WIDTH > SRC_TR_WIDTH, OR
- (BLOCK_TS * SRC_TR_WIDTH)/DST_TR_WIDTH != integer (where SRC_TR_WIDTH, DST_TR_WIDTH is byte width of transfer)

LLPx

- Name: Linked List Pointer Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:
  LLP0 – 0x010
  LLP1 – 0x068
  LLP2 – 0x0c0
- 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

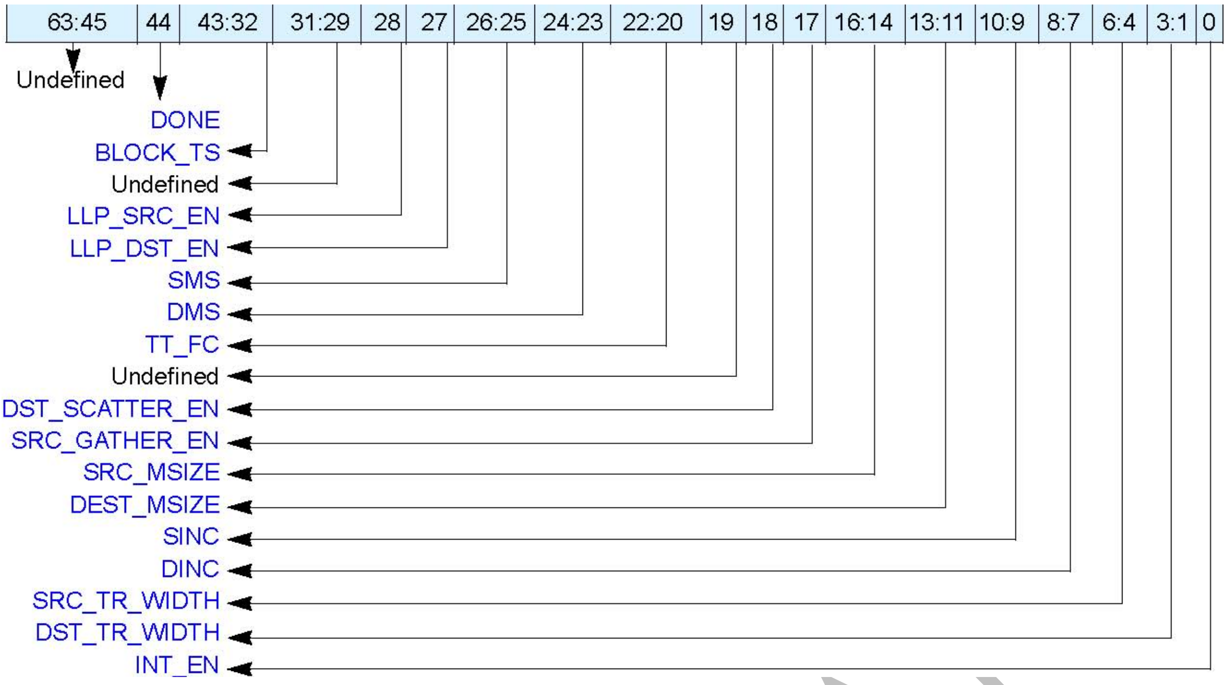
- Name: Control Register for Channel x
- Size: 64 bits
- Address Offset: for x = 0 to 2:
  CTL0 – 0x018
  CTL1 – 0x070
  CTL2 – 0x0c8
- Read/Write Access: Read/Write

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_DMA 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_DMA, 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 10-1 DW_DMA 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 10-2 DW_DMA 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 10-3 DW_DMA CTLx.TT_FC field Decoding

CTLx.TT_FC Field	Transfer Type	Flow Controller
000	Memory to Memory	DW_DMA
001	Memory to Peripheral	DW_DMA
010	Peripheral to Memory	DW_DMA
011	Peripheral to Peripheral	DW_DMA

100	Peripheral to Memory	Peripheral
101	Peripheral to Peripheral	Source Peripheral
110	Memory to Peripheral	Peripheral
111	Peripheral to Peripheral	Destination Peripheral

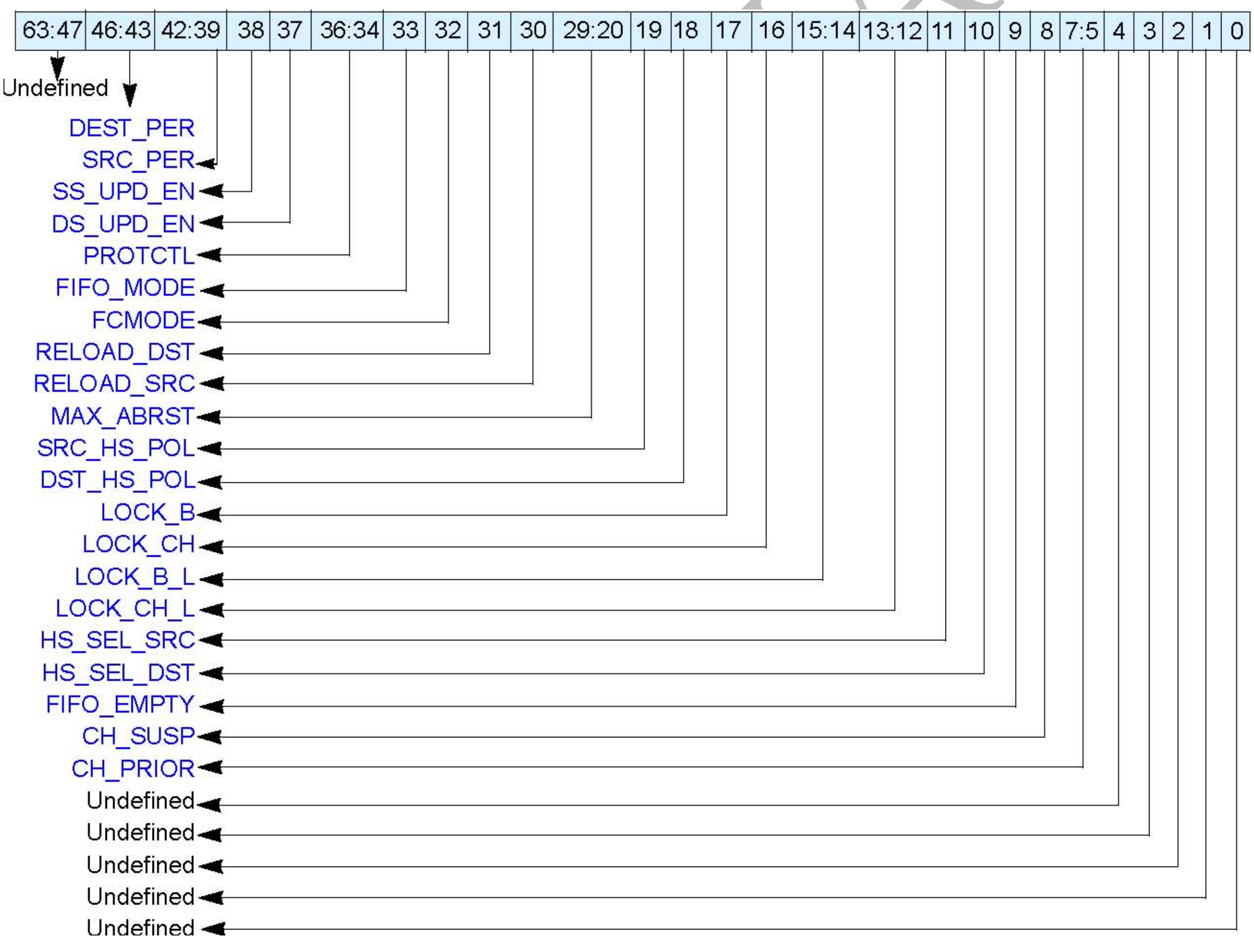
CFGx

- Name: Configuration Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 2:
  CFG0 – 0x040
  CFG1 – 0x098
  CFG2 – 0x0f0
- 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_DMA 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 10-4 DW_DMA PROTCTL field to HPROT Mapping

1‘b1	HPROT[0]
CFGx.PROTCTL[1]	HPROT[1]
CFGx.PROTCTL[2] ->	HPROT[2]
CFGx.PROTCTL[3] ->	HPROT[3]

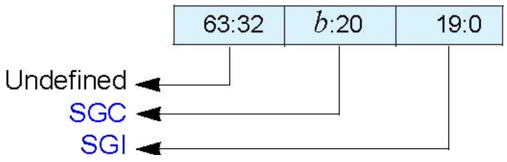
SGRx

- Name: Source Gather Register for Channel x
- Size: 64 bits (upper 32 bits are reserved)
- Address Offset: for x = 0 to 1:
  SGR0 – 0x048
  SGR1 – 0x0a0
- 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 1:
  DSR0 – 0x050
  DSR1 – 0x0a8
- 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.

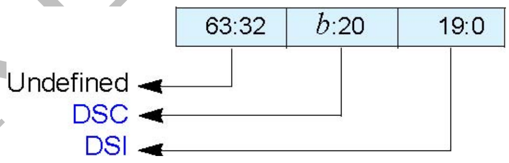


Table 10-5 DW_DMA Destination Scatter Register Description for Channel x

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.

10.3.4 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

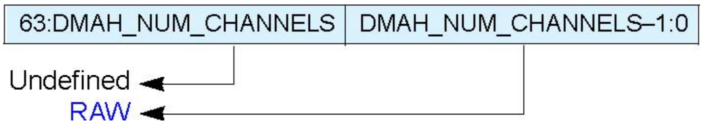
- Name: Interrupt Raw Status Registers
- Size: 64 bits
- Address Offset:
  RawTfr – 0x2c0
  RawBlock – 0x2c8
  RawSrcTran – 0x2d0

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.

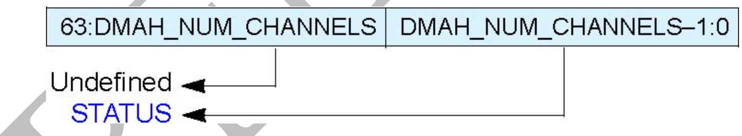


Bits	Name	R/W	Reset	Description
63:4	Undefined	N/A	0x0	Reserved
3:0	RAW	R	0x0	Raw interrupt status.

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_DMA.



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

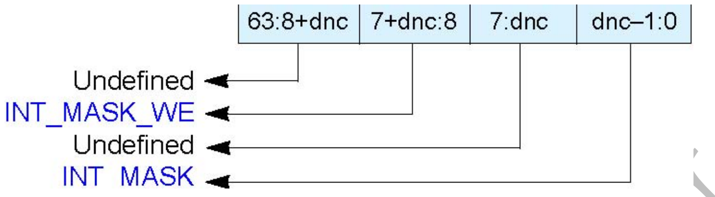
- Name: Interrupt Mask Registers
- Size: 64 bits
- Address Offset:
  MaskTfr – 0x310
  MaskBlock – 0x318
  MaskSrcTran – 0x320
  MaskDstTran – 0x328
  MaskErr – 0x330
- Read/Write Access: Read/Write

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_DMA to set the appropriate bit in the Status registers and int_* port signals.

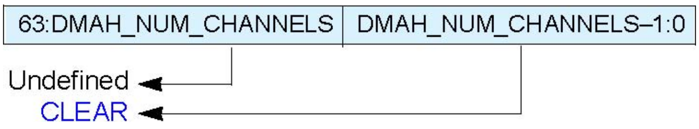


Bits	Name	R/W	Description
63:12	Undefined	N/A	**Reset Value:** 0x0
11:8	INT_MASK_WE	W	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled **Reset Value:** 0x0
7:4	Undefined	N/A	Reset Value: 0x0
3:0	INT_MASK	R/W	Interrupt Mask 0 = masked 1 = unmasked **Reset Value:** 0x0

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

.

10.4 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_dma 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_DMA.

10.5 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.

10.6 DW_DMA 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_DMA 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_DMA 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_DMA 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_DMA 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.

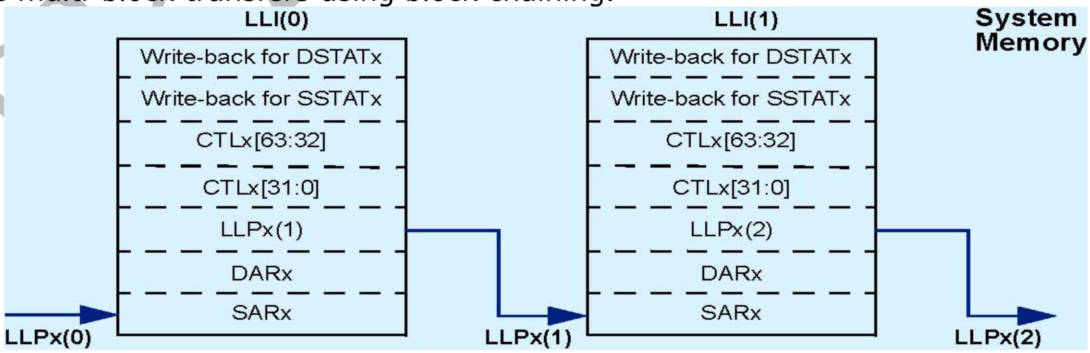
In this case, the DW_DMA reprograms the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

DW_DMA 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.

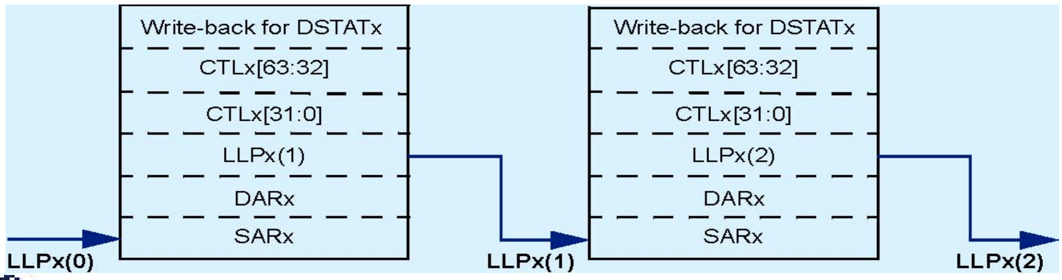
The SARx, DARx, LLPx, and CTLx registers are fetched from system memory on an LLI update. If configuration parameter DMAH_CHx_CTL_WB_EN = True, then the updated contents of the CTLx, SSTATx, and DSTATx registers are written back to memory on block completion. Figure 2 and Figure 3 show how you use chained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



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



Note

In order to not confuse the SARx, DARx, LLPx, CTLx, STATx, and DSTATx register locations of the LLI with the corresponding DW_DMA 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

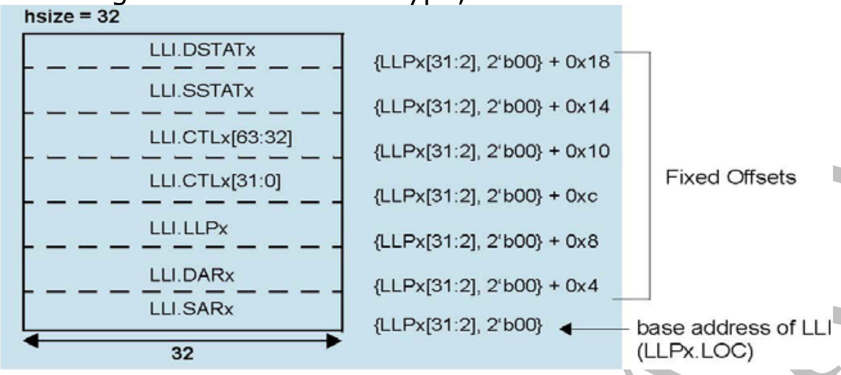
For rows 6 through 10 of Table 5, the LLI.CTLx, LLI.LLPx, LLI.SARx, and LLI.DARx register locations of the LLI are always affected at the start of every block transfer. The LLI.LLPx and LLI.CTLx locations are always used to reprogram the DW_DMA LLPx and CTLx registers. However, depending on the Table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

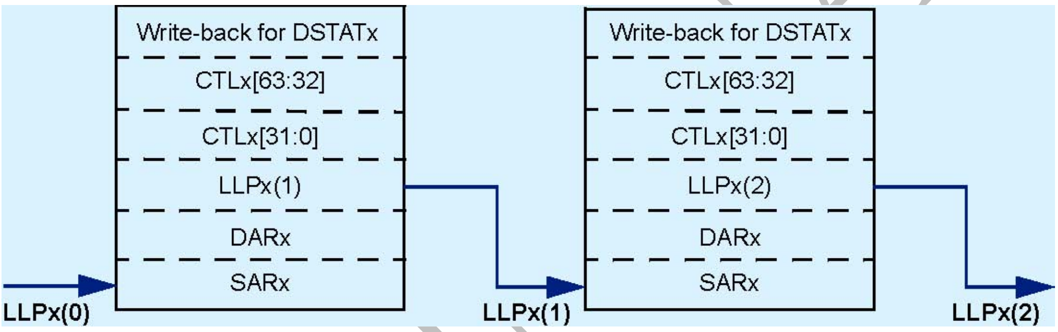
Table 10-6 Programming of Transfer Types and Channel Register Update Method

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 multi-block transfer with contiguous SAR	Yes	0	0	0	1	CTLx, LLPx are reloaded from initial values.	Con-tiguous	Auto-Reload	No
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	No	1	0	0	1	CTLx, LLPx loaded from	Linked List	Auto-Reload	Yes

transfer with auto-reload DAR						next Linked List item.			
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”.


hsize = 32
LLI.DSTATx
LLI.SSTATx
LLI.CTLx[63:32]
LLI.CTLx[31:0]
LLI.LLPx
LLI.DARx
LLI.SARx
32
{LLPx[31:2], 2'b00} + 0x18
{LLPx[31:2], 2'b00} + 0x14
{LLPx[31:2], 2'b00} + 0x10
{LLPx[31:2], 2'b00} + 0xc
{LLPx[31:2], 2'b00} + 0x8
{LLPx[31:2], 2'b00} + 0x4
{LLPx[31:2], 2'b00}
Fixed Offsets
base address of LLI
(LLPx.LOC)



Chapter 11 XDMA

11.1 Overview

The Direct Memory Access (XDMA) is part of the DSP platform. The XDMA transfers data from a source to a destination without any Core intervention. This transfer is carried out in either an untouched data format or in a restructured data format, depending on the requirements of the particular application.

The fully integrated XDMA enables the Core or an external device to first define and initiate data-transfer processes, and then enable the Core or external device to continue its operations while the XDMA executes data transfers in parallel.

The XDMA contains up-to 16 independent programmable channels that support 16 different contexts (data transfers) for the XDMA operation.

11.1.1 Features

The XDMA has the following main features:

- 16 configurable XDMA channels
- 3D XDMA transfer
- Three AHB-Lite master interfaces, one of which is for the XDMA manager
- AHB-Lite slave interface for registers configuration
- Access to the entire DSP internal data memory
- 8/16/32/64-bit data transfer support
- Configurable burst length
- Two levels of XDMA Channels
  - Full version - including the entire features channels 0-3
  - Basic version - includes basic features only channels 4-15
- 32-bit address space
- Programmable channel priority
- Programmable source and destination addresses with a post-modification option
- Configurable external channel triggering (edge or level)
- Interrupt generation
- pause and resume operations
- Chaining-channels operating mode
- Linked list-transfer operating mode
- Auto-channel initialization mode
- XDMA manager support
- Breakpoint generation for emulation support
- Halt on breakpoint
- Eight-stage memory buffer FIFO
- Data pack and un-pack.
- Power save modes

XDMS only used in DSP sub system normally. For detailed information about XDMS controller, please refer to **RK28xx DSP sub-system.pdf.**

Chapter 12 Interrupt Controller (INTC)

12.1 Design Overview

12.1.1 Overview

The INTC is a configurable, vectored interrupt controller for AMBA-based systems. It is an AMBA 2.0-compliant Advanced High-speed Bus (AHB) slave device.

12.1.2 Features

- 40 IRQ normal interrupt sources
- 2 FIQ fast interrupt sources
- Software interrupts
- Priority filtering
- Masking
- Scan mode
- Programmable interrupt priorities
- Configuration ID registers
- Encoded parameters

12.2 Architecture

This section describes the functional operation of AHB Interrupt Controller.

12.2.1 Block Diagram

The INTC comprises with:

- Slave I/F – AHB bus interface
- IRQ_Generation – IRQ generation module
- FIQ_Generation – FIQ generation module
- Mask – Interrupt Mask module

The diagram is shown as followed

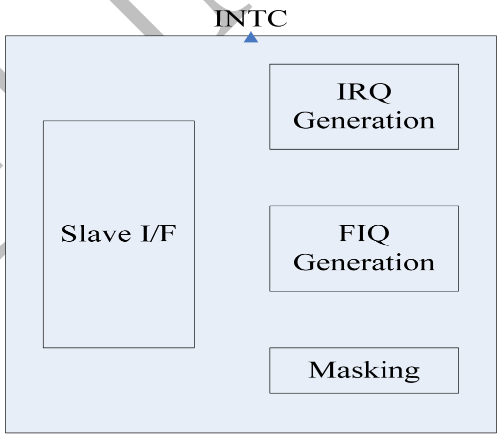


Fig. 12-1 Interrupt Controller Architecture

12.3 Registers

This section describes the control/status registers of the design

12.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
IRQ_INTEN_L	0x00	W	0x0	IRQ interrupt source enable register (low)
IRQ_INTEN_H	0x04	W	0x0	IRQ interrupt source enable

				register (high).
IRQ_INTMASK_L	0x08	W	0x0	IRQ interrupt source mask register (low).
IRQ_INTMASK_H	0x0C	W	0x0	IRQ interrupt source mask register (high).
IRQ_INTFORCE_L	0x10	W	0x0	IRQ interrupt force register (low).
IRQ_INTFORCE_H	0x14	W	0x0	IRQ interrupt force register (high).
IRQ_RAWSTATUS_L	0x18	W	0x0	IRQ raw status register (low).
IRQ_RAWSTATUS_H	0x1c	W	0x0	IRQ raw status register (high)
IRQ_STATUS_L	0x20	W	0x0	IRQ status register (low)
IRQ_STATUS_H	0x24	W	0x0	IRQ status register (high)
IRQ_MASKSTATUS_L	0x28	W	0x0	IRQ interrupt mask status register (low)
IRQ_MASKSTATUS_H	0x2c	W	0x0	IRQ interrupt mask status register (high)
IRQ_FINALSTATUS_L	0x30	W	0x0	IRQ interrupt final status (low)
IRQ_FINALSTATUS_H	0x34	W	0x0	IRQ interrupt final status (high)
FIQ_INTEN	0xc0	W	0x0	Fast interrupt enable register
FIQ_INTMASK	0xc4	W	0x0	Fast interrupt mask register
FIQ_INTFORCE	0xc8	W	0x0	Fast interrupt force register
FIQ_RAWSTATUS	0xcc	W	0x0	Fast interrupt source raw status register
FIQ_STATUS	0xd0	W	0x0	Fast interrupt status register
FIQ_FINALSTATUS	0xd4	W	0x0	Fast interrupt final status register
IRQ_PLEVEL	0xd8	W	0x0	IRQ System Priority Level Register
IRQ_PN_OFFSET	0xe8 + N*4	W	N	Interrupt N priority level register(s), where N is from 0 to 15
IRQ_PN_OFFSET	0xe8 + N*4	W	N-16	Interrupt N priority level register(s), where N is from 16 to 31
IRQ_PN_OFFSET	0xe8 + N*4	W	N-32	Interrupt N priority level register(s), where N is from 32 to 39
AHB_ICTL_COMP_VERSION	0x3f8	W	0x3230342a	Version register
ICTL_COMP_TYPE	0x3fc	W	0x44571120	Component Type 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

IRQ_INTEN_L
Address: Operational Base + offset(0x00)
Interrupt Source Enable (Low) Register

bit	Attr	Reset Value	Description

31:0	RW	0x0	Interrupt enable bits for lower 32 interrupt sources. A 1 in any bit position enables the corresponding interrupt. 0: disable interrupt 1: enable interrupt

IRQ_INTEN_H
Address: Operational Base + offset(0x04)
Interrupt Source Enable (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Interrupt enable bit for upper 8 interrupt sources. A 1 in any bit position enables the corresponding interrupt. 0: disable interrupt 1: enable interrupt

IRQ_INTMASK_L
Address: Operational Base + offset(0x08)
Interrupt Source Mask (Low) Register

bit	Attr	Reset Value	Description
31:0	RW	0x0	Interrupt mask bits for the lower 32 interrupt sources. A 1 in any bit position masks (disables) the corresponding interrupt. By default, all bits are unmasked. 0: unmask interrupt 1: mask interrupt

IRQ_INTMASK_H
Address: Operational Base + offset(0x0c)
Interrupt Source Mask (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Interrupt mask bits for the upper 8 interrupt sources. 0: unmask interrupt 1: mask interrupt

IRQ_INTFORCE_L
Address: Operational Base + offset(0x10)
Interrupt Force (Low) Register

bit	Attr	Reset Value	Description
31:0	RW	0x0	Interrupt force bits for the lower 32 interrupt sources. Each bit in this register corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register correspond to the polarity of the associated irq_intsrc input. If the interrupt input is configured to be active high, the corresponding bit in the register is also active high. 0: active low 1: active high

IRQ_INTFORCE_H
Address: Operational Base + offset(0x14)
Interrupt Force (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	RW	0x0	Interrupt force bits for the upper 8 interrupt sources.

			Each bit in this register corresponds to one bit of the irq_intsrc input.The polarity of the bits in the register correspond to the polarity of the associated irq_intsrc input. If the interrupt input is configured to be active high, the corresponding bit in the register is also active high. The reset state of the force bits is always inactive. 0: active low 1: active high

IRQ_RAWSTATUS_L

Address: Operational Base + offset(0x18)

Interrupt Raw Status (Low) Register

bit	Attr	Reset Value	Description
31:0	R	0x0	Actual interrupt source.

IRQ_RAWSTATUS_H

Address: Operational Base + offset(0x1c)

Interrupt Raw Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Actual interrupt source. These are the upper 8 interrupt sources.

IRQ_STATUS_L

Address: Operational Base + offset(0x20)

Interrupt Status (Low) Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Interrupt status after the forcing and interrupt enabling stage. These are the interrupt status signals for the lower 32 interrupt sources.

IRQ_STATUS_H

Address: Operational Base + offset(0x24)

Interrupt Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Interrupt status after the forcing and interrupt enabling stage. These are the interrupt status signals for the upper 8 interrupt sources.

IRQ_MASKSTATUS_L

Address: Operational Base + offset(0x28)

Interrupt Mask Status (Low) Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Interrupt status after the masking stage. These are the interrupt status signals for the lower 32 interrupt sources.

IRQ_MASKSTATUS_H

Address: Operational Base + offset(0x2c)

Interrupt Mask Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Interrupt status after the masking stage. These are

			the interrupt status signals for the upper 8 interrupt sources.

IRQ_FINALSTATUS_L
Address: Operational Base + offset(0x30)
Interrupt Final Status (Low) Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Interrupt status after the priority level filtering stage. These are the interrupt status signals for the lower 32 interrupt sources.

IRQ_FINALSTATUS_H
Address: Operational Base + offset(0x34)
Interrupt Final Status (High) Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved.
7:0	R	0x0	Interrupt status after the priority level filtering stage. These are the interrupt status signals for the upper 8 interrupt sources.

FIQ_INTEN
Address: Operational Base + offset(0xc0)
Fast Interrupt Enable Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Fast interrupt enable bits. A 1 in any bit position enables the corresponding interrupt. 0: disable interrupt 1: enable interrupt

FIQ_INTMASK
Address: Operational Base + offset(0xc4)
Fast Interrupt Mask Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Fast interrupt mask bits. A 1 in any bit position masks thecorresponding interrupt. This register does not exist when ICT_HAS_FIQ = 0. 0: unmask interrupt 1: mask interrupt

FIQ_INTFORCE
Address: Operational Base + offset(0xc8)
Fast Interrupt Force Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	RW	0x0	Fast interrupt force bits. Each bit in this register corresponds to one bit of the irq_intsrc input. The polarity of the bits in the register correspond to the polarity of the associated fiq_intsrc input. If the interrupt input is configured to be active high, the corresponding bit in the register is also active high. 0: active low 1: active high

FIQ_RAWSTATUS

Address: Operational Base + offset(0xcc)
Fast Interrupt Source Raw Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	Fast interrupt source raw input status..

FIRQ_STATUS
Address: Operational Base + offset(0xd0)
Fast Interrupt Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	Fast interrupt status after the forcing and interrupt enabling stage. 1: active 0: inactive

FIQ_FINALSTATUS
Address: Operational Base + offset(0xd4)
Fast Interrupt Final Status Register

bit	Attr	Reset Value	Description
31:2	-	-	Reserved.
1:0	R	0x0	Fast interrupt status after the masking stage. 1: active 0: inactive

IRQ_PLEVEL
Address: Operational Base + offset(0xd8)
IRQ System Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	0x0	Interrupt controller system priority level for normal interrupt sources. The default state can be configured so that after reset the interrupt controller will accept only interrupts that are enabled and have a priority the same or greater than the system level priority setting.

IRQ_PN_OFFSET
Address: Operational Base + offset(0xe8 + 4 * n)
IRQ Individual Interrupt Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	n	Individual interrupt priority level. The range of N or n (number of registers) is from 0 to 15. A register's value must be an integer from 0x0 to 0xf.

IRQ_PN_OFFSET
Address: Operational Base + offset(0xe8 + 4 * n)
IRQ Individual Interrupt Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	n-16	Individual interrupt priority level. The range of N or n (number of registers) is from 16 to 31. A register's value must be an integer from 0x0 to 0xf.

irq_pN_offset
Address: Operational Base + offset(0xe8 + 4 * n)

IRQ Individual Interrupt Priority Level Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved.
3:0	RW	n-32	Individual interrupt priority level. The range of N or n (number of registers) is from 32 to 39. A register's value must be an integer from 0x0 to 0xf.

AHB_ICTL_COMP_VERSION
Address: Operational Base + offset(0x3f8)
Component Version Register

bit	Attr	Reset Value	Description
31:0	R	0x3230342a	Specific values for this register are described in the Releases Table

ICTL_COMP_TYPE
Address: Operational Base + offset(0x3fc)
Component Type Register

bit	Attr	Reset Value	Description
31:0	R	0x44571120	Type number = 0x44_57_11_20

12.4 Functional Description

12.4.1 Overview

The INTC supports from two to 40 normal interrupt (IRQ) sources that are processed to produce a single IRQ interrupt to the processor. It supports from one to 2 fast interrupt (FIQ) sources that are processed to produce a single FIQ interrupt to the processor. All interrupt processing is combinational so that interrupts are propagated if the bus interface of the INTC is powered down. This means that reading any of the interrupt status registers (raw, status, or final_status) is simply returning the status of the combinational logic, since there are no flip-flops associated with these registers. It is the user's responsibility to ensure that the interrupts stay asserted until they are serviced

12.4.2 Detail Description

IRQ Interrupt Processing

The INTC processes these interrupt sources to produce a single IRQ interrupt to the processor; irq or irq_n. The processing of theinterrupt sources is shown as followed

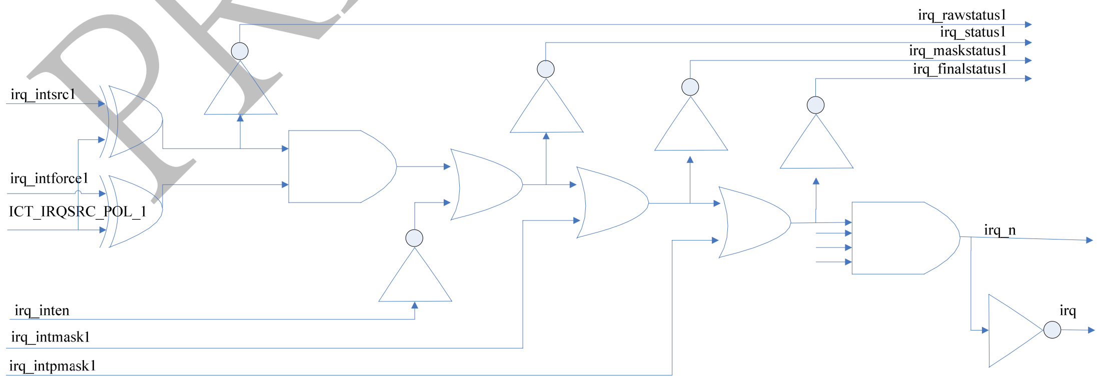


Fig. 12-2 IRQ Interrupt Processing for INTC

IRQ Software-Programmable Interrupts

The INTC supports forcing interrupts from software. To force an interrupt to be active, write to the corresponding bit in the irq_intforce registers (IRQ_INTFORCE_L or IRQ_INTFORCE_H).

IRQ Enable and Masking

To enable each interrupt source independently, write a 1 to the corresponding bit of the irq_inten registers (IRQ_INTEN_L or IRQ_INTEN_H). .

To mask each interrupt source independently, write a 1 to the corresponding bit of the interrupt mask register (IRQ_MASKSTATUS_L/IRQ_MASKSTATUS_H). The reset value for each mask bit is 0 (unmasked)

IRQ Software-Programmable Priority Levels

The INTC supports optional software programmable priority levels. To change the priority level of an interrupt, you write the priority value to the corresponding priority level register in the memory map. There is a priority register for each of the interrupt sources, which can be programmed to one of 16 values from 0x0 to 0xf. Priority registers only exist for available interrupt sources.

IRQ Priority Filter

The INTC supports optional priority filtering. The function of the priority filtering logic is described as follows:

- Each interrupt source is configured to one of 16 priority levels. where 0 is the lowest priority.
- A system priority level can be programmed into the irq_plevel register, which holds values from 0 to 15.
- The INTC filters out any interrupt source with a configured priority level less than the priority currently programmed in the irq_plevel register

IRQ Interrupt Status Registers

The INTC includes up to four status registers used for querying the current status of any interrupt at various stages of the processing. All of the following status registers have the same polarity; a 1 indicates that an interrupt is active, a 0 indicates it is inactive:

- irq_rawstatus

The irq_rawstatus register (irq_rawstatus_l/irq_rawstatus_h) contains the state of the interrupt sources after being adjusted for input polarity. Each bit of this register is set to 1 if the corresponding interrupt source bit is active and is set to 0 if it is inactive.

- irq_status

The irq_status register (irq_status_l/irq_status_h) contains the state of all interrupts after the enabling stage, meaning that an active-high bit indicates that particular interrupt source is active and enabled.

- irq_maskstatus

The irq_maskstatus register (irq_maskstatus_l/irq_maskstatus_h) contains the state of all interrupts after the masking stage, meaning that an active-high bit indicates that particular interrupt source is active, enabled, and not masked.

- irq_finalstatus

This register (irq_finalstatus_l/irq_finalstatus_h) contains the state of all interrupts after the priority filtering stage, meaning an active-high bit indicates that particular interrupt source is active, enabled, not masked, and its configured priority level is greater or equal to the value programmed in the irq_plevel register. If priority filtering has not been selected, this register will contain the same value as the irq_maskstatus register (the final stage of processing).

FIQ Interrupt Processing

FIQ interrupt processing is similar to IRQ interrupt processing, except that priority filtering and interrupt vectors are not supported for the FIQ interrupts. This section describes how the INTC handles the FIQ interrupt processing. the processing of the interrupt sources is described as followed.

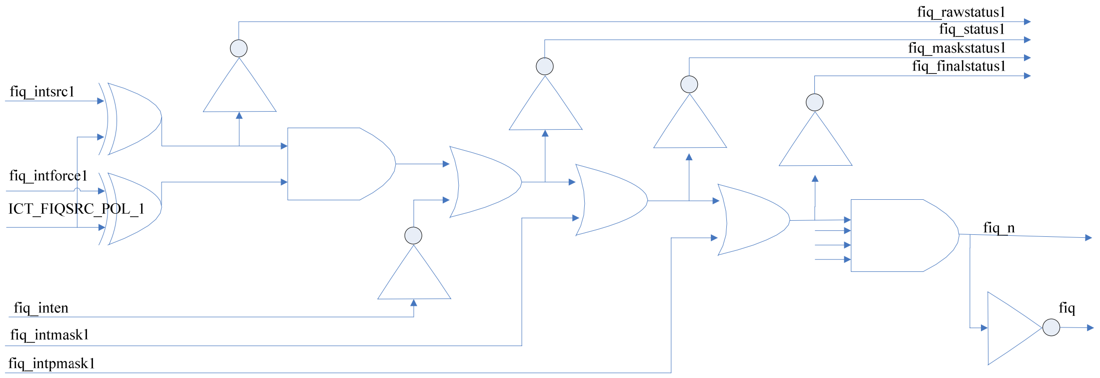


Fig. 12-3 FIQ Interrupt Processing for INTC

FIQ Software-Programmable Interrupts

The INTC supports forcing interrupts from software. You force an interrupt to be active by writing to the corresponding bit in the FIQ_INTFORCE register. The polarity of each bit in this register is the same as the polarity of the corresponding interrupt source signal.

FIQ Enable and Masking

You can enable each interrupt source independently by writing a 1 to the corresponding bit of the FIQ_INTEN register.

You can mask each interrupt source independently by writing a 1 to the corresponding bit of the FIQ_INTMASK register. The reset value for each mask bit is 0; that is, unmasked.

FIQ Interrupt Status Registers

The INTC includes three status registers that you can use to query the current status of any FIQ interrupt at various stages of the processing:

- fiq_rawstatus

  The fiq_rawstatus register contains the state of the interrupt sources after being adjusted for input polarity. Each bit of this register is set to 1 if the corresponding interrupt source bit is active and is set to 0 if it is inactive.

- fiq_status

  The fiq_status register contains the state of all interrupts after the enabling stage, meaning that an active-high bit indicates that particular interrupt source is active and enabled.

- fiq_finalstatus

  The fiq_finalstatus register contains the state of all interrupts after the masking, meaning that an active-high bit indicates that particular interrupt source is active, enabled, and unmasked.

Chapter 13 High-Speed ADC Interface

13.1 Design Overview

13.1.1 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.

13.1.2 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 two channel 8-bit/10-bit data input
- 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 2-bit GPS data input
- Support MPEG transport stream data input
- 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.

For detailed information about HSADC controller, please refer to **RK28xx DSP sub-system.pdf**。

Chapter 14 Host Inteface (HIF)

14.1 Design Overview

14.1.1 Overview

Host Interface (HIF) will focus on high-speed data transfer between RK28xx and Modem chip. There is a 2KB size dual-port SRAM buffer, which can be used to complete data exchange by interactive interrupt for each other.

Another, HIF function can be disabled by software and buffer will become share memory between CPU and DSP.

14.1.2 Features

- 8bits / 16 bits parallel bus for data transfer, it is programmable
- Configurable MCU interface signal valid polarity
- 2KB internal Dual Port SRAM buffer
- Interrupt request for data exchange
- Support HIF function disable
- Two AHB slave interface for memory share of two processors
- MCU interface to communicate between RK28xx and Modem chip
- Support address self-increment for burst transfer when accessing buffer by MCU interface
- Support LCDC interface bypass from HIF interface

14.2 Architecture

14.2.1 Block Diagram

The following diagram illustrates the block diagram for HIF module.

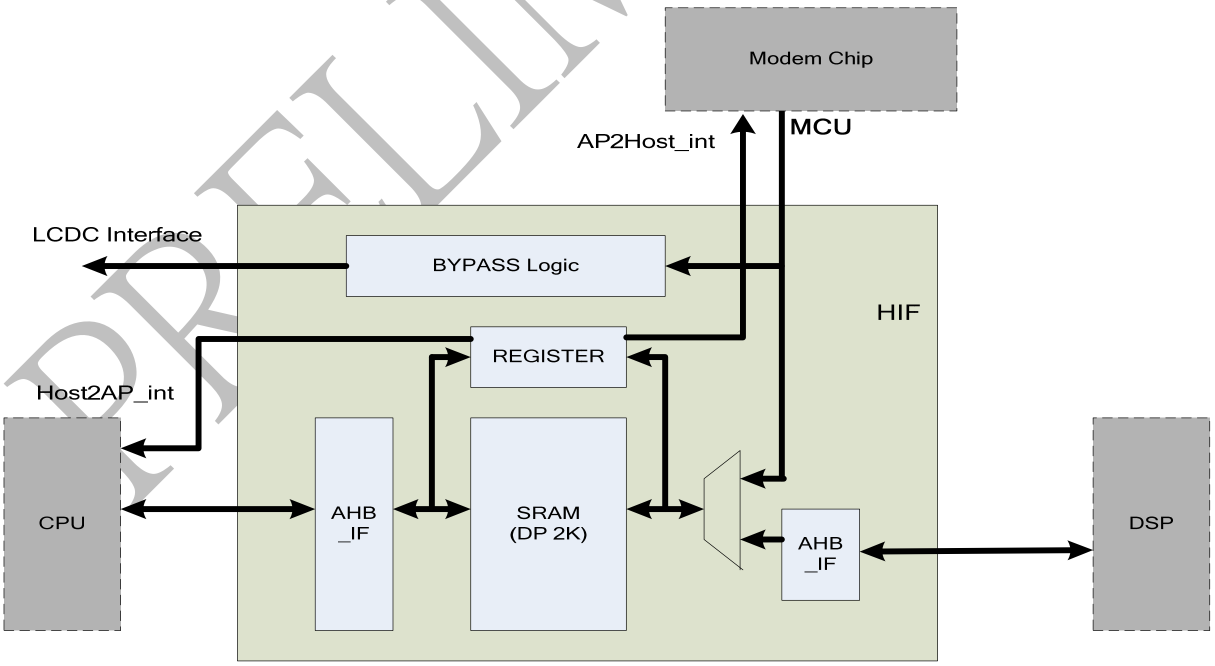


Fig. 14-1 HIF block diagrams

14.3 Registers

This section describes the registers of the HIF.

14.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
HIF_CON	0x00	b	0x0	HIF Control register
HIF_INITCOUNT	0x04	hw	0x0	HIF Initial Transfer Count register
HIF_INITADDR	0x08	hw	0x0	HIF Initial Address register
HIF_ADDR	0x0c	hw	0x0	HIF real Address register,only read by AP
HIF_COUNT	0x10	hw	0x0	HIF real Transfer Count register,only read by AP

14.3.2 Detail Register Description

HIF_CON
Address: Base Addr+0x00
Hif control register

bit	Attr	Reset Value	Description
7:6	R/W	0x0	Reserved
5	R/W	0x0	Modem chip to RK28xx interrupt , driven by Modem chip ，only write by Modem chip
4	R/W	0x0	RK28xx to Modem chip interrupt , driven by RK28xx , only write by RK28xx
3	R/W	0x0	Valid level select for MCU interface signal 0 : low level 1 : high level Only write by RK28xx
2	R/W	0x0	Register access select 0: HIF_ADDR register 1: HIF_COUNT register
1	R/W	0x0	Byte Control Bit for count register: 0 : low byte 1 : high byte Modem chip must set this bit if necessary before every access, and do automatic refresh, only write by modem chip
0	R/W	0x0	Byte Control Bit for addr register: 0 : low byte 1 : high byte Modem chip must set this bit if necessary before every access, and do automatic refresh, only write by modem chip

HIF_INITCOUNT
Address: Base Addr+0x04
Hif transfer data byte register

bit	Attr	Reset Value	Description
15:0	R/W	0x0	Total byte numbers for data transfered

HIF_INITADDR
Address: Base Addr+0x08
Hif start address register

bit	Attr	Reset Value	Description
15:0	RW	0x0	Intial Address value for modem chip accessing RK28xx SRAM

HIF_ADDR
Address: Base Addr+0x0c
Hif address register

bit	Attr	Reset Value	Description
15:0	R	0x0	Access Address value for Modem chip accessing RK28xx SRAM,updated during accessing,only read by RK28xx

HIF_COUNT
Address: Base Addr+0x10
Hif transfer byte register

bit	Attr	Reset Value	Description
15:0	R	0x0	real rest byte numbers for data transferred, updated during accessing, only read by RK28xx

14.4 Application Notes

- **Host interface address map**
  - Modem chip can read/write registers and buffer inside RK28xx HIF module
  - Modem chip can access buffer with single mode or continuous burst mode
  - The above access type is decided by host_addr[1:0] from host interface, the detailed address map is shown as follows :

Table 14-1 Host interface address map table

host_addr[1:0]	Access type
2'b00	HIF_CON register
2'b01	HIF_INITCOUNT or HIF_INITADDR register
2'b10	2K SRAM with single mode
2'b11	2K SRAM with continuous burst mode, Internal address can be self-increased

- **Host interface function**
  - HIF support 8bits (default) and 16bits data bus width, which can be Programmable by bit 26 in CPU_APB_REG4. Refer to Chapter 34 (General Register File in CPU System) for detailed information.
  - It must have at least 2 cycles hclk interval between read and write operation of Host interface

- **Share Memory function**

HIF function can be disabled (default) by software set. Refer to bit 27 of CPU_APB_REG4 in Chapter 34 (General Register File in CPU System) for detailed information. After that, the 2KB size buffer will use for share memory between DSP and CPU.

- **LCD interface bypass function**

HIF interface can be used to bypass to LCDC interface of RK28xx, then modem chip will control LCD panel by HIF interface , which is controlled by RK28xx before. Remind that the panel type for this application scene is only MCU panel. As for the detailed information, please refer to bit 25 of CPU_APB_REG4 in Chapter 34.

The following table will list pin mapping between HIF and LCDC interface.

Table 14-2 Pin mapping between HIF and LCDC interface

HIF Pin Name	HIF port name	LCDC port name	LCDC Pin Name

IO_GPIO2[7:0]	host_wdata[7:0]	lcdc_wdata[7:0]	IO_LCDC_DATA[7:0]
IO_HOST_DATA_H8[7:0]	host_wdata[15:8]	lcdc_wdata[15:8]	IO_GPIO0_D[7:0]
IO_GPIO2[10]	host_csn	lcdc_vsync/lcdc_csn	IO_GPIO2[25]
IO_GPIO2[12]	host_wrn	lcdc_hsync/lcdc_wen	IO_LCDC_HSYNC
IO_GPIO2[8]	host_addr[0]	lcdc_dclk /lcdc_rs	IO_LCDC_DCLK

- **Host interface timing requirement**

The following waveform has shown the requirement for host interface timing. In them , the T is period for AHB bus clock inside RK28xx.

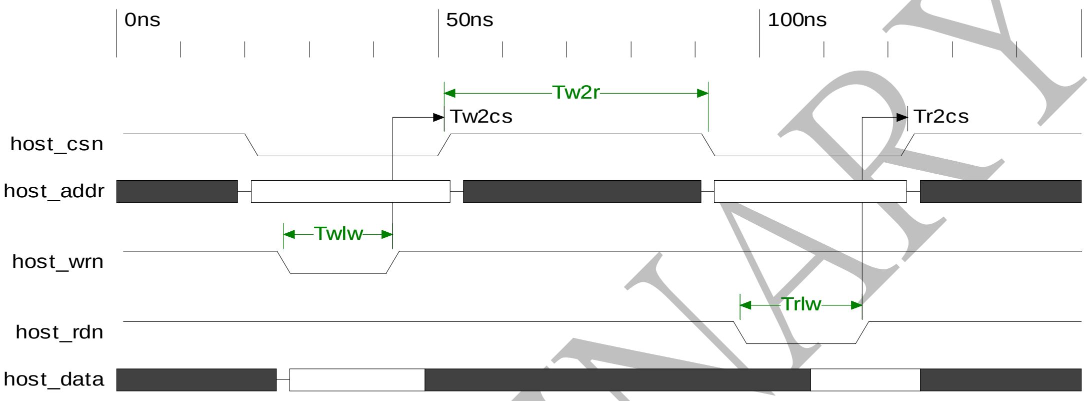


Fig. 14-2 Timing Diagram for host interface

Symbol	Description	Min (AHB cycle)	Max
Twlw	Write low width	2T	N/A
Trlw	Read low width	3T	N/A
Tw2cs	width from wrn invalid to csn invalid	T	N/A
Tr2cs	width from rdn invalid to csn invalid	T	N/A
Tw2r	interval width between read and write	3T	N/A

Chapter 15 USB OTG Controller

15.1 Design Overview

15.1.1 Overview

USB OTG Controller is a Dual-Role Device controller, which supports both device and host functions and is fully compliant with OTG Supplement to USB2.0 specification, and support high-speed (480Mbps), full-speed (12Mbps), low-speed (1.5Mbps) transfer. This controller will support UTMI+ Level 3 PHY interface. It connects to the industry-standard AMBA AHB for communication with the application and system memory. And it is optimized for portable electronic devices, point-to-point applications (no hub, direct coernnection to device) and multi-point applications to devices.

15.1.2 Features

- Compliant with the OTG Supplement to the USB2.0 Specification
- Operates in High-Speed and Full-Speed mode
- Supports UTMI+ Level 3 interfaces, and 16bit data bused will be used.
- Support Session Request Protocol(SRP) and Host Negotiation Protocol(HNP)
- Support 6 channels in host mode
- 6 Device mode endpoints in addition to control endpoint 0 , 3 in and 3 out
- Built-in one 1777 x 35bits FIFO
- Internal DMA with scatter/gather function
- Supports packet-based, dynamic FIFO memory allocation for endpoints for flexible,efficient use of RAM
- Provides support to change an endpoint`s FIFO memory size during transfers

For detailed information about USB OTG controller, please refer to **RK28xx USB OTG Controller.pdf**。

Chapter 16 System Control Unit (SCU)

16.1 Design Overview

16.1.1 Overview

The SCU is an APB slave module that is designed for generating all of the internal and system clocks, resets of chip and power domain control. 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.

16.1.2 Key Features

- Compliance to the AMBA APB interface
- Centralization of clock and reset sources control
- Five power management mode --- normal , slow , idle , stop, power off
- General peripheral control bit and chip status record
- Power domain control

16.2 Registers

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

16.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SCU_APLL_CON	0x0000	w	0x01850412	ARM PLL output freqrency control register (133MHz)
SCU_DPLL_CON	0x0004	w	0x01830310	DSP PLL output freqrency control register (300MHz)
SCU_CPLL_CON	0x0008	w	0x01980ff2	CODEC PLL output freqrency control register (122.88MHz)
SCU_MODE_CON	0x000c	w	0x00700207	System work mode control register
SCU_PMU_CON	0x0010	w	0x00000000	Power management control register
SCU_CLKSEL0_CON	0x0014	w	0x06300734	Clock divider frequency and select control register
SCU_CLKSEL1_CON	0x0018	w	0x0003000c	Clock divider frequency and select control register
SCU_CLKGATE0_CON	0x001c	w	0x00000000	Clock gating control register
SCU_CLKGATE1_CON	0x0020	w	0x00000000	Clock gating control register
SCU_CLKGATE2_CON	0x0024	w	0x00000000	Clock gating control register
SCU_SOFTRST_CON	0x0028	w	0x00000010	Soft reset control register
SCU_CHIPCFG_CON	0x002c	w	0x0bb80000	Chip general configuration register

SCU_CPUPD	0x0030	w	0x00000000	ARM926E Power down control register

16.2.2 Detail Register Description

SCU_APLL_CON
Address: Base Addr+0x00
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	0x41	ARM PLL CLKF factor control
3:1	RW	0x1	ARM PLL CLKOD factor control
0	RW	0x0	ARM PLL Bypass mode control 1: bypass 0: no bypass

SCU_DPLL_CON
Address: Base Addr+0x04
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	0x3	DSP PLL CLKR factor control
15:4	RW	0x31	DSP PLL CLKF factor control
3:1	RW	0x0	DSP PLL CLKOD factor control
0	RW	0x0	DSP PLL Bypass mode control

SCU_CPLL_CON
Address: Base Addr+0x08
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	0xff	CODEC PLL CLKF factor control
3:1	RW	0x1	CODEC PLL CLKOD factor control
0	RW	0x0	CODEC PLL Bypass mode control

SCU_MODE_CON
Address: Base Addr+0x0c
System work mode control register

bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	SCU interrupt clear bit 0 pending 1 clear
7	RW	0x0	Wakeup pin type selection 0 positive polarity 1 negative polarity
6	RW	0x0	Disable RTC alarm or interrupt wakeup stop mode 0 Enable RTC alarm or interrupt wakeup 1 Disable RTC alarm or interrupt wakeup
5	RW	0x0	Disable external wakeup stop mode 0 Enable external wakeup pin 1 Disable external wakeup pin
4	RW	0x0	Stop mode enable 0: disable 1: stop mode
3:2	RW	0x00	CPU work mode 00: CPU subsys slow mode 01: Normal mode 10: CPU subsys deep slow mode 11: CPU subsys slow mode
1:0	RW	0x00	DSP work mode 00: DSP subsys slow mode 01: Normal mode 10: DSP subsys deep slow mode 11: DSP subsys slow mode

SCU_PMU_MODE
Address: Base Addr+0x10
Power domain control register

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	R	0x0	Share Memory (Power Domain4) power on/off status 0 : power on 1 : power off
7	R	0x0	Demodulator (Power Domain3) power on/off status 0 : power on 1 : power off
6	R	0x0	CPU Subsys (Power Domain2) power on/off

			status 0 : power on 1 : power off
5	R	0x0	DSP Subsys (Power Domain1) power on/off status 0 : power on 1 : power off
4	RW	0x0	Enable CPU subsys(Power Domain2) power switch by external pin 0 Disable (default) 1 Enable
3	RW	0x0	Control Share Memory (Power Domain4) Power down or on 0 : Power on (default) 1 : Power down
2	RW	0x0	Control Demodulator (Power Domain3) Power down or on 0 : Power on (default) 1 : Power down
1	RW	0x0	Control CPU subsys (Power Domain2) Power down or on 0 : Power on (default) 1 : Power down
0	RW	0x0	Control DSP subsys (Power Domain1) Power down or on 0 : Power on (default) 1 : Power down

SCU_CLKSEL0_CON
Address: Base Addr+0x14
Internal clock select and divide register0

bit	Attr	Reset Value	Description
31:28	RW	4’b0	Reserved
27:25	RW	3’b011	Control sd/mmc1 clk frequency (hclk/(1~6)) mmc1_clk = arm_hclk /(mmc1clk_sel+1) default : arm_hclk/4
24:23	RW	2’b00	Select Sensor CLK 00: select 24MHz (default) 01 : select 27MHz 10 : select 48MHz
22:20	RW	3’b011	Control 48MHz clock divider frequency (armclk/(1~8)) clk48m = armclk/(clk48m_sel+1) default : armclk/4
19:18	RW	2’b00	Select USB PHY clk 00 : 24MHz (default) 01 : 12MHz 10 : 48MHz
17:16	RW	2’b00	Control LCDC CLK divider frequency source 00 : select armpll_clk (default) 01 : select dsppll_clk 10 : select codecpll_clk
15:8	RW	8’b00000111	Control LCDC CLK divider frequency value (pllclk/(1~128)) lcdclk = pllclk/(lcdclk_div_sel+1) default : pllclk/8
7	RW	1’b0	Select LCDC CLK

			0 : select divider output 1 : select 27MHz from external clock
6:4	RW	3'b011	Control sd/mmc0 clk frequency (hclk/(1~6)) mmc0_clk = arm_hclk /(mmc0clk_sel+1) default : arm_hclk/4
3:2	RW	2'b01	Control arm subsys pclk frequency 00 : hclk:pclk = 1:1 01 : hclk:pclk = 2:1 (default) 10 : hclk:pclk = 4:1
1:0	RW	2'b00	Control arm subsys hclk frequency 00 : armclk:hclk=1:1 (default) 01 : armclk:hclk=2:1 10 : armclk:hclk=3:1 11 : armclk:hclk=4:1

SCU_CLKSEL1_CON
Address: Base Addr+0x18
Internal clock select and divide register1

bit	Attr	Reset Value	Description
31	-	-	Reserved
30	RW	1'b0	Select share meory 1 clock from demod_clock or arm_clock 0: select demod_clock 1: select arm_hclk
29	RW	1'b0	Select share meory 0 clock from demod_clock or arm_clock 0: select demod_clock 1: select arm_hclk
28	RW	1'b0	Select HS_ADC clock output 0 : select demod_clock/2 (default) 1 : select demod_clock/2 inverted
27	RW	1'b0	Select GPS tuner input clock or not to hsadc interface 0 : not from GPS input clock (default) 1 : from GPS tuner input
26	RW	1'b0	Select demodulator clk from external clock or not 0 : internal divider out (default) 1 : external clock input
25:24	RW	2'b00	Control demodulator CLK divider frequency source 00 : select codecpll_clk (default) 01 : select armpll_clk 10 : select dsppll_clk
23:16	RW	8'b00000011	Control demodulator CLK divider frequency (xpll_clk/(1~128)) demod_clk = xpll_clk/(demod_clk_divcon+1) default : xpll_clk/4
15:8	RW	8'b0	Control LS_ADC CLK divider frequency (pclk/(1~128)) ladcclk = pclk/(ladcclk_sel+1)
7:3	RW	5'b00001	Control CODECCLK divider frequency (cpll_clk/(1~32)) codecclk = codecpll_clk/(codecclk_sel+1) default : cpll_clk/2
2	RW	1'b1	codecclk12m_sel Control CODECCLK work frequency 0 : select divider output from codec pll 1 : select 12MHz from osc input (default)

1:0	RW	2’b00	Codec PLL slow mode select 00 : slow mode, clock from external 24m osc (default) 01 : normal mode, clock from PLL 10 : deep slow mode

SCU_CLKGATE0_CON

Address: Base Addr+0x1c

Internal clock gating control register0

Bit	Attr	Reset Value	Description
31	RW	0x0	SD/MMC1 clock and hclk disable. When HIGH,disable clock
30	RW	0x0	Share_memory 1 (EQU) clock disable.
29	RW	0x0	Share_memory 0 (LDPC) clock disable.
28	RW	0x0	SAR-ADC clock and pclk disable. When HIGH,disable clock
27	RW	0x0	Rtc pclk disable. When HIGH,disable clock
26	RW	0x0	WDT pclk disable. When HIGH,disable clock
25	RW	0x0	Timer pclk disable. When HIGH,disable clock
24	RW	0x0	PWM pclk disable. When HIGH,disable clock
23	RW	0x0	Spi1 clock disable. When HIGH,disable clock
22	RW	0x0	spi0 clock disable. When HIGH,disable clock
21	RW	0x0	I2c1 clock disable. When HIGH, disable clock
20	RW	0x0	i2c0 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	Gpio E-H clock disable. When HIGH,disable clock
16	RW	0x0	Gpio A-D pclk disable. When HIGH,disable clock
15	RW	0x0	Embedded Rom clock disable. When HIGH,disable clock
14	RW	0x0	SD/MMC0 clock and hclk disable. When HIGH,disable clock
13	RW	0x0	i2s clock and pclk disable. When HIGH,disable clock
12	RW	0x0	viu clock and hclk disable. When HIGH,disable clock
11	RW	0x0	lcdc clock disable. When HIGH,disable clock
10	RW	0x0	deblocking hclk clock disable. When HIGH,disable clock
9	RW	0x0	intc hclk clock disable. When HIGH,disable clock
8	RW	0x0	nandc hclk clock disable. When HIGH,disable clock
7	RW	0x0	usb otg phy clock disable. When HIGH,disable clock
6	RW	0x0	usb otg bus side clock disable. When HIGH,disable clock
5	RW	0x0	HIF&SRAM block hif clock disable. When HIGH,disable clock
4	RW	0x0	HIF&SRAM block dsp bus clock disable. When HIGH,disable clock
3	RW	0x0	HIF&SRAM block arm bus clock disable. When HIGH,disable clock
2	RW	0x0	dma clock disable. When HIGH,disable clock
1	RW	0x0	dsp clock disable. When HIGH,disable clock

0	RW	0x0	arm core clock disable . When HIGH,disable clock

SCU_CLKGATE1_CON

Address: Base Addr+0x20

Internal clock gating control register1

Bit	Attr	Reset Value	Description
31:20	-	-	Reserved
19	RW	0x0	lcdc hclk clock disable. When HIGH,disable clock
18	RW	0x0	demodulator clock gating,this clock will always on when using demodulator. When HIGH,disable clock
17	RW	0x0	Msdr extmem hclk clock disable. When HIGH,disable clock
16	RW	0x0	Sdr extmem hclk clock disable. When HIGH,disable clock
15	RW	0x0	extmem hclk clock disable. When HIGH,disable clock
14	RW	0x0	Demodulator rs logic working clock disable
13	RW	0x0	Demodulator viterbi logic working clock disable
12	RW	0x0	Demodulator ldpc memory shared by pre-fft clock disable
11	RW	0x0	Demodulator ldpc memory shared by viterbi clock disable
10	RW	0x0	Demodulator memory shared by bit_deinterleave and viterbi logic clock disable
9	RW	0x0	Demodulator fft memory clock disable
8	RW	0x0	Demodulator frame_detect logic clock disable
7	RW	0x0	Demodulator iq_imbalance logic clock disable
6	RW	0x0	Demodulator pre fft logic clock disable
5	RW	0x0	Demodulator downmixer logic clock disable
4	RW	0x0	Demodulator agc logic clock disable
3	RW	0x0	Other demodulator submodules logic clock disable, including post_fft_clk, demod_equ_clk, deinter_clk, ldpc_clk and ldpc_bus_mem_clk . When HIGH,disable clock
2	RW	0x0	Demodulator AHB Bus 60MHz clock disable. When HIGH,disable clock
1	RW	0x0	Demodulator fifo logic clock disable. When HIGH,disable clock
0	RW	0x0	HS-ADC interface and logic clock disable. When HIGH,disable clock

SCU_CLKGATE2_CON

Address: Base Addr+0x24

Internal clock gating control register2

Bit	Attr	Reset Value	Description
31:9	-	-	Reserved
8	RW	0x0	ARM ITCM clock disable. When HIGH, disable clock
7	RW	0x0	ARM DTCM0 clock disable. When HIGH, disable clock logic
6	RW	0x0	ARM DTCM1 clock disable. When HIGH, disable clock
5	RW	0x0	EFUSE IP clock disable. When HIGH, disable clock
4	RW	0x0	APB bus logic clock disable. When HIGH, disable

			clock
3	RW	0x0	EXP AHB bus clock disable. When HIGH, disable clock
2	RW	0x0	DSP AHB bus clock disable. When HIGH, disable clock
1	RW	0x0	ARMD bus clock disable. When High, disable clock
0	RW	0x0	ARMI bus clock disable. When HIGH, disable clock

SCU_SOFTRST_CON
Address: Base Addr+0x28
Internal soft reset control register

Bit	Attr	Reset Value	Description
31:29			Reserved
28	RW	0x0	SDRAM controller reset request When HIGH, reset relative logic
27	RW	0x0	share memory 1 soft reset request. When HIGH, reset relative logic
26	RW	0x0	share memory 0 soft reset request. When HIGH, reset relative logic
25	RW	0x0	DSP A2A bridge soft reset request. When HIGH, reset relative logic
24	RW	0x0	SD/MMC1 soft reset request. When HIGH, reset relative logic
23	RW	0x0	ARM core soft reset request. When HIGH, reset relative logic
22	RW	0x0	Demodulator general logic soft reset request. When HIGH, reset relative logic
21	RW	0x0	Demodulator PRE_FFT soft reset request. When HIGH, reset relative logic
20	RW	0x0	Demodulator RS logic soft reset request. When HIGH, reset relative logic
19	RW	0x0	Demodulator viterbi & bit deinterleave logic soft reset request. When HIGH, reset relative logic
18	RW	0x0	Demodulator viterbi logic soft reset request. When HIGH, reset relative logic
17	RW	0x0	Demodulator fft logic soft reset request. When HIGH, reset relative logic
16	RW	0x0	Demodulator frame_detect logic soft reset request. When HIGH, reset relative logic
15	RW	0x0	Demodulator iq_imbalance logic soft reset request. When HIGH, reset relative logic
14	RW	0x0	Demodulator downmixer logic soft reset request. When HIGH, reset relative logic
13	RW	0x0	Demodulator agc logic soft reset request. When HIGH, reset relative logic
12	RW	0x0	USB PHY reset request. When HIGH, reset relative logic
11	RW	0x0	USB controller logic soft reset request. When HIGH, reset relative logic
10	RW	0x0	Demodulator soft reset request. When HIGH, reset relative logic
9	RW	0x0	SD/MMC0 soft reset request. When HIGH, reset relative logic
8	RW	0x0	Deblocking soft reset request. When HIGH, reset relative logic
7	RW	0x0	SAR_ADC soft reset request. When HIGH, reset

			relative logic
6	RW	0x0	I2S 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	NandC soft reset request. When HIGH, reset relative logic
2	RW	0x0	VIP soft reset request. When HIGH, reset relative logic
1	RW	0x0	LCDC soft reset request. When HIGH, reset relative logic
0	RW	0x0	USB OTG soft reset request,in HCLK domain. When HIGH, reset relative logic

SCU_CHIPCFG_CON
Address: Base Addr+0x2c
Chip config register

bit	Attr	Reset Value	Description
31:16	RW	0x0bb8	pll lock period control
15:0	-	-	reserved

SCU_CPUPD
Address: Base Addr+0x30
ARM power down control register

bit	Attr	Reset Value	Description
31:0	RW	0x0	If write " 0xdeed_babe" will stop ARM926 clock

16.3 Application Notes

16.3.1 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_APLL_CON, SCU_DPLL_CON and SCU_CPLL_CON 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:	97.7 KHz - 800MHz
Fref/NR * NF value range requirement:	160MHz - 800MHz

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_APLL_CON, SCU_DPLL_CON and SCU_CPLL_CON register or change chip from normal to slow mode by programming SCU_MODE_CON and SCU_CLKSEL1_CON register. The later method is recommended. Then until PLL is in lock state by check CPU_APB_REG0 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_APLL_CON, SCU_DPLL_CON and SCU_CPLL_CON register. After PLL will be out of powerdown mode, you can check CPU_APB_REG0 register to confirm PLL in lock state.

16.3.2 Power mode 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	133MHz (ARM) 300MHz (DSP) 122.88 (CODEC)
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 setting	Stop unused IP clock by software setting	On	Normal frequency or slow
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_CLKGATEx_CON(x=0~2), 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_CLKSELx_CON(x=0,1) register to select system clock source from PLL to the external OSC and set SCU_APLL_CON, SCU_DPLL_CON and SCU_CPLL_CON registers 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. 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.

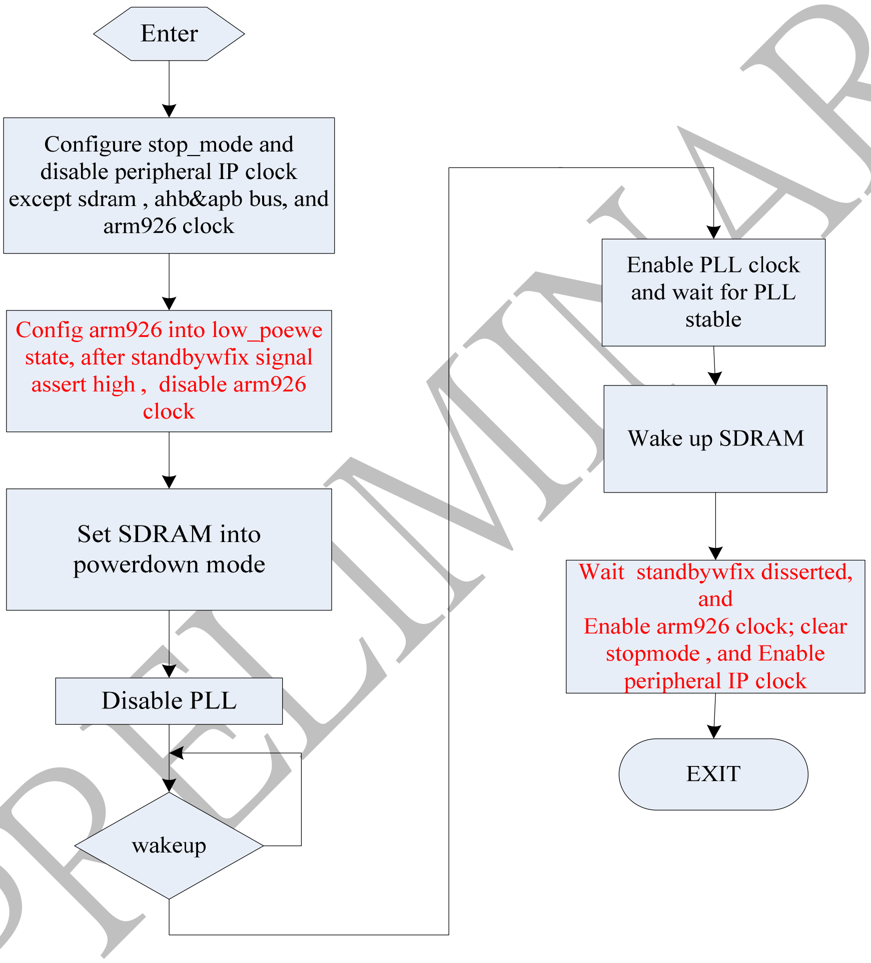

Enter
Configure stop_mode and disable peripheral IP clock except sdram , ahb&apb bus, and arm926 clock
Config arm926 into low_poewe state, after standbywfix signal assert high , disable arm926 clock
Set SDRAM into powerdown mode
Disable PLL
wakeup
Enable PLL clock and wait for PLL stable
Wake up SDRAM
Wait standbywfix disserted, and Enable arm926 clock; clear stopmode , and Enable peripheral IP clock
EXIT

Fig. 16-1 RK28xx system stop mode operation flow

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_CLKGATEx_CON(x=0~2) register.
- Slow mode:
  - SCU_MODE_CON[3:0] register to “1111” to select clock source to low speed clock.

  - Turn off the PLL by setting SCU_APLL_CON[22], SCU_DPLL_CON[22]
  - Before switch to PLL clock , turn on PLL by setting SCU_APLL_CON[22], SCU_DPLL_CON[22] and read CPU_APB_REG0[7] to check PLL lock status or delay 0.3ms.

- IDLE mode:
  - Program ARM926 to low_power state by instruction
    MCR p15,0 <Rd>,c7,c0,4
  - Idle mode will exit by external interrupt or wakeup pin

- STOP mode
  - Set SCU_CLKGATEx_CON(x=0~2) to disable all peripheral IP clock except sdram , ahb and apb bus, and arm926 clock
  - Set SCU_MODE_CON[6:5] to select wakeup stop mode method
  - Set RTC alarm time in RTC control register if use RTC alarm to wakeup
  - Set SCU_MODE_CON[7] to select ewakeup signal polarity if use external wakeup pin to wakeup
  - Set SCU_MODE_CON[4] and set SCU_CPUPD to 0xdeed_babe to enable stop mode
  - Program ARM926 to low_power state by instruction
    MCR p15,0 <Rd>,c7,c0,4
  - Check the interrupt status when wakeup from stop mode and use SCU_MODE_CON[8] to clear SCU_INT if system wakeup by external pin.

Chapter 17 PMU in CPU System

17.1 Design Overview

17.1.1 Overviews

The Power Management Unit (PMU) focuses on power on/off switch for differrent power domain in RK28xx. RK28xx has been divided into 6 independent power domain such as CPU System,DSP System,Share Memory , Demodulator and SCU , RTC . In them, SCU and RTC is always on power domain, not be switched off, the four other modules can be switched on/off for their power by software method. Therefore, when one module is not used in some application, we can make it power off to save power, even leakage power.

Another, after CPU system will be powered off, we can wake up it by external pin.

17.1.2 Features

The PMU has the following main feature:

- Support power off/on switch by software for 4 power domain
- Support external wake up for main module (CPU System)

17.2 Power Domain Architecture

The following diagram shows the different power domain in different colors.

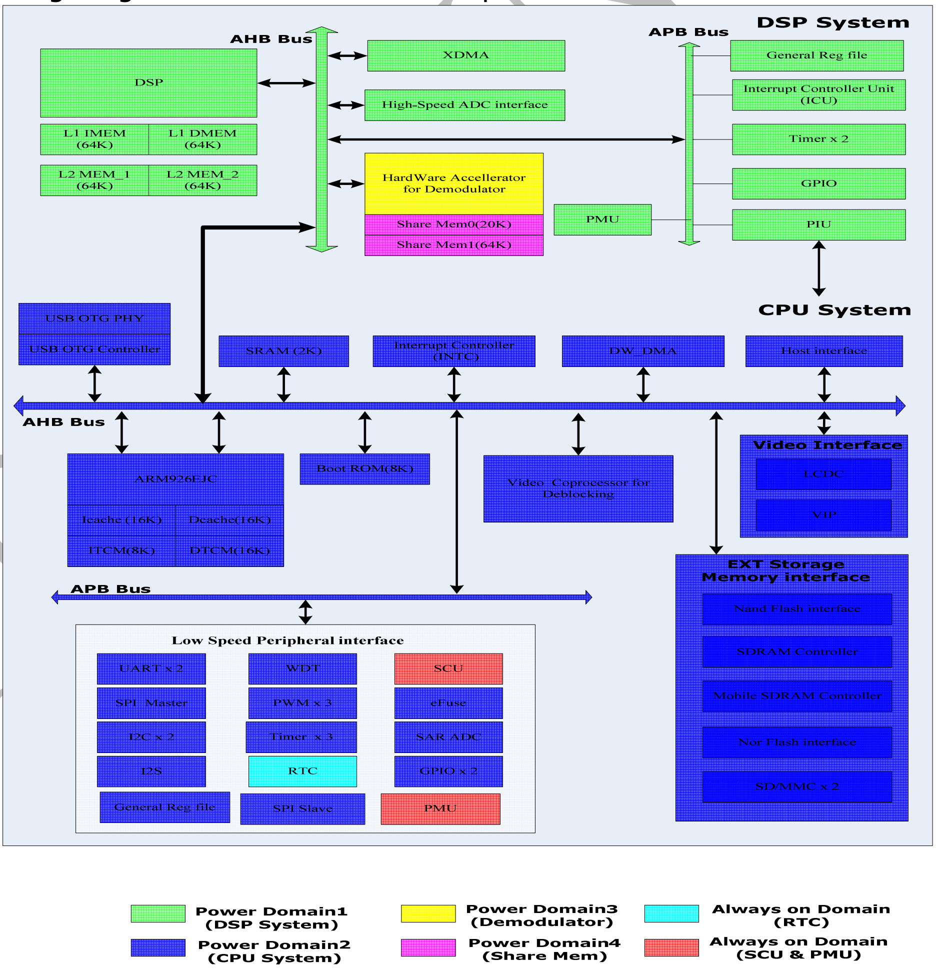


Fig. 17-1 RK28xx power domain architecture

17.3 Registers

As for PMU register, please refer to register SCU_PMU_MODE in Chapter 16(System Control Unit) for more detailed descriptions.

Chapter 18 Processor Interface Unit (PIU)

18.1 Overview

18.1.1 Overview

The Processor Interface Unit (PIU) is a simple XAPB peripheral that allows both the DSP and an external master MCU to share a semaphore and a command/reply protocol. The peripheral is mapped on the XAPB subsystem. XAPB registers are accessible by both the DSP I/O port and any external master via the ASHB Bridge. In addition, the module features an address snooping mechanism allowing an interrupt to be generated when a preset address range of the DSP is accessed by an external MCU.

18.1.2 Features

The PIU has the following main feature:

- Protocol registers are mapped on a XAPB peripheral bus
- Three Command/Reply protocols
- Three Semaphore registers
- Address Snooping mechanism with dedicated interrupt
- Configurable interrupt generation and five interrupt types, including:
  Three semaphore-related interrupts
  One command-reply-related interrupt
- Separate semaphore and command/reply interrupts for the MCU and the DSP

18.2 Registers

This section describe the registers of the PIU.

18.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SEM0S	0x00	W	0x00000000	Semaphore 0 Set Register
SEM1S	0x04	W	0x00000000	Semaphore 1 Set Register
SEM2S	0x08	W	0x00000000	Semaphore 2 Set Register
SEM0C	0x0c	W	0x00000000	Semaphore 0 Clear Register
SEM1C	0x10	W	0x00000000	Semaphore 1 Clear Register
SEM2C	0x14	W	0x00000000	Semaphore 2 Clear Register
MCU_MASK0	0x18	W	0x00000000	Semaphore 0 Interrupt Mask Register for MCU Interrupts
MCU_MASK1	0x1c	W	0x00000000	Semaphore 1 Interrupt Mask Register for MCU Interrupts
MCU_MASK2	0x20	W	0x00000000	Semaphore 2 Interrupt Mask Register for MCU Interrupts
CX_MASK0	0x24	W	0x00000000	Semaphore 0 Interrupt Mask Register for DSP Interrupt
CX_MASK1	0x28	W	0x00000000	Semaphore 1 Interrupt Mask Register for DSP Interrupt
CX_MASK20	0x2c	W	0x00000000	Semaphore 2 Interrupt Mask Register for DSP Interrupt
COM0	0x30	W	0x00000000	PIU Command Register 0
COM1	0x34	W	0x00000000	PIU Command Register 1
COM2	0x38	W	0x00000000	PIU Command Register 2
REP0	0x3c	W	0x00000000	PIU Reply Register 0
REP1	0x40	W	0x00000000	PIU Reply Register 1

REP2	0x44	W	0x00000000	PIU Reply Register 2
INTMSK	0x48	W	0x00000000	PIU Interrupt Mask Register
STATUS	0x4c	W	0x00000000	PIU Status Register
SNP_BASE0	0x50	W	0x00000000	Snoop Mechanism Base Register 0
SNP_BASE1	0x54	W	0x00000000	Snoop Mechanism Base Register 1
SNP_MSK0	0x58	W	0x00000000	Snoop Mechanism Mask Register 0
SNP_MSK1	0x5c	W	0x00000000	Snoop Mechanism Mask Register 1
SNP_EN	0x60	W	0x00000000	Snoop Mechanism Enable Register
SNP_STAT	0x64	W	0x00000000	Snoop Mechanism Status Register

19.2.2 Detail Registers Description

SEM0S

Address: PIU_BASE + offset(0x00)

Semaphore 0 Set Register

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SEM0S ： Semaphore 0 Set Register: SEM0S[x] = 0: No effect. SEM0S[x] = 1: Bit x in semaphore 0 is set. Reading SEM0S returns the semaphore 0 value.

SEM1S

Address: PIU_BASE + offset(0x04)

Semaphore 1 Set Register

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SEM1S ： Semaphore 1 Set Register: SEM1S[x] = 0: No effect. SEM1S[x] = 1: Bit x in semaphore 1 is set. Reading SEM1S returns the semaphore 1 value.

SEM2S

Address: PIU_BASE + offset(0x08)

Semaphore 2 Set Register

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SEM2S ： Semaphore 1 Set Register: SEM1S[x] = 0: No effect. SEM1S[x] = 1: Bit x in semaphore 2 is set. Reading SEM2S returns the semaphore 2 value.

SEM0C

Address: PIU_BASE + offset(0x0c)

Semaphore 0 Clear Register

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SEM0C ： Semaphore 0 Clear Register: SEM0C[x] = 0: No effect. SEM0C[x] = 1: Bit x in semaphore 0 is cleared. Reading SEM0C returns the semaphore 0 value.

SEM1C

Address: PIU_BASE + offset(0x10)

Semaphore 1 Clear Register

Bit	Att	Reset Value	Description

31:0	RW	0x00000000	SEM1C ： Semaphore 1 Clear Register: SEM1C[x] = 0: No effect. SEM1C[x] = 1: Bit x in semaphore 1 is cleared. Reading SEM1C returns the semaphore 1 value.

SEM2C
Address: PIU_BASE + offset(0x14)
Semaphore 2 Clear Register

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SEM2C ： Semaphore 2 Clear Register: SEM2C[x] = 0: No effect. SEM2C[x] = 1: Bit x in semaphore 2 is cleared. Reading SEM2C returns the semaphore 2 value.

MCU_MASK0
Address: PIU_BASE + offset(0x18)
Semaphore 0 Interrupt Mask Register for MCU Interrupt

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	MCU_MASK0： Disables/Enables Activation of HINT_SEM[0] when Semaphore 0 Bits are Set: MCU_MASK0[x] = 0: HINT_SEM[0] is not asserted when bit x in semaphore 0 is set. MCU_MASK0[x] = 1: HINT_SEM[0] is asserted when bit x in semaphore 0 is set.

MCU_MASK1
Address: PIU_BASE + offset(0x1c)
Semaphore 1 Interrupt Mask Register for MCU Interrupt

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	MCU_MASK1： Disables/Enables Activation of HINT_SEM[1] when Semaphore 1 Bits are Set: MCU_MASK1[x] = 0: HINT_SEM[1] is not asserted when bit x in semaphore 1 is set. MCU_MASK1[x] = 1: HINT_SEM[1] is asserted when bit x in semaphore 1 is set.

MCU_MASK2
Address: PIU_BASE + offset(0x20)
Semaphore 2 Interrupt Mask Register for MCU Interrupt

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	MCU_MASK2： Disables/Enables Activation of HINT_SEM[2] when Semaphore 2 Bits are Set: MCU_MASK2[x] = 0: HINT_SEM[2] is not asserted when bit x in semaphore 2 is set. MCU_MASK2[x] = 1: HINT_SEM[2] is asserted when bit x in semaphore 2 is set.

CX_MASK0
Address: PIU_BASE + offset(0x24)
Semaphore 0 Interrupt Mask Register for DSP Interrupt

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	CX_MASK0：

			Disables/Enables Activation of PIU_SEM_INT[0] when Semaphore 0 Bits are Set: CX_MASK0[x] = 0: PIU_SEM_INT[0] is not asserted when bit x in semaphore 0 is set. CX_MASK0[x] = 1: PIU_SEM_INT [0] is asserted when bit x in semaphore 0 is set.

CX_MASK1
Address: PIU_BASE + offset(0x28)
Semaphore 1 Interrupt Mask Register for DSP Interrupt

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	Disables/Enables Activation of PIU_SEM_INT[1] when Semaphore 1 Bits are Set: CX_MASK1[x] = 0: PIU_SEM_INT[1] is not asserted when bit x in semaphore 1 is set. CX_MASK1[x] = 1: PIU_SEM_INT [1] is asserted when bit x in semaphore 1 is set.

CX_MASK2
Address: PIU_BASE + offset(0x2c)
Semaphore 2 Interrupt Mask Register for DSP Interrupt

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	CX_MASK2： Disables/Enables Activation of PIU_SEM_INT[2] when Semaphore 2 bits are Set: CX_MASK2[x] = 0: PIU_SEM_INT[2] is not asserted when bit x in semaphore 2 is set. CX_MASK2[x] = 1: PIU_SEM_INT [2] is asserted when bit x in semaphore 2 is set.

COM0
Address: PIU_BASE + offset (0x30)
PIU Command Register 0

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	CX_MASK2： PIU Command 0 - Either the MCU or the DSP may access this register for transferring commands according to the software protocol.

COM1
Address: PIU_BASE + offset(0x34)
PIU Command Register 1

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	COM1： PIU Command 1 - Either the MCU or the DSP may access this register for transferring commands according to the software protocol.

COM2
Address: PIU_BASE + offset(0x38)
PIU Command Register 2

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	COM2： PIU Command 2 - Either the MCU or the DSP may access this register for transferring commands according to the software protocol.

REP0

Address: PIU_BASE + offset(0x3c)

PIU Reply Register 0

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	REP0： PIU Reply 0 - Either the MCU or the DSP may access this register for transferring replies according to the software protocol.

REP1

Address: PIU_BASE + offset(0x40)

PIU Reply Register 1

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	REP1： PIU Reply 1 - Either the MCU or the DSP may access this register for transferring replies according to the software protocol.

REP2

Address: PIU_BASE + offset(0x44)

PIU Reply Register 2

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	REP2： PIU Reply 2 - Either the MCU or the DSP may access this register for transferring replies according to the software protocol.

INTMSK

Address: PIU_BASE + offset(0x48)

PIU Interrupt Mask Register

Bit	Att	Reset Value	Description
31:12	-	-	Reserved
11	RW	0x0	R2RDCXIE ： Reply Register 2 DSP Interrupt Enable at Read: R2RDCXIE = 0: PIU_INT_CR is not affected by a Read access to REP2 register. R2RDCXIE = 1: A Read access to REP2 asserts PIU_INT_CR interrupt.
10	RW	0x0	R1RDCXIE ： Reply Register 1 DSP Interrupt Enable at Read: R1RDCXIE = 0: PIU_INT_CR is not affected by a Read access to REP1 register. R1RDCXIE = 1: A Read access to REP1 asserts PIU_INT_CR interrupt.
9	RW	0x0	R0RDCXIE ： Reply Register 0 DSP Interrupt Enable at Read: R0RDCXIE = 0: PIU_INT_CR is not affected by a Read access to REP0 register. R0RDCXIE = 1: A Read access to REP0 asserts PIU_INT_CR interrupt.
8	RW	0x0	C2WRCXIE : Command Register 2 DSP Interrupt Enable at Write: C2WRHIE = 0: HINT_CR is not affected by a Write access to COM2 register. C2WRHIE = 1: A Write access to COM2 asserts HINT_CR interrupt.

7	RW	0x0	C1WRCXIE : Command Register 1 DSP Interrupt Enable at Write: C1WRHIE = 0: HINT_CR is not affected by a Write access to COM1 register. C1WRHIE = 1: A Write access to COM1 asserts HINT_CR interrupt.
6	RW	0x0	C0WRCXIE : Command Register 0 DSP Interrupt Enable at Write: C0WRHIE = 0: HINT_CR is not affected by a Write access to COM0 register. C0WRHIE = 1: A Write access to COM0 asserts HINT_CR interrupt.
5	RW	0x0	R2WRHIE : Reply Register 2 MCU Interrupt Enable at Write: R2WRHIE = 0: PIU_INT_CR is not affected by a Write access to REP2 register. R2WRHIE = 1: A Write access to REP2 asserts PIU_INT_CR interrupt.
4	RW	0x0	R1WRHIE : Reply Register 1 MCU Interrupt Enable at Write: R1WRHIE = 0: PIU_INT_CR is not affected by a Write access to REP1 register. R1WRHIE = 1: A Write access to REP1 asserts PIU_INT_CR interrupt.
3	RW	0x0	R0WRHIE : Reply Register 0 MCU Interrupt Enable at Write: R0WRHIE = 0: PIU_INT_CR is not affected by a Write access to REP0 register. R0WRHIE = 1: A Write access to REP0 asserts PIU_INT_CR interrupt.
2	RW	0x0	C2RDHIE : Command Register 2 MCU Interrupt Enable at Read: C2RDHIE = 0: HINT_CR is not affected by a Read access to COM2 register. C2RDHIE = 1: A Read access to COM2 asserts HINT_CR interrupt.
1	RW	0x0	C1RDHIE : Command Register 1 MCU Interrupt Enable at Read: C1RDHIE = 0: HINT_CR is not affected by a Read access to COM1 register. C1RDHIE = 1: A Read access to COM1 asserts HINT_CR interrupt.
0	RW	0x0	C0RDHIE : Command Register 0 MCU Interrupt Enable at Read: C0RDHIE = 0: HINT_CR is not affected by a Read access to COM0 register. C0RDHIE = 1: A Read access to COM0 asserts HINT_CR interrupt.

STATUS

Address: PIU_BASE + offset(0x4c)

PIU Status Register

Bit	Att	Reset Value	Description
31:12	-	-	Reserved
11	RW	0x0	R2RDS : REP2 Register Access with Read Status: R2RDS = 0: No REP2 Read access was performed since

			last time this bit was cleared. R2RDS = 1: REP2 was read by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
10	RW	0x0	R1RDS : REP1 Register Access with Read Status: R1RDS = 0: No REP1 Read access was performed since last time this bit was cleared. R1RDS = 1: REP1 was read by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
9	RW	0x0	R0RDS : REP0 Register Access with Read Status: R0RDS = 0: No REP0 Read access was performed since last time this bit was cleared. R0RDS = 1: REP0 was read by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
8	RW	0x0	C2WRS : COM2 Register Access with Write Status: C2WRS= 0: No COM2 Write access was performed since last time this bit was cleared. C2WRS = 1: COM2 was written by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
7	RW	0x0	C1WRS : COM1 Register Access with Write Status: C1WRS= 0: No COM1 Write access was performed since last time this bit was cleared. C1WRS = 1: COM1 was written by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
6	RW	0x0	C0WRS : COM0 Register Access with Write Status: C0WRS= 0: No COM0 Write access was performed since last time this bit was cleared. C0WRS = 1: COM0 was written by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
5	RW	0x0	R2WRS : REP2 Register Access with Write Status: R2WRS= 0: No REP2 Write access was performed since last time this bit was cleared. R2WRS = 1: REP2 was written by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
4	RW	0x0	R1WRS : REP1 Register Access with Write Status: R1WRS= 0: No REP1 Write access was performed since last time this bit was cleared. R1WRS = 1: REP1 was written by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
3	RW	0x0	R0WRS : REP0 Register Access with Write Status: R0WRS= 0: No REP0 Write access was performed since last time this bit was cleared.

			R0WRS = 1: REP0 was written by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
2	RW	0x0	C2RDS : COM2 Register Access with Read Status: C2RDS = 0: No COM2 Read access was performed since last time this bit was cleared. C2RDS = 1: COM2 was read by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
1	RW	0x0	C1RDS : COM1 Register Access with Read Status: C1RDS = 0: No COM1 Read access was performed since last time this bit was cleared. C1RDS = 1: COM1 was read by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.
0	RW	0x0	C0RDS : COM0 Register Access with Read Status: C0RDS = 0: No COM0 Read access was performed since last time this bit was cleared. C0RDS = 1: COM0 was read by software. This bit is set automatically by the hardware and cleared by writing it with 1. Writing 0 does not affect this bit.

SNP_BASE0
Address: PIU_BASE + offset(0x50)
Snoop Mechanism Base Register 0

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SNP_BASE0: Snoop Mechanism Address Match Base Register 0 - Used to set the base address of the address range on which the snoop mechanism should detect an access.

SNP_BASE1
Address: PIU_BASE + offset(0x54)
Snoop Mechanism Base Register 1

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SNP_BASE1: Snoop Mechanism Address Match Base Register 1 - Used to set the base address of the address range on which the snoop mechanism should detect an access.

SNP_MSK0
Address: PIU_BASE + offset(0x58)
Snoop Mechanism Mask Register 0

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SNP_MSK0: 10-bit Mask Field Used to Specify an Address Range Based on SNP_BASE0 - Each set bit in SNP_MSK0 corresponds to one of the least significant address bits of SNP_BASE0. When a bit is set, the match address in the corresponding bit position is Don’t Care

SNP_MSK1
Address: PIU_BASE + offset(0x5c)
Snoop Mechanism Mask Register 1

Bit	Att	Reset Value	Description
31:0	RW	0x00000000	SNP_MSK1: 10-bit Mask Field Used to Specify an Address Range Based on SNP_BASE1 - Each set bit in SNP_MSK1 corresponds to one of the least significant address bits of SNP_BASE1. When a bit is set, the match address in the corresponding bit position is Don't Care

SNP_EN
Address: PIU_BASE + offset(0x60)
Snoop Mechanism Enable Register

Bit	Att	Reset Value	Description
31:4	-	-	Reserved

3	RW	0x0	SNP_MSK1:
			Enable Activation of SNOOP_INT when a Match with Snoop Address 1 Occurs in a Write Transfer:
			SNP1WRIE = 0: Interrupt is disabled.
			SNP1WRIE = 1: Interrupt is enabled.

2	RW	0x0	SNP0WRIE :
			Enable Activation of SNOOP_INT when a Match with Snoop Address 0 Occurs in a Write Transfer:
			SNP0WRIE = 0: Interrupt is disabled.
			SNP0WRIE = 1: Interrupt is enabled.

1	RW	0x0	SNP1RDIE :
			Enable Activation of SNOOP_INT when a Match with Snoop Address 1 Occurs In a Read Transfer:
			SNP1RDIE = 0: Interrupt is disabled.
			SNP1RDIE = 1: Interrupt is enabled.

0	RW	0x0	SNP0RDIE :
			Enable Activation of SNOOP_INT when a Match with Snoop Address 0 Occurs In a Read Transfer:
			SNP0RDIE = 0: Interrupt is disabled.
			SNP0RDIE = 1: Interrupt is enabled.

SNP_STAT
Address: PIU_BASE + offset(0x64)
Snoop Mechanism Status Register

Bit	Att	Reset Value	Description
31:4	-	-	Reserved
3	RW	0x0	SNP1WRS ： Status Bit Indicating a Master Write Access to Address Range Specified by Snp_Base1 And Snp_Msk1: SNP1WRS = 0: Snoop match occurred. SNP1WRS = 1: Snoop match did not occur since last time bit was cleared.
2	RW	0x0	SNP0WRS ： Status Bit Indicating a Master Write Access to Address Range Specified by SNP_BASE0 and SNP_MSK0: SNP0WRS = 0: Snoop match occurred. SNP0WRS = 1: Snoop match did not occur since last time bit was cleared.
1	RW	0x0	SNP1RDS ： Status Bit Indicating a Master Read Access to Address Range Specified by SNP_BASE1 and SNP_MSK1: SNP1RDS = 0: Snoop match occurred.

			SNP1RDS = 1: Snoop match did not occur since last time bit was cleared.
0	RW	0x0	SNP0RDS ： Status Bit Indicating a Master Read Access to Address Range Specified by SNP_BASE0 and SNP_MSK0: SNP0RDS = 0: Snoop match occurred. SNP0RDS = 1: Snoop match did not occur since last time bit was cleared.

18.3 Operation

18.3.1 Semaphore Protocol

The PIU semaphore protocol enables the MCU and the DSP to interface with a semaphore flag mechanism. Three independent 32-bit semaphore registers, SEM0/1/2, allow the sides to send a semaphore word to each other. Each semaphore register has a semaphore-set address and a semaphore-clear address. The semaphore Protocol includes the Semaphore-related Interrupts and Command/Reply Protocol.

To enable interruption of the MCU when a specific semaphore bit is written, the corresponding bit in the MCU_MASKx must be set. The PIU output, HINT[x], may be used as an input to an MCU interrupt controller. To enable interrupting the DSP when a specific semaphore bit is written, the corresponding bit in the CX_MASKx must be set. The PIU output, PIU_INT[x], is connected as one of the interrupt inputs to the ICU module. The three semaphore interrupts are level-active, meaning once set, they retain their logic state until all semaphores enabled for triggering an interrupt are cleared by software.

The Command/Reply protocol allows the MCU and the DSP to bidirectional communicate by commands and replies via the three sets of command and reply registers. In a typical application, the MCU sends commands to the DSP (DSPS) and receives replies from the DSP. According to this terminology, a set of interrupt events may be coupled with reading and writing the command and reply registers. This is summarized in the table below.

Access	COM/REP	Asserted Interrupt Output	Internal Recipient	Bit to Set	In CFG Register	Affected Status Bit
Read	COMx	HINT_CR	MCU	CxRDHIE	INTMSK	CxRDS
Write	COMx	PIU_INT_CR	DSP	CxWRCXIE		CxWRS
Write	REPx	HINT_CR	MCU	RxWRHIE		RxWRS
Read	REPx	PIU_INT_CR	DSP	RxRDCXIE		RxRDS

18.3.2 Address Snooping Mechanism

The PIU can detect an access to one or two AHB address ranges by an external MCU master and selectively generate an interrupt to the DSP on the occurrence of the MCU access. The interrupt output for the snooping mechanism is SNPINT. In order to set an interrupt, take the following steps:

- Set a 32-bit base address value to one of the SNP_BASEx registers.
- Set a 10-bit mask to specify an address range in the corresponding SNP_MSKx field in the SNP_MSKx register. A 1 value in a SNP_MSKx bit specifies the corresponding bit in the SNP_BASEx as a Don't Care bit.
- Enable an interrupt for a Read or a Write event by setting one or both SNPx{RD/WR} bit fields in the CFG register.
- Enable the SNPINT interrupt source in the ICU programmable model. To learn which ICU source is connected to SNPINT, refer to the Top Connectivity appendix.
- After the setup, a specified match event causes the SNPINT bit to be set. In addition, a status bit in SNP_STAT register is set to indicate the condition. The respective status bits are SNPx{RD/WR}s. When a match event occurs and a status bit is set, it must be cleared by the software by writing the corresponding

status bit with 1. Clearing the status bit causes the interrupt signal to become inactive again.

The timing diagram in the following figure demonstrates an interrupt generated by the snooping mechanism. One of the snooping address registers is preset to interrupt when an access to A3 is performed. SNPINT interrupt, is activated when the data phase of the transfer to A3 is complete. The interrupt is level-active and is cleared when the snoop status bit is cleared; Writing it with 1 clears the SNP_STAT register.

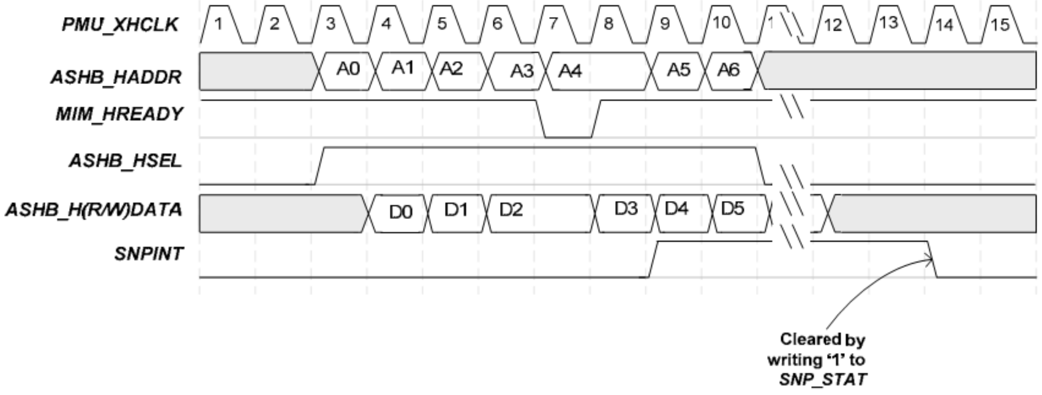


Fig. 18-1 PIU snooping mechanism timing waveform

18.4 Application Notes

<1> Enabling an interrupt in the INTMSK register while the interrupt event's corresponding status bit in the STATUS register is set immediately triggers an interrupt. It is recommended to clear STATUS bits before enabling interrupts in the INTMSK register.

Chapter 19 UART

19.1 Design Overview

19.1.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) is used for serial communication with a peripheral, modem (data carrier equipment, DCE) or data set. Data is written from a master (CPU) over the APB bus to the UART and it is converted to serial form and transmitted to the destination device. Serial data is also received by the UART and stored for the master (CPU) to read back.

19.1.2 Features

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- DMA Controller Interface – Enables interface to a DMA controller over the AMBA bus using a handshaking interface for transfer requests.
- Support interrupt interface to interrupt controller.
- Two 32x8bits fifos for transferring and receiving use respectively.
- Programmable serial data baud rate as calculated by the following:baud rate = (serial clock frequency)/(16×divisor).
- IrDA 1.0 SIR mode support with up to 115.2 Kbaud data rate and a pulse duration
- (width) as follows: width = 3/16 × bit period as specified in the IrDA physical layer specification.
- UART0 supports modem function,UART1 supports IrDA function.

19.2 Architecture

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

19.2.1 Block Diagram

The UART comprises with:

- AMBA APB interface
- FIFO controllers
- Register block
- Modem synchronization block and baud clock generation block
- Serial receiver and serial transmitter

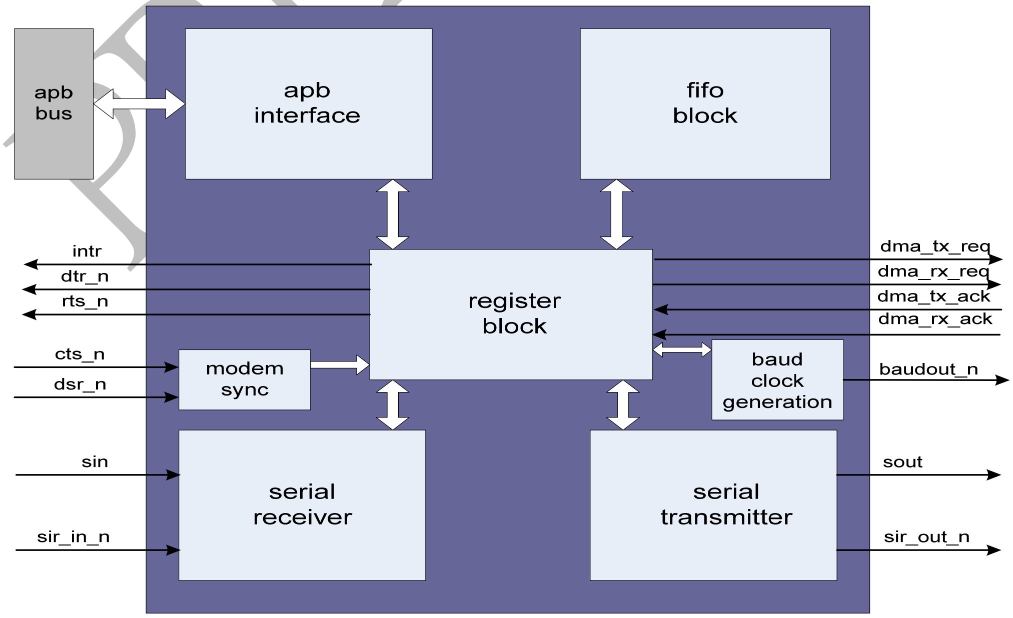


Figure 19-1 UART architecture

19.2.2 Block Descriptions

APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

Register block

Be responsible for the main UART functionality including control, status and interrupt generation.

Modem Synchronization block

Synchronizes the modem input signal.

FIFO block

Be responsible for FIFO control and storage (when using internal RAM) or signaling to control external RAM (when used).

Baud Clock Generator

Produces the transmitter and receiver baud clock along with the output reference clock signal (baudout_n).

Serial Transmitter

Converts the parallel data, written to the UART, into serial form and adds all additional bits, as specified by the control register, for transmission. This makeup of serial data, referred to as a character can exit the block in two forms, either serial UART format or IrDA 1.0 SIR format.

Serial Receiver

Converts the serial data character (as specified by the control register) received in either the UART or IrDA 1.0 SIR format to parallel form. Parity error detection, framing error detection and line break detection is carried out in this block.

19.3 Registers

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

19.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
UART_RBR	0x0000	W	0x0000_0000	Receive Buffer Register
UART_THR				Transmit Holding Register
UART_DLL				Divisor Latch (Low)
UART_DLH	0x0004	W	0x0000_0000	Divisor Latch (High)
UART_IER				Interrupt Enable Register
UART_IIR	0x0008	W	0x0000_0000	Interrupt Identification Register
UART_FCR				FIFO Control Register
UART_LCR	0x000C	W	0x0000_0000	Line Control Register
UART_MCR	0x0010	W	0x0000_0000	Modem Control Register
UART_LSR	0x0014	W	0x0000_0060	Line Status Register
UART_MSR	0x0018	W	0x0000_0000	Modem Status Register
UART_SCR	0x001c	W	0x0000_0000	Scratchpad Register
Reserved	0x0020 -2C	W	0x0000_0000	--
UART_SRBR	0x0030 -6C	W	0x0000_0000	Shadow Receive Buffer Register
UART_STHR		W	0x0000_0000	Shadow Transmit Holding Register
UART_FAR	0x0070	W	0x0000_0000	FIFO Access Register
UART_TFR	0x0074	W	0x0000_0000	Transmit FIFO Read
UART_RFW	0x0078	W	0x0000_0000	Receive FIFO Write
UART_USR	0x007C	W	0x0000_0006	UART Status Register
UART_TFL	0x0080	W	0x0000_0000	Transmit FIFO Level
UART_RFL	0x0084	W	0x0000_0000	Receive FIFO Level
UART_SRR	0x0088	W	0x0000_0000	Software Reset Register
UART_SRTS	0x008C	W	0x0000_0000	Shadow Request to Send
UART_SBCR	0x0090	W	0x0000_0000	Shadow Break Control Register

UART_SDMAM	0x0094	W	0x0000_0000	Shadow DMA Mode
UART_SFE	0x0098	W	0x0000_0000	Shadow FIFO Enable
UART_SRT	0x009C	W	0x0000_0000	Shadow RCVR Trigger
UART_STET	0x00A0	W	0x0000_0000	Shadow TX Empty Trigger
UART_HTX	0x00A4	W	0x0000_0000	Halt TX
UART_DMASA	0x00A8	W	0x0000_0000	DMA Software Acknowledge
Reserved	0x00AC -F0	W	0x0000_0000	--
UART_CPR	0x00F4	W	0x0000_0000	Component Parameter Register
UART_UCV	0x00F8	W	0x3330_372a	UART Component Version
UART_CTR	0x00FC	W	0x4457_0110	Component Type Register

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

19.3.2 Detail Register Description

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

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line Status Register (LCR) is set. If in non-FIFO mode (FIFO_MODE == NONE) or FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an over-run error. If in FIFO mode (FIFO_MODE != NONE) and FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.

UART_THR
Address: Operational Base + offset(0x00)
Transmit Holding Register

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR when the THR Empty (THRE) bit (LSR[5]) is set. If in non-FIFO mode or FIFOs are disabled (FCR[0] = 0) and THRE is set, writing a single character to the THR clears the THRE. Any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If in FIFO mode and FIFOs are enabled (FCR[0] = 1) and THRE is set, x number of characters of data may

			be written to the THR before the FIFO is full. The number x (default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.

UART_DLL

Address: Operational Base + offset(0x00)

Divisor Latch (Low)

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Lower 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DLAB bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The output baud rate is equal to the serial clock (sclk) frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor). Note that with the Divisor Latch Registers (DLL and DLH) set to zero, the baud clock is disabled and no serial communications occur. Also, once the DLH is set, at least 8 clock cycles of the slowest Uart clock should be allowed to pass before transmitting or receiving data.

UART_DLH

Address: Operational Base + offset(0x04)

Divisor Latch (High)

bit	Attr	Reset Value	Description
31:8	RW	0X0	Reserved
7:0	RW	0x0	Upper 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.

UART_IER

Address: Operational Base + offset(0x04)

Interrupt Enable Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	Reserved and read as zero
7	RW	0x0	Programmable THRE Interrupt Mode Enable This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled
6:4	Reserved and read as zero		
3	RW	0X0	Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0 = disabled 1 = enabled
2	RW	0X0	Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0 = disabled 1 = enabled

1	RW	0X0	Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = disabled 1 = enabled
0	RW	0X0	Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled). These are the second highest priority interrupts. 0 = disabled 1 = enabled

UART_IIR
Address: Operational Base + offset(0x08)
Interrupt Identification Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	
7:6	R	0x01	FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00 = disabled 11 = enabled
5:4	Reserved and read as zero		
3:0	R	0x01	Interrupt ID. This indicates the highest priority pending interrupt which can be one of the following types: 0000 = modem status 0001 = no interrupt pending 0010 = THR empty 0100 = received data available 0110 = receiver line status 0111 = busy detect 1100 = character timeout

UART_FCR
Address: Operational Base + offset(0x08)
FIFO Control Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	Reserved and read as zero
7:6	W	0x0	RCVR Trigger. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. In auto flow control mode it is used to determine when the rts_n signal is de-asserted. It also determines when the dma_rx_req_n signal is asserted in certain modes of operation. The following trigger levels are supported: 00 = 1 character in the FIFO 01 = FIFO ¼ full 10 = FIFO ½ full 11 = FIFO 2 less than ful
5:4	W	0x0	TX Empty Trigger. This is used to select the empty threshold level at which the THRE Interrupts are generated when the mode is active. It also determines when the dma_tx_req_n signal is asserted when in certain

			modes of operation. The following trigger levels are supported: 00 = FIFO empty 01 = 2 characters in the FIFO 10 = FIFO ¼ full 11 = FIFO ½ full
3	W	0x0	DMA Mode. This determines the DMA signalling mode used for the dma_tx_req_n and dma_rx_req_n output signals when additional DMA handshaking signals are not selected . 0 = mode 0 1 = mode 11100 = character timeout
2	W	0x0	XMIT FIFO Reset. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals when additional DMA handshaking signals are selected . Note that this bit is 'self-clearing'. It is not necessary to clear this bit.
1	W	0x0	RCVR FIFO Reset. This resets the control portion of the receive FIFO and treats the FIFO as empty. This also de-asserts the DMA RX request and single signals when additional DMA handshaking signals are selected . Note that this bit is 'self-clearing'. It is not necessary to clear this bit
0	W	0x0	FIFO Enable. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.

UART_LCR

Address: Operational Base + offset(0x0C)

Line Control Register

bit	Attr	Reset Value	Description
31:8	RW	0x0	Reserved and read as zero
7	RW	0x0	Divisor Latch Access Bit.Writeable only when UART is not busy (USR[0] is zero), always readable. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers
6	RW	0x0	Break Control Bit.This is used to cause a break condition to be transmitted to the receiving device. If set to one the serial output is forced to the spacing (logic 0) state. When not in Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. If MCR[6] set to one, the sir_out_n line is continuously pulsed. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low
5	Reserved and read as zero		
4	RW	0x0	Even Parity Select. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select between even and odd parity, when parity is enabled (PEN set to one). If set to one, an even number of logic 1s is transmitted or checked. If set to

			zero, an odd number of logic 1s is transmitted or checked.
3	RW	0x0	Parity Enable. Writeable only when UART is not busy (USR[0] is zero), always readable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively. 0 = parity disabled 1 = parity enabled
2	RW	0x0	Number of stop bits. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data.If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. 0 = 1 stop bit 1 = 1.5 stop bits when DLS (LCR[1:0]) is zero, else 2 stop bit
1:0	RW	0x0	Data Length Select. Writeable only when UART is not busy (USR[0] is zero), always readable. This is used to select the number of data bits per character that the peripheral transmits and receives. The number of bit that may be selected areas follows: 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits

UART_MCR
Address: Operational Base + offset(0x10)
Modem Control Register

bit	Attr	Reset Value	Description
31:7	RW	0x0	Reserved and read as zero
6	RW	0x0	SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode . 0 = IrDA SIR Mode disabled 1 = IrDA SIR Mode enabled
5	RW	0x0	Auto Flow Control Enable. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled
4	RW	0x0	LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes.
3	RW	0x0	OUT2. This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is: 0 = out2_n de-asserted (logic 1) 1 = out2_n asserted (logic 0)
2	RW	0x0	OUT1
1	RW	0x0	Request to Send. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is

			ready to exchange data
0	RW	0x0	Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is: 0 = dtr_n de-asserted (logic 1) 1 = dtr_n asserted (logic 0)

UART_LSR
Address: Operational Base + offset(0x14)
Line Status Register

bit	Attr	Reset Value	Description
31:8	Reserved and read as zero		
7	R	0x0	Receiver FIFO Error bit. This bit is relevant FIFOs are enabled (FCR[0] set to one). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0 = no error in RX FIFO 1 = error in RX FIFO
6	R	0x1	Transmitter Empty bit. If FIFOs enabled (FCR[0] set to one), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.
5	R	0x1	Transmit Holding Register Empty bit. If THRE mode is disabled (IER[7] set to zero) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THRE Interrupt to occur, if the THRE Interrupt is enabled. If IER[7] set to one and FCR[0] set to one respectively, the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.
4	R	0x0	Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.
3	R	0x0	Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data.
2		0x0	Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) bit (LCR[3]) is set.
1	R	0x0	Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read.
0	R	0x0	Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR or the receiver FIFO. 0 = no data ready 1 = data ready

UART_MSR
Address: Operational Base + offset(0x18)
Modem Status Register

bit	Attr	Reset Value	Description
31:8	Reserved		
7	R	0x0	Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n.
6	R	0x0	Ring Indicator. This is used to indicate the current state of the modem control line ri_n.
5	R	0x0	Data Set Ready. This is used to indicate the current state of the modem control line dsr_n.
4	R	0x0	Clear to Send. This is used to indicate the current state of the modem control line cts_n.
3	R	0x0	Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read.
2	R	0x0	Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read.
1	R	0x0	Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read.
0	R	0x0	Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read.

UART_SCR
Address: Operational Base + offset(0x1C)
Scratchpad Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x0	This register is for programmers to use as a temporary storage space.

UART_SRBR
Address: Operational Base + offset(0x30-6C)
Shadow Receive Buffer Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	This is a shadow register for the RBR and has been allocated sixteen 32-bit locations so as to accommodate burst accesses from the master. This register contains the data byte received on the serial input port (sin) in UART mode or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if the Data Ready (DR) bit in the Line status Register (LSR) is set. If FIFOs are disabled (FCR[0] set to zero), the data in the RBR must be read before the next data arrives, otherwise it is overwritten, resulting in an overrun error. If FIFOs are enabled (FCR[0] set to one), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO are preserved, but any incoming data is lost. An

			overrun error also occurs

UART_STHR
Address: Operational Base + offset(0x30-6C)
Shadow Transmit Holding Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	This is a shadow register for the THR.

UART_FAR
Address: Operational Base + offset(0x70)
FIFO Access Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are implemented and enabled. When FIFOs are not enabled it allows the RBR to be written by the master and the THR to be read by the master. 0 = FIFO access mode disabled 1 = FIFO access mode enabled

UART_TFR
Address: Operational Base + offset(0x74)
Transmit FIFO Read

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x0	Transmit FIFO Read. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are implemented and enabled, reading this register gives the data at the top of the transmit FIFO. Each consecutive read pops the transmit FIFO and gives the next data value that is currently at the top of the FIFO. When FIFOs are not implemented or not enabled, reading this register gives the data in the THR.

UART_RFW
Address: Operational Base + offset(0x78)
Receive FIFO Write

bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9	W	0x0	Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).
8	W	0x0	Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).
7:0	W	0x0	Receive FIFO Write Data. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one). When FIFOs are enabled, the data that is written to the RFWD is pushed into the receive FIFO. Each consecutive write pushes the new data to the next write location in the receive FIFO. When FIFOs not enabled, the data that is written to the

			RFWD is pushed into the RBR.

UART_USR
Address: Operational Base + offset(0x7C)
UART Status Register

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4	R	0x0	Receive FIFO Full. This is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full
3	R	0x0	Receive FIFO Not Empty. This is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty
2	R	0x1	Transmit FIFO Empty. This is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty
1	R	0x1	Transmit FIFO Not Full. This is used to indicate that the transmit FIFO in not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full.
0	R	0x0	UART Busy. This is indicates that a serial transfer is in progress, when cleared indicates that the uart is idle or inactive. 0 = Uart is idle or inactive 1 = Uart is busy (actively transferring data)

UART_TFL
Address: Operational Base + offset(0x80)
Transmit FIFO Level

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4:0	R	0x0	Transmit FIFO Level. This is indicates the number of data entries in the transmit FIFO.

UART_RFL
Address: Operational Base + offset(0x84)
Receive FIFO Level

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4:0	R	0x0	Receive FIFO Level. This is indicates the number of data entries in the receive FIFO.

UART_SRR
Address: Operational Base + offset(0x88)
Software Reset Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	W	0x0	XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]).

1	W	0x0	RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]).
0	W	0x0	UART Reset. This asynchronously resets the Uart and synchronously removes the reset assertion. For a two clock implementation both pclk and sclk domains are reset.

UART_SRTS
Address: Operational Base + offset(0x8C)
Shadow Request to Send

bit	Attr	Reset Value	Description
31:1	-	-	
0	RW	0x0	Shadow Request to Send. This is a shadow register for the RTS bit (MCR[1]), this can be used to remove the burden of having to performing a read-modify-write on the MCR.

UART_SBCR
Address: Operational Base + offset(0x90)
Shadow Break Control Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	Shadow Break Control Bit. This is a shadow register for the Break bit (LCR[6]), this can be used to remove the burden of having to performing a read modify write on the LCR.

UART_SDMAM
Address: Operational Base + offset(0x94)
Shadow DMA Mode

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
	RW	0x0	Shadow DMA Mode. This is a shadow register for the DMA mode bit (FCR[3]).

UART_SFE
Address: Operational Base + offset(0x98)
Shadow FIFO Enable

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]).

UART_SRT
Address: Operational Base + offset(0x9C)
Shadow RCVR Trigger

bit	Attr	Reset Value	Description
31:2	-	-	Reserved
1:0	RW	0x0	Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]).

UART_STET
Address: Operational Base + offset(0Xa0)
Shadow TX Empty Trigger

bit	Attr	Reset Value	Description
31:2	-	-	Reserved

1:0	RW	0x0	Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]).

UART_HTX
Address: Operational Base + offset(0Xa4)
Halt TX

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
	RW	0x0	This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 0 = Halt TX disabled 1 = Halt TX enabled

UART_DMASA
Address: Operational Base + offset(0xa8)
RTC counter reset register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	W	0x0	This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition.

UART_UCV
Address: Operational Base + offset(0xf8)
UART Component Version

bit	Attr	Reset Value	Description
31:0	R	0x330372a	ASCII value for each number in the version

UART_CTR
Address: Operational Base + offset(0xfc)
Component Type Register

bit	Attr	Reset Value	Description
31:0	R	0x44570110	This register contains the peripherals identification code.

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

19.4 Functional Description

19.4.1 Operation

- **UART (RS232) Serial Protocol**

Because the serial communication is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. An additional parity bit may be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure to perform simple error checking on the received data., as shown in Figure.

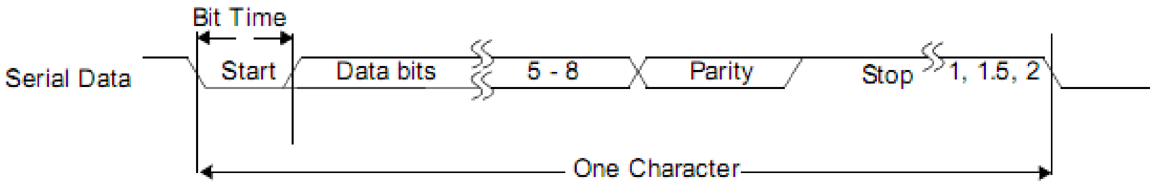


Figure 19-2 UART Serial protocol

- **Baud Clock**

The baud rate controlled by the serial clock (sclk or pclk in a single clock implementation) and the Divisor Latch Register (DLH and DLL). As the exact number of baud clocks that each bit was transmitted for is known, calculating the mid point for sampling is not difficult, that is every 16 baud clocks after the mid point sample of the start bit.

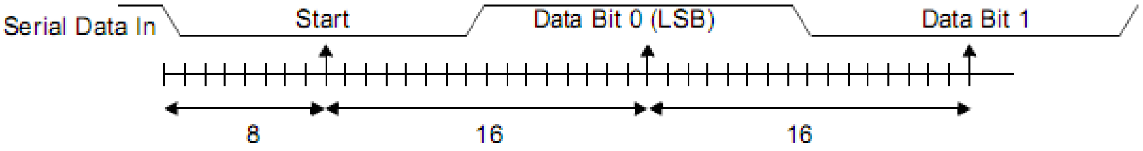


Figure 19-3 UART baud rate

- **IrDA 1.0 SIR Protocol**

The Infrared Data Association (IrDA) 1.0 Serial Infrared (SIR) mode supports bi-directional data communications with remote devices using infrared radiation as the transmission medium. IrDA 1.0 SIR mode specifies a maximum baud rate of 115.2 Kbaud. Transmitting a single infrared pulse signals a logic zero, while a logic one is represented by not sending a pulse. The width of each pulse is 3/16ths of a normal serial bit time. data transfers can only occur in half-duplex fashion when IrDA SIR mode is enabled.

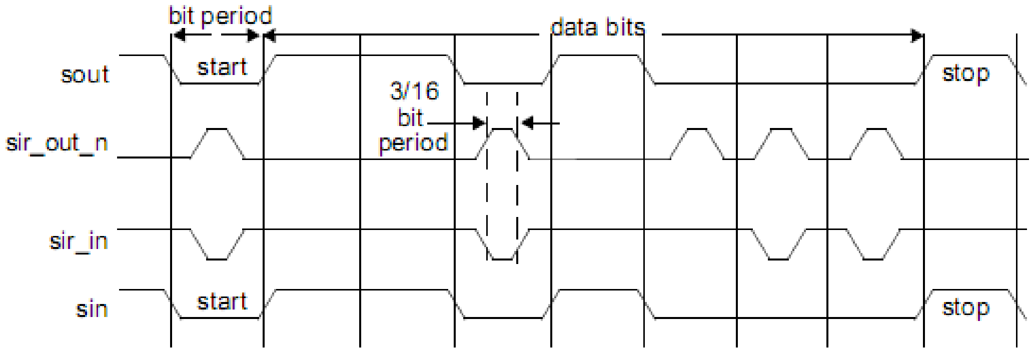


Figure 19-4 IrDA1.0 timing waveform

- **FIFO Support**

1. NONE FIFO MODE

If FIFO support is not selected, then no FIFOs are implemented and only a single receive data byte and transmit data byte can be stored at a time in the RBR and THR.

2. FIFO MODE

The FIFO depth is 32, enabled by register FCR[0].

- **Interrupts**

The following interrupt types can be enabled with the IER register.
Receiver Error;
Receiver Data Available;
Character Timeout (in FIFO mode only);
Transmitter Holding Register Empty at/below threshold (in Programmable THRE I nterrupt mode);
Modem Status;

- **DMA Support**

The uart supports DMA signalling with the use of two output signals (dma_tx_req_n and dma_rx_req_n) to indicate when data is ready to be read or when the transmit FIFO is empty.

The dma_tx_req_n signal is asserted under the following conditions:

a) When the Transmitter Holding Register is empty in non-FIFO mode
b) When the transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled
c) When the transmitter FIFO is at, or below the programmed threshold with Programmable THRE interrupt mode enabled.

The dma_rx_req_n signal is asserted under the following conditions:

a) When there is a single character available in the Receive Buffer Register in non-FIFO mode

b) When the Receiver FIFO is at or above the programmed trigger level in FIFO mode

- **Auto Flow Control**

The uart can be configured to have a 16750-compatible Auto RTS and Auto CTS serial data flow control mode available. If FIFOs are not implemented, then this mode cannot be selected. When Auto Flow Control mode has been selected it can be enabled with the Modem Control Register (MCR[5]). Following figure shows a block diagram of the Auto Flow Control functionality.

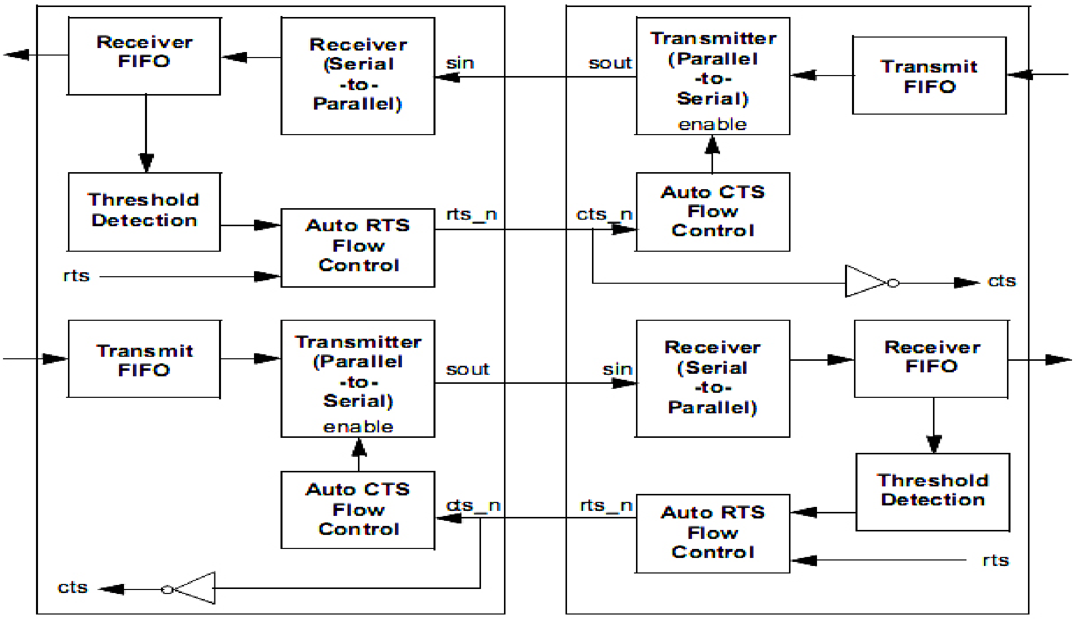


Figure 19-5 UART Auto flow control block diagram

Auto RTS – Becomes active when the following occurs:

❍ Auto Flow Control is selected during configuration

❍ FIFOs are implemented

❍ RTS (MCR[1] bit and MCR[5]bit are both set)

❍ FIFOs are enabled (FCR[0]) bit is set)

❍ SIR mode is disabled (MCR[6] bit is not set)

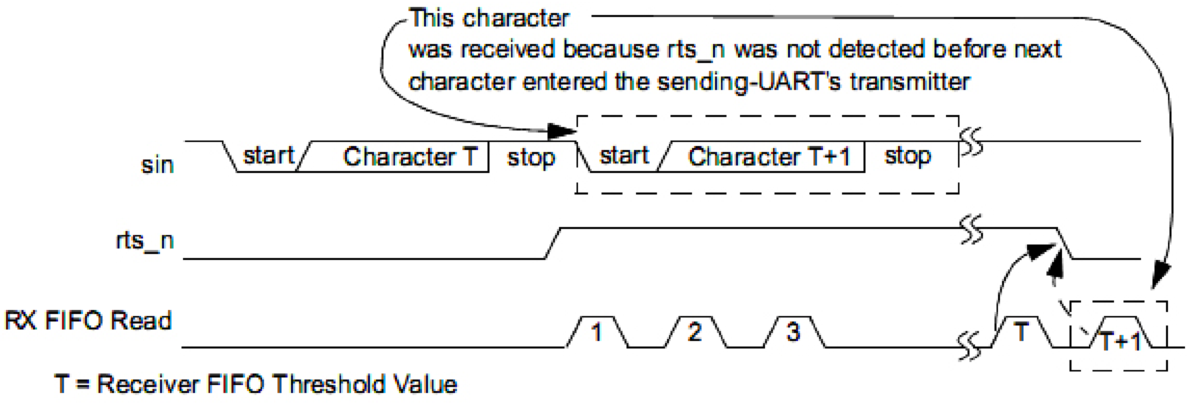

This character was received because rts_n was not detected before next character entered the sending-UART's transmitter
sin
start
Character T
stop
start
Character T+1
stop
rts_n
RX FIFO Read
1
2
3
T
T+1
T = Receiver FIFO Threshold Value

Figure 19-6 UART AUTO RTS TIMING

Auto CTS – becomes active when the following occurs:

❍ Auto Flow Control is selected during configuration

❍ FIFOs are implemented

❍ AFCE (MCR[5] bit is set)

❍ FIFOs are enabled through FIFO Control Register FCR[0] bit

❍ SIR mode is disabled (MCR[6] bit is not set)

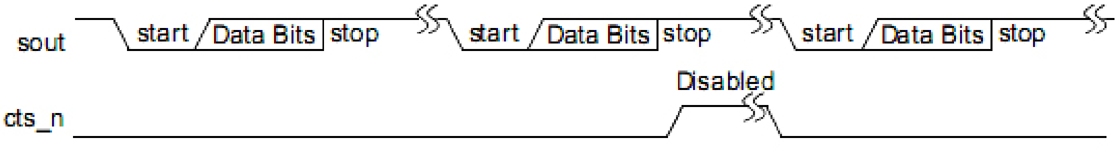


Figure 19-7 AUTO CTS TIMING

19.4.2 Programming sequence

- **None FIFO Mode Transfer Flow**

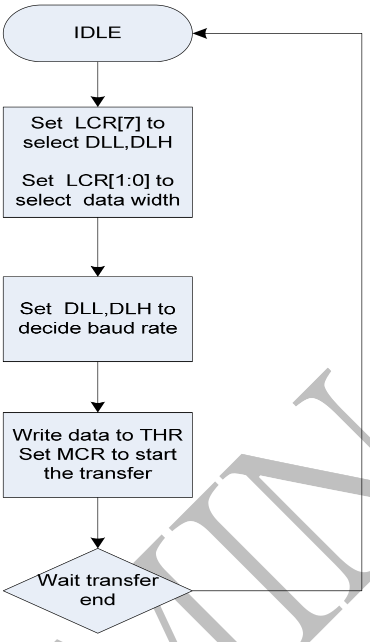

IDLE
Set LCR[7] to select DLL,DLH
Set LCR[1:0] to select data width
Set DLL,DLH to decide baud rate
Write data to THR Set MCR to start the transfer
Wait transfer end

Figure 19-8 Uart none fifo mode

- **FIFO Mode Transfer Flow**

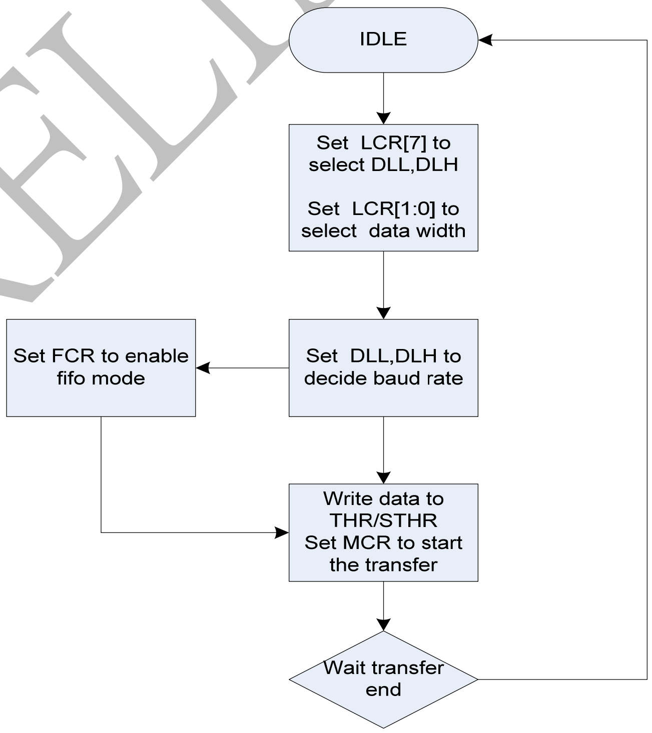

IDLE
Set LCR[7] to select DLL,DLH
Set LCR[1:0] to select data width
Set FCR to enable fifo mode
Set DLL,DLH to decide baud rate
Write data to THR/STHR Set MCR to start the transfer
Wait transfer end

Figure 19-9 Uart fifo mode flow diagram

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.

Chapter 20 SPI Master Controller

20.1 Design Overview

20.1.1 Overview

The Serial Peripheral Interface 0(SPI0) is an APB slave device. A four-wire, full-duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The slave select line is held high when the spi is idle or disabled.SPI master controller only work as master mode.

20.1.2 Features

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- Serial-master operation – Enables serial communication with serial-slave peripheral devices.
- DMA Controller Interface – Enables interface to a DMA controller over the AMBA bus using a handshaking interface for transfer requests.
- Support interrupt interface to interrupt controller, and independently masking of interrupts.
- Dedicated 2 hardware slave-select lines.
- Dynamic control of the serial bit rate of the data transfer.
- Two 16x16 fifos for transferring and receiving use respectively.

20.2 Architecture

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

20.2.1 Block Diagram

The SPI comprises with:

- AMBA APB interface and DMA Controller Interface
- Transmit and receive FIFO controllers and an FSM controller
- Register block
- Shift control and interrupt

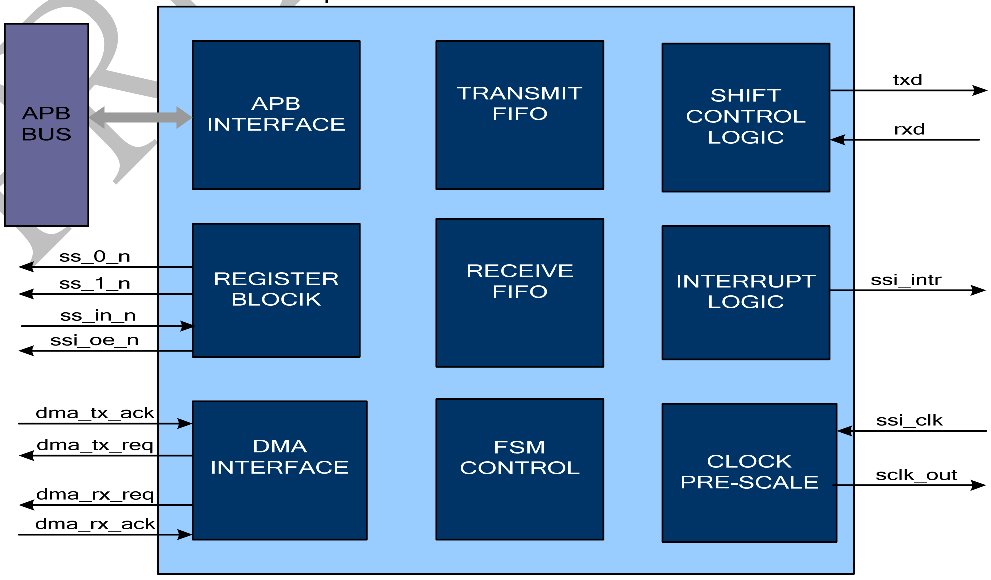


Figure 20-1 SPI Master Controller Block diagram

20.2.2 Block Descriptions

APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

DMA INTERFACE

This block has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA Controller.

FIFO LOGIC

For transmit and receive transfers, data transmitted from the SPI to the external serial device is written into the transmit FIFO. Data received from the external serial device into the SPI is pushed into the receive FIFO. Both fifos are 16x16bits.

FSM CONTROL

Control the state's transformation of the design.

REGISTER BLOCK

All registers in the SPI are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bits wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

SHIFT CONTROL

Shift control logic shift the data from the transmit fifo or to the receive fifo. This logic automatically right-justifies receive data in the receive FIFO buffer

INTERRUPT CONTROL

The SPI supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other SPI interrupts after masking.

20.3 Registers

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

20.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SPIM_CTRLR0	0x0000	W	0x0000_0007	Control register 0
SPIM_CTRLR1	0x0004	W	0x0000_0000	Control register 1
SPIM_SPIENR	0x0008	W	0x0000_0000	Ssi enable register
SPIM_MWCR	0x000C	W	0x0000_0000	Microwire control register
SPIM_SER	0x0010	W	0x0000_0000	Slave enable register
SPIM_BAUDR	0x0014	W	0x0000_0000	Baud rate select
SPIM_TXFTLR	0x0018	W	0x0000_0000	Transmit FIFO Threshold Level
SPIM_RXFTLR	0x001c	W	0x0000_0000	Receive FIFO Threshold Level
SPIM_TXFLR	0x0020	W	0x0000_0000	Transmit FIFO Level Register
SPIM_RXFLR	0x0024	W	0x0000_0000	Receive FIFO Level Register
SPIM_SR	0x0028	W	0x0000_0006	Status Register
SPIM_IMR	0x002c	W	0x0000_003f	Interrupt Mask Register
SPIM_ISR	0x0030	W	0x0000_0000	Interrupt Status Register
SPIM_RISR	0x0034	W	0x0000_0000	Raw Interrupt Status Register
SPIM_TXOICR	0x0038	W	0x0000_0000	Transmit FIFO Overflow Interrupt Clear Register
SPIM_RXOICR	0x003c	W	0x0000_0000	Receive FIFO Overflow Interrupt Clear Register

SPIM_RXUICR	0x0040	W	0x0000_0000	Receive FIFO Underflow Interrupt Clear Register
SPIM_MSTICR	0x0044	W	0x0000_0000	Multi-Master Interrupt Clear Register
SPIM_ICR	0x0048	W	0x0000_0000	Interrupt Clear Register
SPIM_DMACR	0x004c	W	0x0000_0000	DMA Control Register
SPIM_DMATDLR	0x0050	W	0x0000_0000	DMA Transmit Data Level
SPIM_DMARDLR	0x0054	W	0x0000_0000	DMA Receive Data Level
SPIM_IDR	0x0058	W	0xffff_ffff	Identification Register
SPIM_SPI_COMP_V ERSION	0x005c	W	0x3331_302a	coreKit version ID register
SPIM_DR	0x0060 −9C	W	0x0000_0000	Data Register

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

20.3.2 Detail Register Description

SPIM_CTRL0
Address: Operational Base + offset (0x00)
Control register 0

bit	Attr	Reset Value	Description
15:12	RW	0X0	Control Frame Size. Selects the length of the control word for the Microwire frame format. For the field decode,
11	RW	0x0	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 – Normal Mode Operation 1 – Test Mode Operation
10	RW	0x0	Slave Output Enable.
9:8	RW	0x0	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. 00 –- Transmit & Receive 01 –- Transmit Only 10 –- Receive Only 11 –- EEPROM Read
7	RW	0x0	Serial Clock Polarity 0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high
6	RW	0x0	Serial Clock Phase 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit
5:4	RW	0x0	Frame Format. Selects which serial protocol transfers the data. 00 –- Motorola SPI 01 –- Texas Instruments SSP 10 –- National Semiconductors Microwire 11 –- Reserved
3:0	RW	0x7	Data Frame Size. Selects the data frame length . When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded. You must right-justify transmit data before writing into the

			transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. 0000: Reserved – undefined operation 0001: Reserved – undefined operation 0010: Reserved – undefined operation 0011: 4-bit serial data transfer 0100: 5-bit serial data transfer 0101: 6-bit serial data transfer 0110: 7-bit serial data transfer 0111: 8-bit serial data transfer 1000: 9-bit serial data transfer 1001: 10-bit serial data transfer 1010: 11-bit serial data transfer 1011: 12-bit serial data transfer 1100: 13-bit serial data transfer 1101: 14-bit serial data transfer 1110: 15-bit serial data transfer 1111: 16-bit serial data transfer

SPIM_CTRL1
Address: Operational Base + offset(0x04)
Control register 1

bit	Attr	Reset Value	Description
15:0	RW	0x0	Number of Data Frames. When TMOD = 10, this register field sets the number of data frames to be continuously received by the SPI. The Spi continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. When the SPI is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the Spi is configured as a serial slave

SPIM_SPIENR
Address: Operational Base + offset(0x08)
Spi enable register

bit	Attr	Reset Value	Description
0	RW	0x0	SPI Enable. Enables and disables all Spi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the Spi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

SPIM_MWCR
Address: Operational Base + offset(0x0C)
Microwire control register

bit	Attr	Reset Value	Description
2	RW	0x0	Microwire Handshaking.
1	RW	0x0	Microwire Control.
0	RW	0x0	Microwire Transfer Mode.

SPIM_SER

Address: Operational Base + offset(0x10)
Slave enable register

bit	Attr	Reset Value	Description
1:0	RW	0x0	Slave Select Enable Flag. Each bit in this register corresponds to a slave select line (ss_x_n) from the Spi master.

SPIM_BAUDR
Address: Operational Base + offset(0x14)
Baud rate select

Bit	Attr	Reset Value	Description
15:0	RW	0x0	SPI Clock Divider. The LSB for this field is always set to 0 and is unaffected by a write operation, which ensures an even value is held in this register. If the value is 0, the serial output clock (sclk_out) is disabled. The frequency of the sclk_out is derived from the following equation: $F_{sclk\_out} = F_{ssi\_clk}/ SCKDV$ where SCKDV is any even value between 2 and 65534. For example: for F_{ssi_clk} = 3.6864MHz and SCKDV =2 F_{sclk_out} = 3.6864/2 = 1.8432MHz

SPIM_TXFTLR
Address: Operational Base + offset(0x18)
Transmit FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.

SPIM_RXFTLR
Address: Operational Base + offset(0x1C)
Receive FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

SPIM_TXFLR
Address: Operational Base + offset(0x20)
Transmit FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

SPIM_RXFLR
Address: Operational Base + offset(0x24)
Receive FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO

SPIM_SR

Address: Operational Base + offset(0x28)

Status Register

bit	Attr	Reset Value	Description
6	R	0x0	Data Collision Error. Relevant only when the Spi is configured as a master device. This bit is set if the Spi master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error
5	R	0x0	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the Spi is configured as a slave device. Data from the previous transmission is resent on the txd line. This bit is cleared when read. 0 – No error 1 – Transmission error
4	R	0x0	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When thereceive FIFO contains one or more empty location, this bit is cleared. 0 – Receive FIFO is not full 1 – Receive FIFO is full
3	R	0x0	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty
2	R	0x1	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty
1	R	0x1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full
0	R	0x0	SPI Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that the Spi is idle or disabled. 0 – Spi is idle or disabled 1 – Spi is actively transferring data

SPIM_IMR

Address: Operational Base + offset(0x2C)

Interrupt Mask Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	RW	0x1	Multi-Master Contention Interrupt Mask. This bit field is not present if the spi is configured as a serial-slave device. 0 – ssi_mst_intr interrupt is masked 1 – ssi_mst_intr interrupt is not masked
4	RW	0x1	Receive FIFO Full Interrupt Mask

			0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked
3	RW	0x1	Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked
2	RW	0x1	Receive FIFO Underflow Interrupt Mask 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked
1	RW	0x1	Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked
0	RW	0x1	Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked

SPIM_ISR
Address: Operational Base + offset(0x30)
Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt not active after masking 1 = ssi_mst_intr interrupt is active after masking
4	R	0x0	Receive FIFO Full Interrupt Status 0 = ssi_rxf_intr interrupt is not active after masking 1 = ssi_rxf_intr interrupt is full after masking
3	R	0x0	Receive FIFO Overflow Interrupt Status 0 = ssi_rxo_intr interrupt is not active after masking 1 = ssi_rxo_intr interrupt is active after masking
2	R	0x0	Receive FIFO Underflow Interrupt Status 0 = ssi_rxu_intr interrupt is not active after masking 1 = ssi_rxu_intr interrupt is active after masking
1	R	0x0	Transmit FIFO Overflow Interrupt Status 0 = ssi_txo_intr interrupt is not active after masking 1 = ssi_txo_intr interrupt is active after masking
0	R	0x0	Transmit FIFO Empty Interrupt Status 0 = ssi_txe_intr interrupt is not active after masking 1 = ssi_txe_intr interrupt is active after masking

SPIM_RISR
Address: Operational Base + offset(0x34)
Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt is not active prior to masking 1 = ssi_mst_intr interrupt is active prior masking
4	R	0x0	Receive FIFO Full Raw Interrupt Status 0 = ssi_rxf_intr interrupt is not active prior to masking 1 = ssi_rxf_intr interrupt is active prior to masking
3	R	0x0	Receive FIFO Overflow Raw Interrupt Status 0 = ssi_rxo_intr interrupt is not active prior to masking 1 = ssi_rxo_intr interrupt is active prior masking
2	R	0x0	Receive FIFO Underflow Raw Interrupt Status

			0 = ssi_rxu_intr interrupt is not active prior to masking 1 = ssi_rxu_intr interrupt is active prior to masking
1	R	0x0	Transmit FIFO Overflow Raw Interrupt Status 0 = ssi_txo_intr interrupt is not active prior to masking 1 = ssi_txo_intr interrupt is active prior masking
0	R	0x0	Transmit FIFO Empty Raw Interrupt Status 0 = ssi_txe_intr interrupt is not active prior to masking 1 = ssi_txe_intr interrupt is active prior masking

SPIM_TXOICR
Address: Operational Base + offset(0x38)
Transmit FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

SPIM_RXOICR
Address: Operational Base + offset(0x3C)
Receive FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.

SPIM_RXUICR
Address: Operational Base + offset(0x40)
Receive FIFO Underflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect

SPIM_MSTICR
Address: Operational Base + offset(0x44)
Multi-Master Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect.

SPIM_ICR
Address: Operational Base + offset(0x48)
Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.

SPIM_DMACR
Address: Operational Base + offset(0x4C)
DMA Control Register

bit	Attr	Reset Value	Description
1	RW	0x0	Transmit DMA Enable. This bit enables/disables the

			transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
0	RW	0x0	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 = Receive DMA disabled 1 = Receive DMA enabled

SPIM_DMATDLR
Address: Operational Base + offset(0x50)
DMA Transmit Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

SPIM_DMARDLR
Address: Operational Base + offset(0x54)
DMA Receive Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.

SPIM_IDR
Address: Operational Base + offset(0x58)
Identification Register

bit	Attr	Reset Value	Description
31:0	R	0xffffffff	Identification Code.

SPIM_SPI_COMP_VERSION
Address: Operational Base + offset(0x5C)
coreKit version ID register

bit	Attr	Reset Value	Description
31:0	R	0x3331302a	Contains the hex representation of the component version.

SPIM_DR
Address: Operational Base + offset(0x60-9C)
Data Register

bit	Attr	Reset Value	Description

15:0	RW	0x0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified.
			Read = Receive FIFO buffer
			Write = Transmit FIFO buffer

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

20.4 Functional Description

20.4.1 Operation

- **Operation Modes**

The Spi can be configured in the following two fundamental modes of operation: Master Mode, Slave Mode.In our design,SPI0 work as master mode with ID 0xffff_ffff and SPI1 work as slave mode with ID 0xffff_ffff1.

- **Transfer Modes**

The Spi operates in the following four modes when transferring data on the serial bus:

1.Transmit and Receive:

When TMOD = 2‘b00, both transmit and receive logic are valid.

2.Transmit Only:

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO.

3.Receive Only:

When TMOD = 2‘b10, the transmit data are invalid.

4.EEPROM Read:

When TMOD = 2‘b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). When the transmit FIFO becomes empty (all control information has been sent), data on the eceive line (rxd) is valid and is stored in the receive FIFO. The serial transfer continues until the number of data frames received by the Spi master matches the value of the NDF field in the CTRLR1 register + 1

- **Clock Ratios**

A summary of the frequency ratio restrictions between the bit-rate clock (sclk_out/sclk_in) and the Spi peripheral clock (ssi_clk) are described as:

Master: F_{ssi_clk} >= 2 × (maximum F_{sclk_out})
Slave (receive only): F_{ssi_clk} >= 6 × (maximum F_{sclk_in})
Slave: F_{ssi_clk} >= 8 × (maximum F_{sclk_in})

- **SPI Operation**

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 16 bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. Figure 2 shows a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPOL = 1.

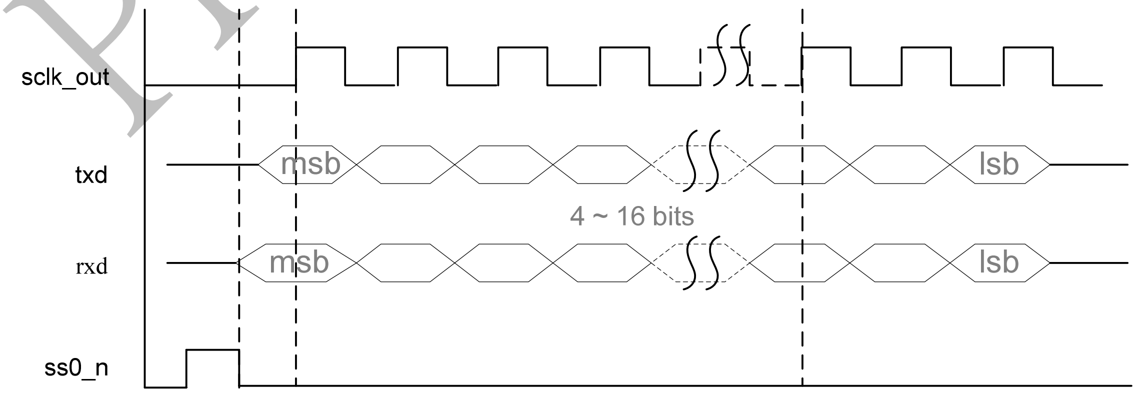


Figure 20-2 SPI Master Serial Format (SCPH = 0)

When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. Figure 3 shows the timing diagram for the SPI format when the configuration parameter SCPH = 1.

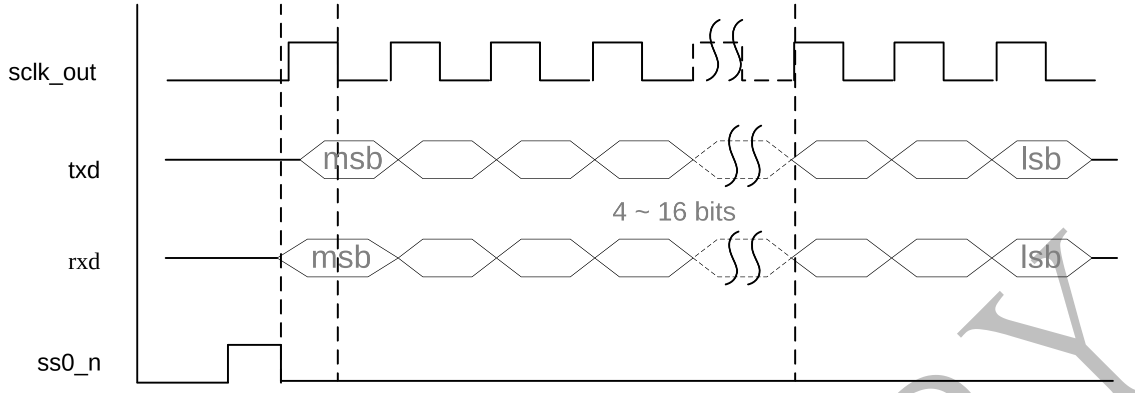


Figure 20-3 SPI Master Serial Format (SCPH = 1)

20.4.2 Programming sequence

- **Master Transfer Flow**

When configured as a serial-master device, the spi initiates and controls all serial transfers. The serial bit-rate clock, generated and controlled by the spi, is driven out on the sclk_out line. When the spi is disabled (SPI_EN = 0), no serial transfers can occur and sclk_out is held in "inactive" state, as defined by the serial protocol under which it operates.

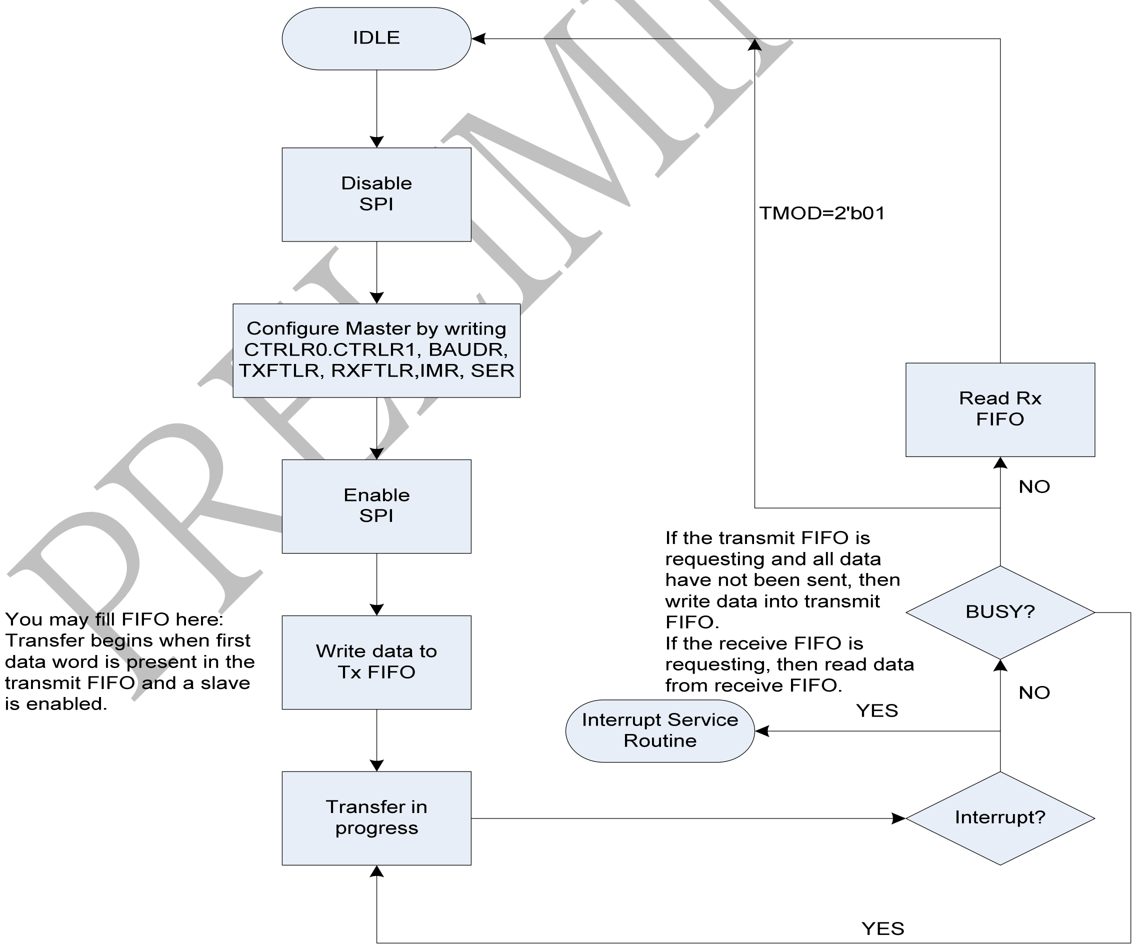

IDLE
Disable SPI
Configure Master by writing CTRLR0.CTRLR1, BAUDR, TXFTLR, RXFTLR,IMR, SER
Enable SPI
Write data to Tx FIFO
Transfer in progress
You may fill FIFO here: Transfer begins when first data word is present in the transmit FIFO and a slave is enabled.
TMOD=2'b01
Read Rx FIFO
NO
If the transmit FIFO is requesting and all data have not been sent, then write data into transmit FIFO.
If the receive FIFO is requesting, then read data from receive FIFO.
BUSY?
NO
YES
Interrupt Service Routine
Interrupt?
YES

Figure 20-4 SPI Master transfer flow

Chapter 21 SPI Slave Controller

21.1 Design Overview

21.1.1 Overview

The Serial Peripheral Interface (SPI) is an APB slave device. A four-wire, full-duplex serial protocol from Motorola. There are four possible combinations for the serial clock phase and polarity. The clock phase (SCPH) determines whether the serial transfer begins with the falling edge of the slave select signal or the first edge of the serial clock. The slave select line is held high when the spi is idle or disabled.SPI slave controller only work as slave mode.

21.1.2 Features

- AMBA APB interface – Allows for easy integration into a Synthesizable Components for AMBA 2 implementation.
- Serial-slave operation – Enables serial communication with serial-master peripheral devices.
- DMA Controller Interface – Enables interface to a DMA controller over the AMBA bus using a handshaking interface for transfer requests.
- Support interrupt interface to interrupt controller, and independently masking of interrupts.
- Dynamic control of the serial bit rate of the data transfer.
- Two 16x16 fifos for transferring and receiving use respectively.

21.2 Architecture

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

21.2.1 Block Diagram

The SPI comprises with:

- AMBA APB interface and DMA Controller Interface
- Transmit and receive FIFO controllers and an FSM controller
- Register block
- Shift control and interrupt

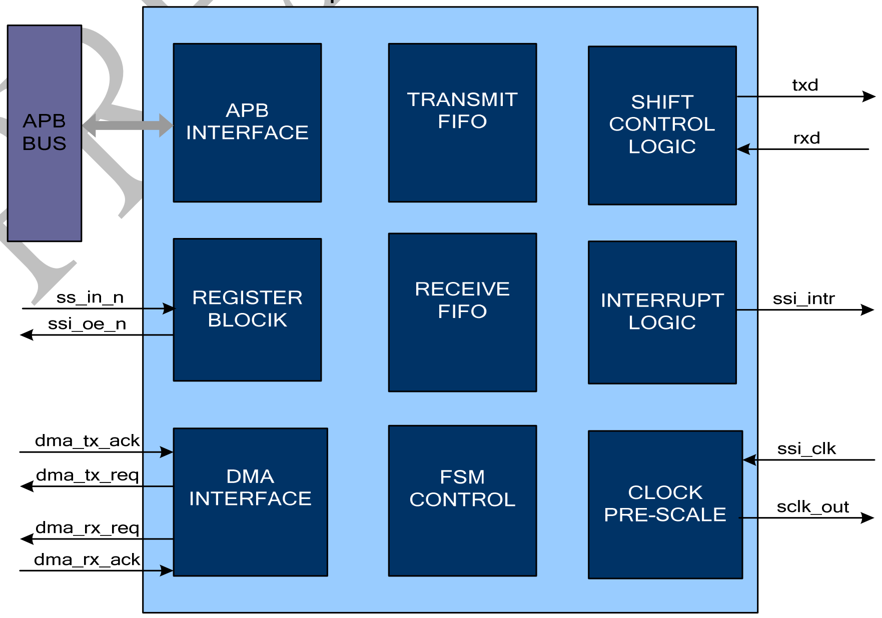


Figure 21-1 SPI Slave Controller Block diagram

21.2.2 Block Descriptions

APB INTERFACE

The host processor accesses data, control, and status information on the SPI through the APB interface. The SPI supports APB data bus widths of 8, 16, and 32 bits.

DMA INTERFACE

This block has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA Controller.

FIFO LOGIC

For transmit and receive transfers, data transmitted from the SPI to the external serial device is written into the transmit FIFO. Data received from the external serial device into the SPI is pushed into the receive FIFO. Both fifos are 16x16bits.

FSM CONTROL

Control the state's transformation of the design.

REGISTER BLOCK

All registers in the SPI are addressed at 32-bit boundaries to remain consistent with the AHB bus. Where the physical size of any register is less than 32-bits wide, the upper unused bits of the 32-bit boundary are reserved. Writing to these bits has no effect; reading from these bits returns 0.

SHIFT CONTROL

Shift control logic shift the data from the transmit fifo or to the receive fifo. This logic automatically right-justifies receive data in the receive FIFO buffer

INTERRUPT CONTROL

The SPI supports combined and individual interrupt requests, each of which can be masked. The combined interrupt request is the ORed result of all other SPI interrupts after masking.

21.3 Registers

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

21.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SPIS_CTRLR0	0x0000	W	0x0000_0007	Control register 0
SPIS_CTRLR1	0x0004	W	0x0000_0000	Control register 1
SPIS_SPIENR	0x0008	W	0x0000_0000	Ssi enable register
SPIS_MWCR	0x000C	W	0x0000_0000	Microwire control register
SPIS_TXFTLR	0x0018	W	0x0000_0000	Transmit FIFO Threshold Level
SPIS_RXFTLR	0x001c	W	0x0000_0000	Receive FIFO Threshold Level
SPIS_TXFLR	0x0020	W	0x0000_0000	Transmit FIFO Level Register
SPIS_RXFLR	0x0024	W	0x0000_0000	Receive FIFO Level Register
SPIS_SR	0x0028	W	0x0000_0006	Status Register
SPIS_IMR	0x002c	W	0x0000_001f	Interrupt Mask Register
SPIS_ISR	0x0030	W	0x0000_0000	Interrupt Status Register
SPIS_RISR	0x0034	W	0x0000_0000	Raw Interrupt Status Register
SPIS_TXOICR	0x0038	W	0x0000_0000	Transmit FIFO Overflow Interrupt Clear Register
SPIS_RXOICR	0x003c	W	0x0000_0000	Receive FIFO Overflow Interrupt Clear Register
SPIS_RXUICR	0x0040	W	0x0000_0000	Receive FIFO Underflow Interrupt Clear Register

SPIS_MSTICR	0x0044	W	0x0000_0000	Multi-Master Interrupt Clear Register
SPIS_ICR	0x0048	W	0x0000_0000	Interrupt Clear Register
SPIS_DMACR	0x004c	W	0x0000_0000	DMA Control Register
SPIS_DMATDLR	0x0050	W	0x0000_0000	DMA Transmit Data Level
SPIS_DMARDLR	0x0054	W	0x0000_0000	DMA Receive Data Level
SPIS_IDR	0x0058	W	0xffff_fff1	Identification Register
SPIS_COMP_VERSION	0x005c	W	0x3331_302a	coreKit version ID register
SPIS_DR	0x0060-9C	W	0x0000_0000	Data 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

SPIS_CTRL0
Address: Operational Base + offset(0x00)
Control register 0

bit	Attr	Reset Value	Description
15:12	RW	0X0	Control Frame Size. Selects the length of the control word for the Microwire frame format. For the field decode,
11	RW	0x0	Shift Register Loop. Used for testing purposes only. When internally active, connects the transmit shift register output to the receive shift register input. 0 – Normal Mode Operation 1 – Test Mode Operation
10	RW	0x0	Slave Output Enable.
9:8	RW	0x0	Transfer Mode. Selects the mode of transfer for serial communication. This field does not affect the transfer duplicity. Only indicates whether the receive or transmit data are valid. 00 –- Transmit & Receive 01 –- Transmit Only 10 –- Receive Only 11 –- EEPROM Read
7	RW	0x0	Serial Clock Polarity 0 – Inactive state of serial clock is low 1 – Inactive state of serial clock is high
6	RW	0x0	Serial Clock Phase 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit
5:4	RW	0x0	Frame Format. Selects which serial protocol transfers the data. 00 –- Motorola SPI 01 –- Texas Instruments SSP 10 –- National Semiconductors Microwire 11 –- Reserved
3:0	RW	0x7	Data Frame Size. Selects the data frame length . When the data frame size is programmed to be less than 16 bits, the receive data are automatically right-justified by the receive logic, with the upper bits of the receive FIFO zero-padded.

			You must right-justify transmit data before writing into the transmit FIFO. The transmit logic ignores the upper unused bits when transmitting the data. 0000: Reserved – undefined operation 0001: Reserved – undefined operation 0010: Reserved – undefined operation 0011: 4-bit serial data transfer 0100: 5-bit serial data transfer 0101: 6-bit serial data transfer 0110: 7-bit serial data transfer 0111: 8-bit serial data transfer 1000: 9-bit serial data transfer 1001: 10-bit serial data transfer 1010: 11-bit serial data transfer 1011: 12-bit serial data transfer 1100: 13-bit serial data transfer 1101: 14-bit serial data transfer 1110: 15-bit serial data transfer 1111: 16-bit serial data transfer

SPIS_CTRL1
Address: Operational Base + offset(0x04)
Control register 1

bit	Attr	Reset Value	Description
15:0	RW	0x0	Number of Data Frames. When TMOD = 10, this register field sets the number of data frames to be continuously received by the SPI. The Spi continues to receive serial data until the number of data frames received is equal to this register value plus 1, which enables you to receive up to 64 KB of data in a continuous transfer. When the SPI is configured as a serial slave, the transfer continues for as long as the slave is selected. Therefore, this register serves no purpose and is not present when the Spi is configured as a serial slave

SPIS_SPIENR
Address: Operational Base + offset(0x08)
Spi enable register

bit	Attr	Reset Value	Description
0	RW	0x0	SPI Enable. Enables and disables all Spi operations. When disabled, all serial transfers are halted immediately. Transmit and receive FIFO buffers are cleared when the device is disabled. It is impossible to program some of the Spi control registers when enabled. When disabled, the ssi_sleep output is set (after delay) to inform the system that it is safe to remove the ssi_clk, thus saving power consumption in the system.

SPIS_MWCR
Address: Operational Base + offset(0x0C)
Microwire control register

bit	Attr	Reset Value	Description
2	RW	0x0	Microwire Handshaking.
1	RW	0x0	Microwire Control.
0	RW	0x0	Microwire Transfer Mode.

SPIS_TXFTLR

Address: Operational Base + offset(0x18)

Transmit FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Transmit FIFO Threshold. Controls the level of entries (or below) at which the transmit FIFO controller triggers an interrupt.

SPIS_RXFTLR

Address: Operational Base + offset(0x1C)

Receive FIFO Threshold Level

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	RW	0x0	Receive FIFO Threshold. Controls the level of entries (or above) at which the receive FIFO controller triggers an interrupt.

SPIS_TXFLR

Address: Operational Base + offset(0x20)

Transmit FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

SPIS_RXFLR

Address: Operational Base + offset(0x24)

Receive FIFO Level Register

bit	Attr	Reset Value	Description
15:4	-	-	Reserved
3:0	R	0x0	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO

SPIS_SR

Address: Operational Base + offset(0x28)

Status Register

bit	Attr	Reset Value	Description
6	R	0x0	Data Collision Error. Relevant only when the Spi is configured as a master device. This bit is set if the Spi master is actively transmitting when another master selects this device as a slave. This informs the processor that the last data transfer was halted before completion. This bit is cleared when read. 0 – No error 1 – Transmit data collision error
5	R	0x0	Transmission Error. Set if the transmit FIFO is empty when a transfer is started. This bit can be set only when the Spi is configured as a slave device. Data from the previous transmission is resent on the txd line. This bit is cleared when read. 0 – No error 1 – Transmission error
4	R	0x0	Receive FIFO Full. When the receive FIFO is completely full, this bit is set. When thereceive FIFO contains one

			or more empty location, this bit is cleared. 0 – Receive FIFO is not full 1 – Receive FIFO is full
3	R	0x0	Receive FIFO Not Empty. Set when the receive FIFO contains one or more entries and is cleared when the receive FIFO is empty. This bit can be polled by software to completely empty the receive FIFO. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty
2	R	0x1	Transmit FIFO Empty. When the transmit FIFO is completely empty, this bit is set. When the transmit FIFO contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty
1	R	0x1	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full
0	R	0x0	SPI Busy Flag. When set, indicates that a serial transfer is in progress; when cleared indicates that the Spi is idle or disabled. 0 – Spi is idle or disabled 1 – Spi is actively transferring data

SPIS_IMR
Address: Operational Base + offset(0x2C)
Interrupt Mask Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	RW	0x1	Multi-Master Contention Interrupt Mask. This bit field is not present if the spi is configured as a serial-slave device. 0 – ssi_mst_intr interrupt is masked 1 – ssi_mst_intr interrupt is not masked
4	RW	0x1	Receive FIFO Full Interrupt Mask 0 – ssi_rxf_intr interrupt is masked 1 – ssi_rxf_intr interrupt is not masked
3	RW	0x1	Receive FIFO Overflow Interrupt Mask 0 – ssi_rxo_intr interrupt is masked 1 – ssi_rxo_intr interrupt is not masked
2	RW	0x1	Receive FIFO Underflow Interrupt Mask 0 – ssi_rxu_intr interrupt is masked 1 – ssi_rxu_intr interrupt is not masked
1	RW	0x1	Transmit FIFO Overflow Interrupt Mask 0 – ssi_txo_intr interrupt is masked 1 – ssi_txo_intr interrupt is not masked
0	RW	0x1	Transmit FIFO Empty Interrupt Mask 0 – ssi_txe_intr interrupt is masked 1 – ssi_txe_intr interrupt is not masked

SPIS_ISR
Address: Operational Base + offset(0x30)
Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved

5	R	0x0	Multi-Master Contention Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt not active after masking 1 = ssi_mst_intr interrupt is active after masking
4	R	0x0	Receive FIFO Full Interrupt Status 0 = ssi_rxf_intr interrupt is not active after masking 1 = ssi_rxf_intr interrupt is full after masking
3	R	0x0	Receive FIFO Overflow Interrupt Status 0 = ssi_rxo_intr interrupt is not active after masking 1 = ssi_rxo_intr interrupt is active after masking
2	R	0x0	Receive FIFO Underflow Interrupt Status 0 = ssi_rxu_intr interrupt is not active after masking 1 = ssi_rxu_intr interrupt is active after masking
1	R	0x0	Transmit FIFO Overflow Interrupt Status 0 = ssi_txo_intr interrupt is not active after masking 1 = ssi_txo_intr interrupt is active after masking
0	R	0x0	Transmit FIFO Empty Interrupt Status 0 = ssi_txe_intr interrupt is not active after masking 1 = ssi_txe_intr interrupt is active after masking

SPIS_RISR
Address: Operational Base + offset(0x34)
Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:6	-	-	Reserved
5	R	0x0	Multi-Master Contention Raw Interrupt Status. This bit field is not present if the Spi is configured as a serial-slave device. 0 = ssi_mst_intr interrupt is not active prior to masking 1 = ssi_mst_intr interrupt is active prior masking
4	R	0x0	Receive FIFO Full Raw Interrupt Status 0 = ssi_rxf_intr interrupt is not active prior to masking 1 = ssi_rxf_intr interrupt is active prior to masking
3	R	0x0	Receive FIFO Overflow Raw Interrupt Status 0 = ssi_rxo_intr interrupt is not active prior to masking 1 = ssi_rxo_intr interrupt is active prior masking
2	R	0x0	Receive FIFO Underflow Raw Interrupt Status 0 = ssi_rxu_intr interrupt is not active prior to masking 1 = ssi_rxu_intr interrupt is active prior to masking
1	R	0x0	Transmit FIFO Overflow Raw Interrupt Status 0 = ssi_txo_intr interrupt is not active prior to masking 1 = ssi_txo_intr interrupt is active prior masking
0	R	0x0	Transmit FIFO Empty Raw Interrupt Status 0 = ssi_txe_intr interrupt is not active prior to masking 1 = ssi_txe_intr interrupt is active prior masking

SPIS_TXOICR
Address: Operational Base + offset(0x38)
Transmit FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Transmit FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_txo_intr interrupt; writing has no effect.

SPIS_RXOICR
Address: Operational Base + offset(0x3C)
Receive FIFO Overflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Overflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxo_intr interrupt; writing has no effect.

SPIS_RXUICR
Address: Operational Base + offset(0x40)
Receive FIFO Underflow Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Receive FIFO Underflow Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_rxu_intr interrupt; writing has no effect

SPIS_MSTICR
Address: Operational Base + offset(0x44)
Multi-Master Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Multi-Master Contention Interrupt. This register reflects the status of the interrupt. A read from this register clears the ssi_mst_intr interrupt; writing has no effect.

SPIS_ICR
Address: Operational Base + offset(0x48)
Interrupt Clear Register

bit	Attr	Reset Value	Description
0	R	0x0	Clear Interrupts. This register is set if any of the interrupts below are active. A read clears the ssi_txo_intr, ssi_rxu_intr, ssi_rxo_intr, and the ssi_mst_intr interrupts. Writing to this register has no effect.

SPIS_DMACR
Address: Operational Base + offset(0x4C)
DMA Control Register

bit	Attr	Reset Value	Description
1	RW	0x0	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
0	RW	0x0	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel 0 = Receive DMA disabled 1 = Receive DMA enabled

SPIS_DMATDLR
Address: Operational Base + offset(0x50)
DMA Transmit Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It

			is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMAE = 1.

SPIS_DMARDLR
Address: Operational Base + offset(0x54)
DMA Receive Data Level

bit	Attr	Reset Value	Description
3:0	RW	0x0	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or above this field value + 1, and RDMAE=1.

SPIS_IDR
Address: Operational Base + offset(0x58)
Identification Register

bit	Attr	Reset Value	Description
31:0	R	0xffffffff	Identification Code.

SPIS_COMP_VERSION
Address: Operational Base + offset(0x5C)
coreKit version ID register

bit	Attr	Reset Value	Description
31:0	R	0x3331302a	Contains the hex representation of the component version.

SPIS_DR
Address: Operational Base + offset(0x60-9C)
Data Register

bit	Attr	Reset Value	Description

15:0	RW	0x0	Data Register. When writing to this register, you must right-justify the data. Read data are automatically right-justified.
			Read = Receive FIFO buffer
			Write = Transmit FIFO buffer

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

21.4 Functional Description

21.4.1 Operation

◆ **Operation Modes**

The Spi can be configured in the following two fundamental modes of operation: Master Mode, Slave Mode.In our design,SPI0 work as master mode with ID 0xffff_ffff and SPI1 work as slave mode with ID 0xffff_ffff1.

◆ **Transfer Modes**

The Spi operates in the following four modes when transferring data on the serial bus:

1.Transmit and Receive:

When TMOD = 2‘b00, both transmit and receive logic are valid.

2.Transmit Only:

When TMOD = 2‘b01, the receive data are invalid and should not be stored in the receive FIFO.

3.Receive Only:

When TMOD = 2‘b10, the transmit data are invalid.

4.EEPROM Read:

When TMOD = 2‘b11, the transmit data is used to transmit an opcode and/or an address to the EEPROM device. Typically this takes three data frames (8-bit opcode followed by 8-bit upper address and 8-bit lower address). When the transmit FIFO becomes empty (all control information has been sent), data on the eceive line (rxd) is valid and is stored in the receive FIFO. The serial transfer continues until the number of data frames received by the Spi master matches the value of the NDF field in the CTRLR1 register + 1

- **Clock Ratios**

A summary of the frequency ratio restrictions between the bit-rate clock (sclk_out/sclk_in) and the Spi peripheral clock (ssi_clk) are described as:

Master: F_{ssi_clk} >= 2 × (maximum F_{sclk_out})

Slave (receive only): F_{ssi_clk} >= 6 × (maximum F_{sclk_in})

Slave: F_{ssi_clk} >= 8 × (maximum F_{sclk_in})

- **SPI Operation**

With the SPI, the clock polarity (SCPOL) configuration parameter determines whether the inactive state of the serial clock is high or low. To transmit data, both SPI peripherals must have identical serial clock phase (SCPH) and clock polarity (SCPOL) values. The data frame can be 4 to 16 bits in length.

When the configuration parameter SCPH = 0, data transmission begins on the falling edge of the slave select signal. The first data bit is captured by the master and slave peripherals on the first edge of the serial clock; therefore, valid data must be present on the txd and rxd lines prior to the first serial clock edge. Figure 2 shows a timing diagram for a single SPI data transfer with SCPH = 0. The serial clock is shown for configuration parameters SCPOL = 0 and SCPOL = 1.

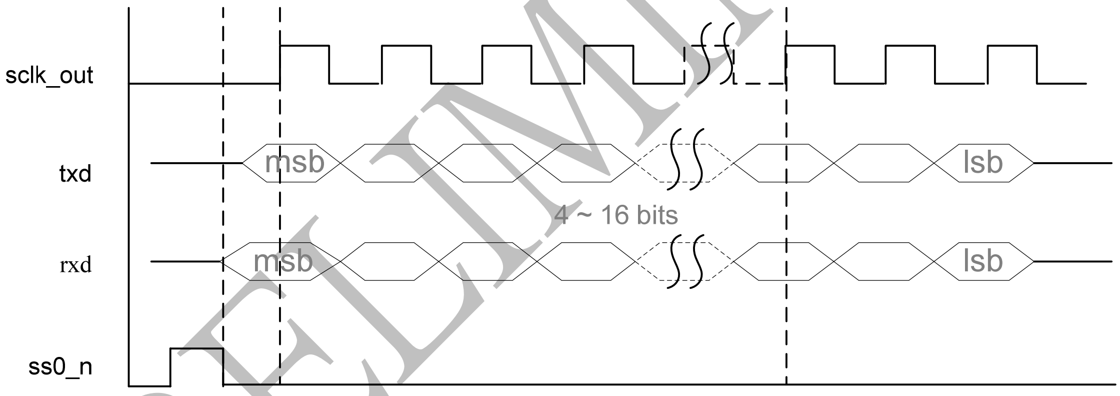


Figure 21-2 SPI Slave Serial Format (SCPH = 0)

When the configuration parameter SCPH = 1, both master and slave peripherals begin transmitting data on the first serial clock edge after the slave select line is activated. The first data bit is captured on the second (trailing) serial clock edge. Data are propagated by the master and slave peripherals on the leading edge of the serial clock. During continuous data frame transfers, the slave select line may be held active-low until the last bit of the last frame has been captured. Figure 3 shows the timing diagram for the SPI format when the configuration parameter SCPH = 1.

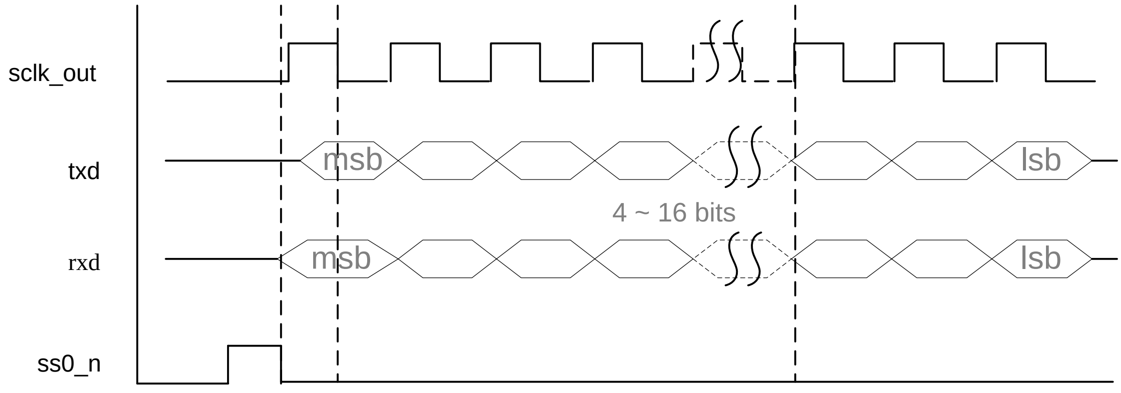


Figure 21-3 SPI Slave Serial Format (SCPH = 1)

21.4.2 Programming sequence

- **Slave Transfer Flow**

When the SPI is configured as a slave device, all serial transfers are initiated and controlled by the serial bus master.

When the SPI serial slave is selected during configuration, it enables its txd data onto the serial bus. All data transfers to and from the serial slave are regulated on the serial clock line (sclk_in), driven from the serial-master device. Data are propagated from the serial slave on one edge of the serial clock line and sampled on the opposite edge.

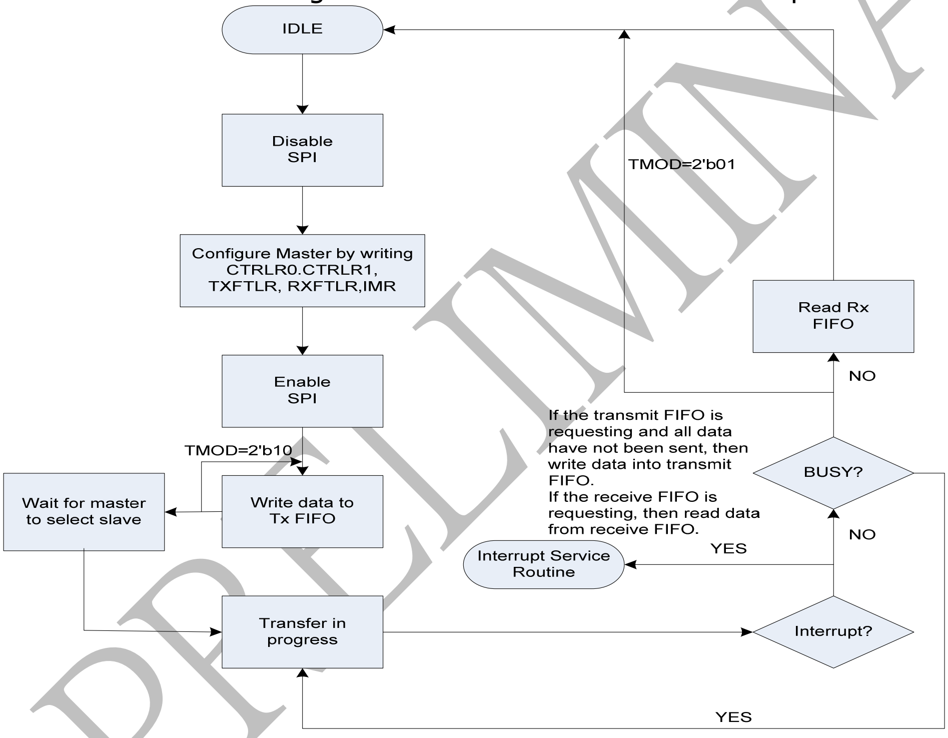

IDLE
Disable SPI
Configure Master by writing CTRLR0.CTRLR1, TXFTLR, RXFTLR,IMR
Enable SPI
TMOD=2'b10
Write data to Tx FIFO
Wait for master to select slave
Transfer in progress
Interrupt?
YES
Interrupt Service Routine
If the transmit FIFO is requesting and all data have not been sent, then write data into transmit FIFO.
If the receive FIFO is requesting, then read data from receive FIFO.
NO
BUSY?
NO
Read Rx FIFO
TMOD=2'b01
YES

Figure 21-4 SPI Slave transfer flow

Chapter 22 Timers in CPU system

22.1 Design Overview

22.1.1 Overview

Timers is a programmable timers peripheral. This component is an APB slave device. Timers count down from a programmed value and generate an interrupt when the count reaches zero.

22.1.2 Features

- Three programmable 32 bits timers
- Two operation modes: free-running and user-defined count
- The clock of all timers is pclk
- Maskable for each individual interrupt

22.2 Architecture

22.2.1 Block Diagram

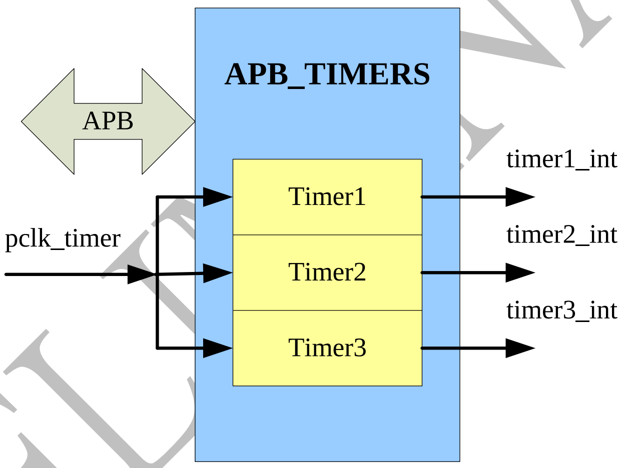


Fig. 22-1 Timers Block Diagram in CPU System

22.3 Registers

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

22.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
Timer1LoadCount	0x0000	W	0x00000000	Timer1 Load Count Register
Timer1CurrentValue	0x0004	W	0x00000000	Timer1 Current Value Register
Timer1ControlReg	0x0008	W	0x00000000	Timer1 Control Register
Timer1EOI	0x000C	W	0x00000000	Timer1 End-of-Interrupt Register
Timer1IntStatus	0x0010	W	0x00000000	Timer1 Interrupt Status Register
Timer2LoadCount	0x0014	W	0x00000000	Timer2 Load Count Register
Timer2CurrentValue	0x0018	W	0x00000000	Timer2 Current Value Register
Timer2ControlReg	0x001c	W	0x00000000	Timer2 Control Register

Timer2EOI	0x0020	W	0x00000000	Timer2 End-of-Interrupt Register
Timer2IntStatus	0x0024	W	0x00000000	Timer2 Interrupt Status Register
Timer3LoadCount	0x0028	W	0x00000000	Timer3 Load Count Register
Timer3CurrentValue	0x002c	W	0x00000000	Timer3 Current Value Register
Timer3ControlReg	0x0030	W	0x00000000	Timer3 Control Register
Timer3EOI	0x0034	W	0x00000000	Timer3 End-of-Interrupt Register
Timer3IntStatus	0x0038	W	0x00000000	Timer3 Interrupt Status Register
TimersIntStatus	0x00a0	W	0x00000000	Timers Interrupt Status Register
TimersEOI	0x00a4	W	0x00000000	Timers End-of-Interrupt Register
TimersRawIntStatus	0x00a8	W	0x00000000	Timers Raw Interrupt Status 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

Timer1LoadCount
Address: Operational Base + offset(0x00)
Timer1 Load Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Value to be loaded into Timer1. This is the value from which counting commences.

Timer1CurrentValue
Address: Operational Base + offset(0x04)
Timer1 Current Value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000	Current Value of Timer1.

Timer1ControlReg
Address: Operational Base + offset(0x08)
Timer1 Control Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 0: not mask 1: mask
1	RW	0x0	Timer mode. 0: free-running mode 1: user-defined count mode
0	RW	0x0	Timer enable. 0: disable 1: enable

Timer1EOI
Address: Operational Base + offset(0x0C)
Timer1 End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved

0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from timer1

Timer1IntStatus

Address: Operational Base + offset(0x10)

Timer1 Interrupt Status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register contains the interrupt status for timer1

Timer2LoadCount

Address: Operational Base + offset(0x14)

Timer2 Load Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Value to be loaded into Timer2. This is the value from which counting commences.

Timer2CurrentValue

Address: Operational Base + offset(0x18)

Timer2 Current Value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000	Current Value of Timer2.

Timer2ControlReg

Address: Operational Base + offset (0x1c)

Timer2 Control Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 0: not mask 1: mask
1	RW	0x0	Timer mode. 0: free-running mode 1: user-defined count mode
0	RW	0x0	Timer enable. 0: disable 1: enable

Timer2EOI

Address: Operational Base + offset(0x20)

Timer2 End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from timer2

Timer2IntStatus

Address: Operational Base + offset(0x24)

Timer2 Interrupt Status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register contains the interrupt status for timer2

Timer3LoadCount

Address: Operational Base + offset(0x28)

Timer3 Load Count Register

bit	Attr	Reset Value	Description
31:0	RW	0x0000	Value to be loaded into Timer3. This is the value from which counting commences.

Timer3CurrentValue

Address: Operational Base + offset(0x2c)
Timer3 Current Value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000	Current Value of Timer3.

Timer3ControlReg

Address: Operational Base + offset(0x30)
Timer3 Control Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 0: not mask 1: mask
1	RW	0x0	Timer mode. 0: free-running mode 1: user-defined count mode
0	RW	0x0	Timer enable. 0: disable 1: enable

Timer3EOI

Address: Operational Base + offset(0x34)
Timer3 End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Reading from this register returns all zeros(0) and clear interrupt from timer3

Timer3IntStatus

Address: Operational Base + offset(0x38)
Timer3 Interrupt Status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register contains the interrupt status for timer3

TimersIntStatus

Address: Operational Base + offset(0xa0)
Timers Interrupt Status Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	This register contains the interrupt status for timer3
1	R	0x0	This register contains the interrupt status for timer2
0	R	0x0	This register contains the interrupt status for timer1

TimersEOI

Address: Operational Base + offset(0xa4)
Timers End-of-Interrupt Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2:0	R	0x0	Reading from this register returns all zeros(0) and

			clear interrupt from all timers

TimersRawIntStatus
Address: Operational Base + offset(0xa8)
Timers Raw Interrupt Status Register

bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	R	0x0	This register contains the interrupt status for timer3 prior to masking
1	R	0x0	This register contains the interrupt status for timer2 prior to masking
0	R	0x0	This register contains the interrupt status for timer1 prior to masking

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

22.4 Functional Description

22.4.1 Operation

Timer operation

Timers count down from a programmed value and generate an interrupt when the count reaches zero.

The input clock for each timer is pclk_timer from SCU block(Chapter16).

The initial value for each timer – that is, the value from which it counts down – is loaded using the appropriate load count register (TimerNLoadCount). Two events can cause a timer to load the initial count from its TimerNLoadCount register:

- Timer is enabled after being reset or disabled
- Timer counts down to 0

All interrupt status registers and end-of-interrupt registers can be accessed at any time.

Configuaration

Timers contain three identical but separately-programmable timers, which are accessed through a single AMBA APB interface.

22.4.2 Programming sequence

1. Initialize the timer through the TimerNControlReg register (where N is in the range 1 to 3):

a. Disable the timer by writing a "0" to the timer enable bit (bit 0); accordingly, the timer_en output signal is de-asserted.
b. Program the timer mode—user-defined or free-running—by writing a "0" or "1," respectively, to the timer mode bit (bit 1).
c. Set the interrupt mask as either masked or not masked by writing a "0" or "1," respectively, to the timer interrupt mask bit (bit 2).

2. Load the timer counter value into the TimerNLoadCount register (where N is in the range 1 to 3).

3. Enable the timer by writing a "1" to bit 0 of TimerNControlReg.

Timers Ustage flow

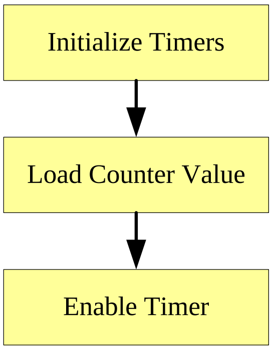


Fig. 22-2 Timers Ustage Flow in CPU System

Chapter 23 Timers in DSP system

23.1 Design Overview

23.1.1 Overview

The TIMER unit groups together two TIMER modules, hereafter TIMER_UNIT_0 and TIMER_UNIT_1, which can function as either two independent timers or as a single cascaded timer.

Each TIMER unit module is designed to count clock cycles or event triggers, and each can operate as a periodic interrupt generator, event counter or pulse-width modulator. TIMER_UNIT_0 can be allocated for a Watchdog function. Each TIMER unit counter operates by either an internal XAPB domain clock source or an external clock source. The clock source for TIMER_UNIT_1 can be configured to be the end-of-count pulse for TIMER_UNIT_0 in a cascaded mode. The input clock source can be locally divided to suit the timing requirements.

23.1.2 Features

Each TIMER unit has the following main features:

- Various counting modes including:
  - Single Count mode
  - Auto-restart mode
  - Free-running mode
  - Event Count mode
  - Watchdog Timer mode
- Pulse Width Modulation (PWM) mechanism
- Pause capability
- Clock divider
- Three possible input clock signals - internal, external and cascaded
- Interrupt request generation at the end of counting
- Debug support, including an Automatic Pause in DSP Breakpoint mechanism
- Reset request at the end of counting
- XAPB interface to the programming module

DSP timer is only used in DSP sub-system normally. For detailed information about DSP timer, please refer to **RK28xx DSP sub-system.pdf.**

Chapter 24 Watchdog Timer (WDT)

24.1 Design Overview

24.1.1 Overview

Watchdog Timer (WDT) is an APB slave peripheral that can be used to prevent system lockup that may becaused by conflicting parts or programs in an SoC. The WDT would generated interrupt or reset signal when it's counter reaches zero, then a reset controller would reset the system.

24.1.2 Features

- 32 bits APB bus width
- WDT counter's clock is pclk
- 32 bits WDT counter width
- Counter counts down from a preset value to 0 to indicate the occurrence of a timeout
- WDT can perform two types of operations when timeout occurs:
- (1) Generate a system reset;
- (2) First generate an interrupt and if this is not cleared by the service routine by the time a second timeout occurs then generate a system reset
- Programmable reset pulse length
- Total 16 defined-ranges of main timeout period

24.2 Architecture

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

24.2.1 Block Diagram

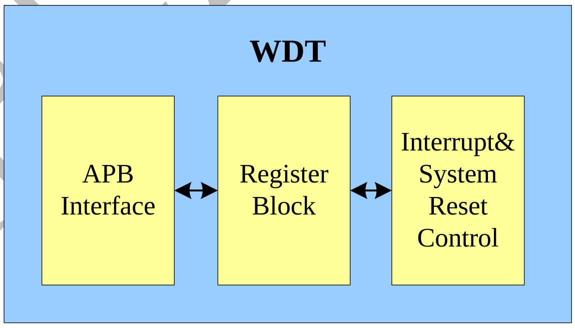


Fig. 24-1 WDT Block Diagram

24.2.2 Block Descriptions

APB Interface

The APB Interface implements the APB slave operation. It's bus width is 32 bits.

Register Block

A register block with read coherency for the current count registers.

Interrupt & system reset control

An interrupt/system reset generation block comprising of a decrementing counter and control logic.

24.3 Registers

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

24.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
WDT_CR	0x0000	W	0x0000000a	Control Register
WDT_TORR	0x0004	W	0x00000000	Timeout range Register
WDT_CCVR	0x0008	W	0x0000ffff	Current counter value Register
WDT_CRR	0x000C	W	0x00000000	Counter restart Register
WDT_STAT	0x0010	W	0x00000000	Interrupt status Register
WDT_EOI	0x0014	W	0x00000000	Interrupt clear Register

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

24.3.2 Detail Register Description

WDT_CR
Address: Operational Base + offset(0x00)
Control Register

bit	Attr	Reset Value	Description
31:5	-	-	Reserved
4:2	RW	0x2	Reset pulse length. This is used to select the number of pclk cycles for which the system reset stays asserted. 000: 2 pclk cycles 001: 4 pclk cycles 010: 8 pclk cycles 011: 16 pclk cycles 100: 32 pclk cycles 101: 64 pclk cycles 110: 128 pclk cycles 111: 256 pclk cycles
1	RW	0x1	Response mode. Selects the output response generated to a timeout. 0: Generate a system reset. 1: First generate an interrupt and if it is not cleared by the time a second timeout occurs then generate a system reset.
0	RW	0x0	WDT enable. 0: WDT disabled. 1: WDT enabled.

WDT_TORR
Address: Operational Base + offset(0x04)
Timeout range Register

bit	Attr	Reset Value	Description
31:4	-	-	Reserved
3:0	RW	0x0	Timeout period. This field is used to select the timeout period from which the watchdog counter restarts. A change of the timeout period takes effect only after the next counter restart (kick).

			The range of values available for a 32-bit watchdog counter are: 0000: 0x0000ffff 0001: 0x0001ffff 0010: 0x0003ffff 0011: 0x0007ffff 0100: 0x000fffff 0101: 0x001fffff 0110: 0x003fffff 0111: 0x007fffff 1000: 0x00ffffff 1001: 0x01ffffff 1010: 0x03ffffff 1011: 0x07ffffff 1100: 0x0fffffff 1101: 0x1fffffff 1110: 0x3fffffff 1111: 0x7fffffff

WDT_CCVR

Address: Operational Base + offset(0x08)
Current counter value Register

bit	Attr	Reset Value	Description
31:0	R	0x0000ffff	This register, when read, is the current value of the internal counter. This value is read coherently when ever it is read

WDT_CRR

Address: Operational Base + offset(0x0C)
Counter restart Register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	This register is used to restart the WDT counter. As a safety feature to prevent accidental restarts, the value 0x76 must be written. A restart also clears the WDT interrupt. Reading this register returns zero.

WDT_STAT

Address: Operational Base + offset(0x10)
Interrupt status Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	This register shows the interrupt status of the WDT. 1: Interrupt is active regardless of polarity. 0: Interrupt is inactive.

WDT_EOI

Address: Operational Base + offset(0x14)
Interrupt clear Register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	R	0x0	Clears the watchdog interrupt. This can be used to clear the interrupt without restarting the watchdog counter.

24.4 Functional Description

24.4.1 Operation

Counter

The WDT counts from a preset (timeout) value in descending order to zero. When the counter reaches zero, depending on the output response mode selected, either a system reset or an interrupt occurs. When the counter reaches zero, it wraps to the selected timeout value and continues decrementing. The user can restart the counter to its initial value. This is programmed by writing to the restart register at any time. The process of restarting the watchdog counter is sometimes referred to as kicking the dog. As a safety feature to prevent accidental restarts, the value 0x76 must be written to the Current Counter Value Register (WDT_CRR).

Interrupts

The WDT can be programmed to generate an interrupt (and then a system reset) when a timeout occurs. When a 1 is written to the response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT_CR), the WDT generates an interrupt. If it is not cleared by the time a second timeout occurs, then it generates a system reset. If a restart occurs at the same time the watchdog counter reaches zero, an interrupt is not generated.

System Resets

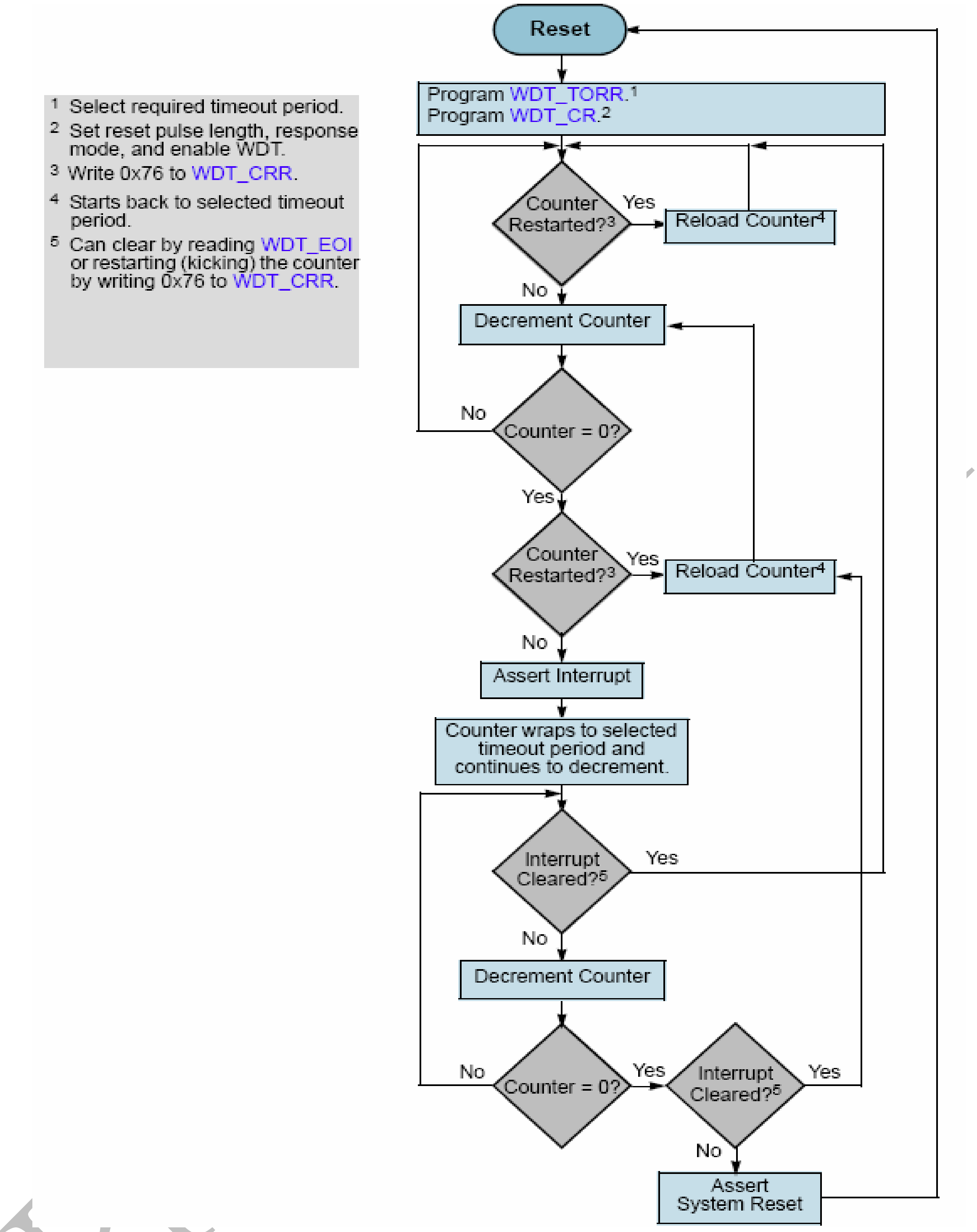
When a 0 is written to the output response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT_CR), the WDT generates a system reset when a timeout occurs.

Reset Pulse Length

The reset pulse length is the number of pclk cycles for which a system reset is asserted. When a system reset is generated, it remains asserted for the number of cycles specified by the reset pulse length or until the system is reset. A counter restart has no effect on the system reset once it has been asserted.

24.4.2 Programming sequence

Operation Flow Chart (Response mode=1)



[1] Select required timeout period.

[2] Set reset pulse length, response mode, and enable WDT.

[3] Write 0x76 to WDT_CRR.

[4] Starts back to selected timeout period.

[5] Can clear by reading WDT_EOI or restarting (kicking) the counter by writing 0x76 to WDT_CRR.

Fig. 24-2 WDT Operation Flow

Chapter 25 Real Time Clock (RTC)

25.1 Design Overview

25.1.1 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 . 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 and IO power is isolated from system power, so the RTC counter can running continuously during system power off.

25.1.2 Features

- 24 hour time mode with highest precision of one sixteenth of a second
- Programmable alarm with interrupt generation
- Maskable 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 core power loss indication

25.2 Architecture

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

25.2.1 Block Diagram

The RTC comprises with:

- APB_RTC_REG - RTC/alarm registers with APB slave interface
- Clock divider for 1Hz and counter
- Internal power off Controller

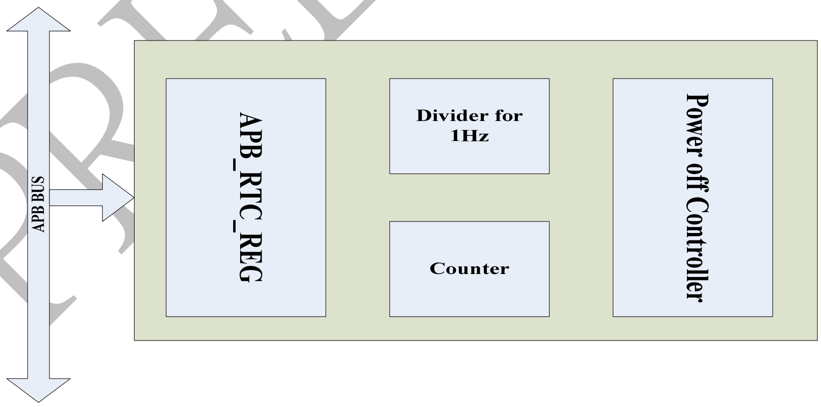


Fig. 25-1 RTC design architecture

For detailed information about RTC, please refer to **RK28xx Real Time Clock.pdf**。

Chapter 26 I2C Controller

26.1 Design Overview

26.1.1 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.

26.1.2 Features

- 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

26.2 Architecture

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

26.2.1 Block Diagram

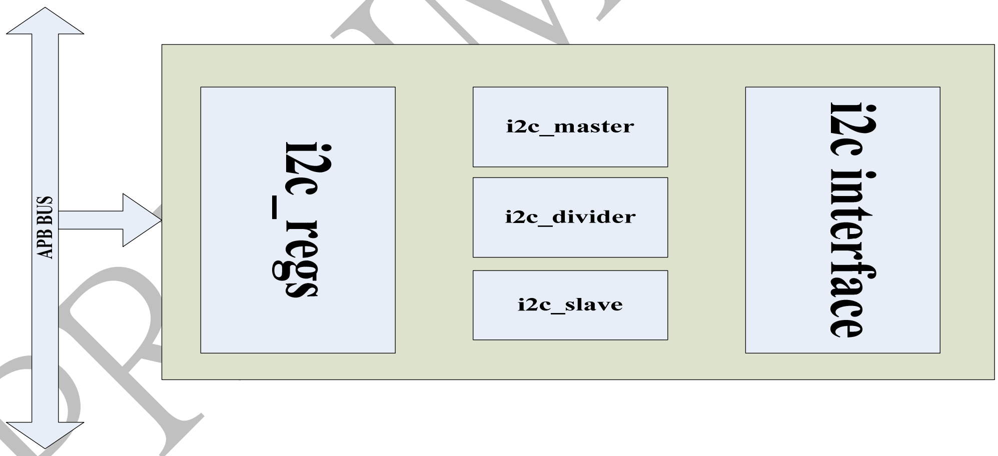


Fig. 26-1 I2C Controller design architecture

26.2.2 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

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.

26.3 Registers

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

26.3.1 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

26.3.2 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

26.4 Functional Description

26.4.1 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.

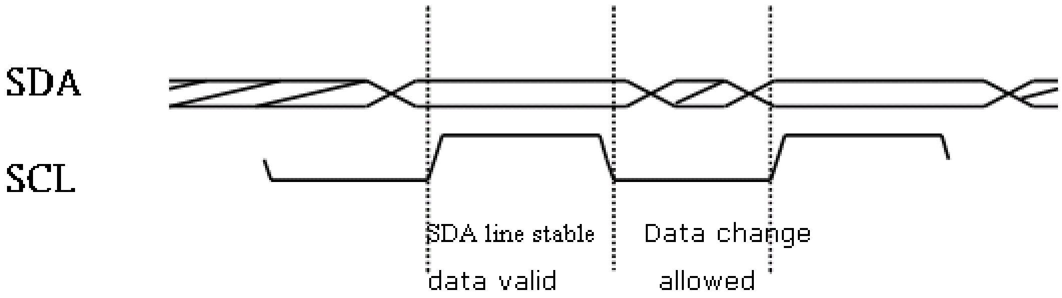
Abnormal stop occurs (bit 6): The bit is asserted when Slave receive abnormal stop.

I2C controller data transfer waveform

- **Bit transferring**

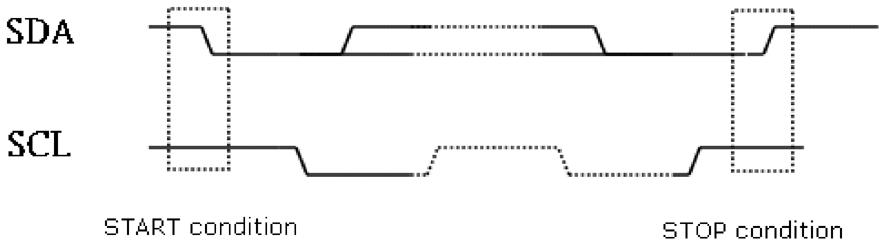
(a) Data Validity

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

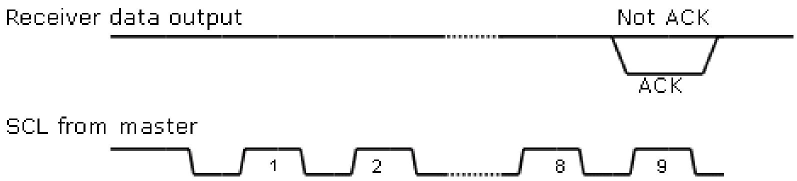
START condition occurs when SDA goes low while SCL is in high period. STOP condition is generated when SDA line goes high while SCL is in high state.



- **Data transfer**

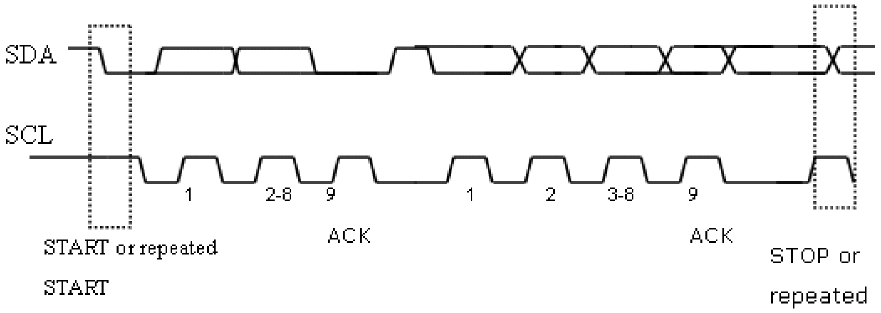
(a) Acknowledge

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".



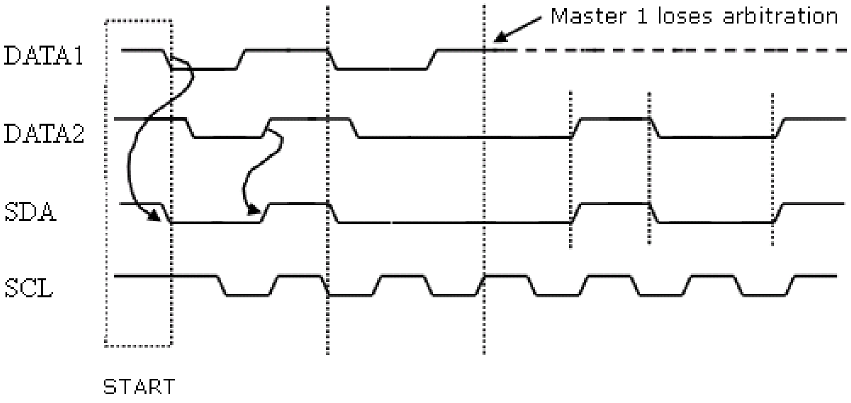
(b) Byte transfer

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.

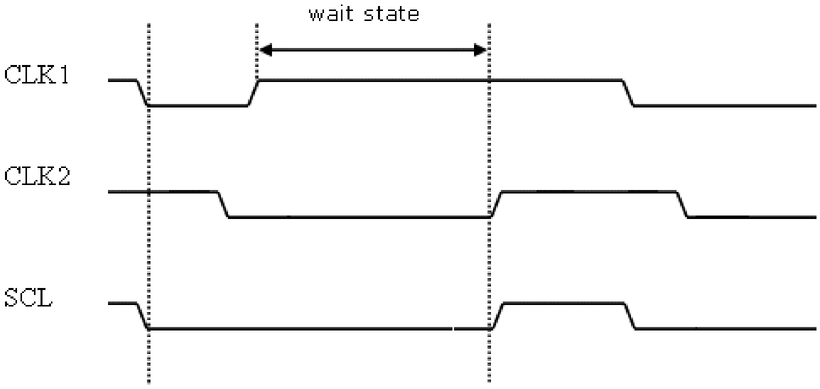


(c) Arbitration

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.



26.4.2 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

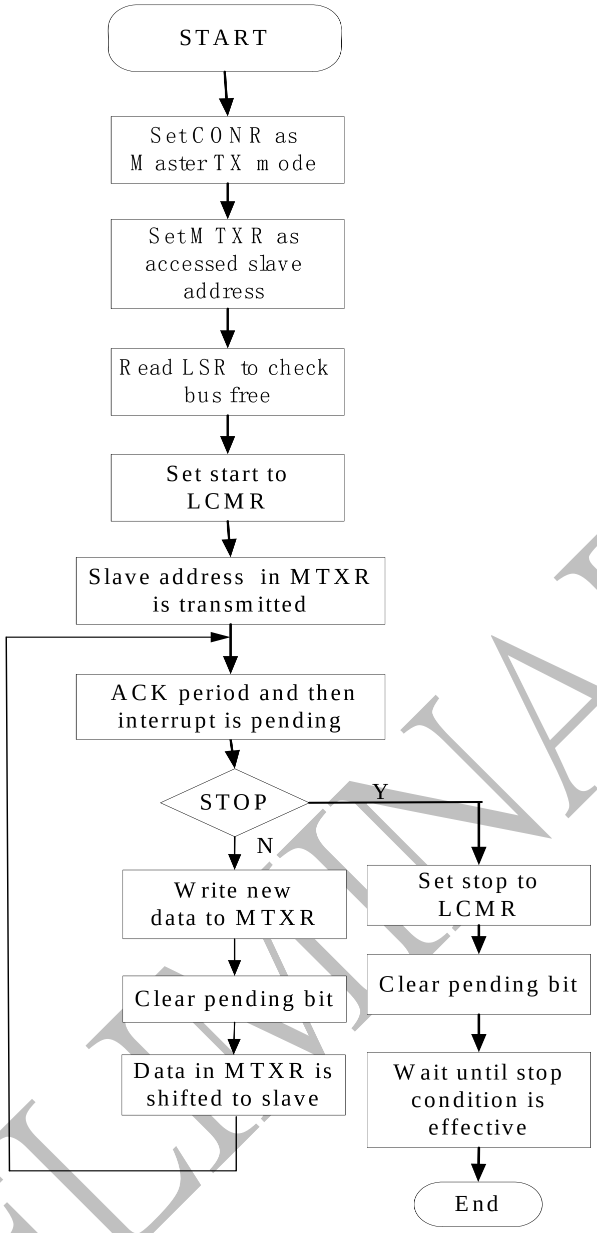


Fig. 26-2 I2C controller operation flow in Master/Transmiter mode

Operations for Master/Receiver Mode

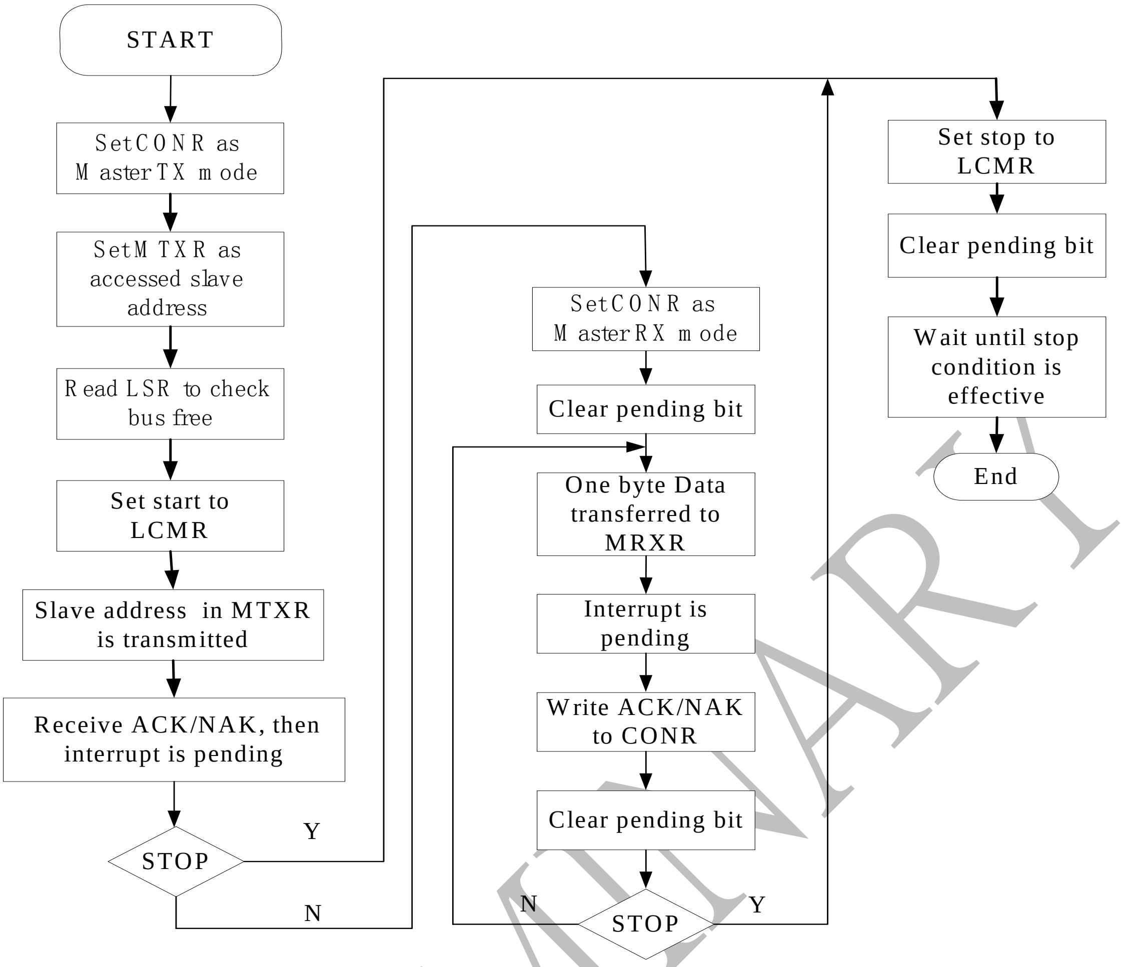

START
SetCONR as MasterTX mode
SetMTXR as accessed slave address
Read LSR to check bus free
Set start to LCMR
Slave address in MTXR is transmitted
Receive ACK/NAK, then interrupt is pending
STOP
Y
N
SetCONR as MasterRX mode
Clear pending bit
One byte Data transferred to MRXR
Interrupt is pending
Write ACK/NAK to CONR
Clear pending bit
STOP
N
Y
Set stop to LCMR
Clear pending bit
Wait until stop condition is effective
End

Fig. 26-3 I2C controller operation flow in Master/Receiver mode

Operations for Slave/Transmitter Mode

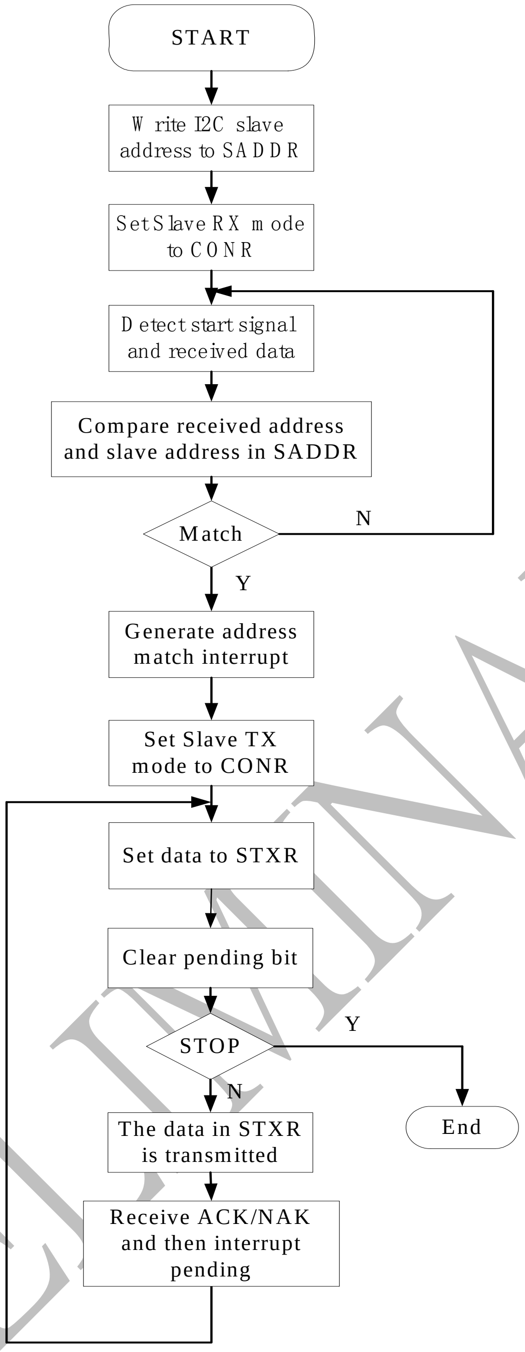

START
Write I2C slave address to SADDR
Set Slave RX mode to CONR
Detect start signal and received data
Compare received address and slave address in SADDR
Match
N
Y
Generate address match interrupt
Set Slave TX mode to CONR
Set data to STXR
Clear pending bit
STOP
Y
N
End
The data in STXR is transmitted
Receive ACK/NAK and then interrupt pending

Fig. 26-4 I2C controller operation flow in Slave/Transmiter mode

Operations for Slave/Receiver Mode

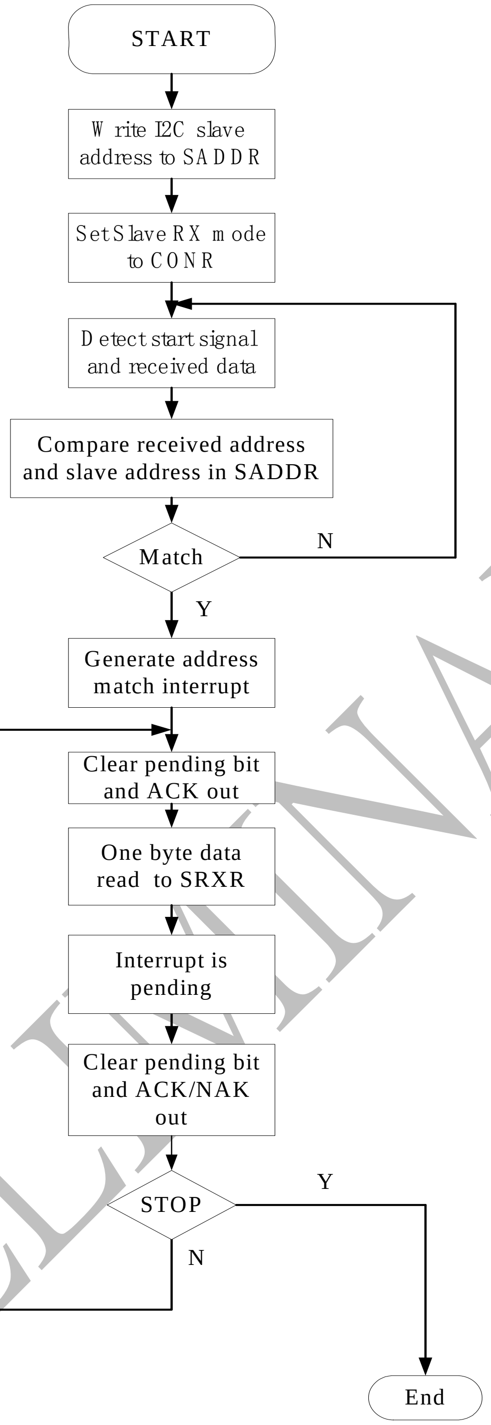

START
Write I2C slave address to SADDR
Set Slave RX mode to CONR
Detect start signal and received data
Compare received address and slave address in SADDR
Match
N
Y
Generate address match interrupt
Clear pending bit and ACK out
One byte data read to SRXR
Interrupt is pending
Clear pending bit and ACK/NAK out
STOP
Y
N
End

Fig. 26-5 I2C controller operation flow in Slave/Receiver mode

Chapter 27 I2S Controller

27.1 Design Overview

27.1.1 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.

27.1.2 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.

27.2 Architecture

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

27.2.1 Block Diagram

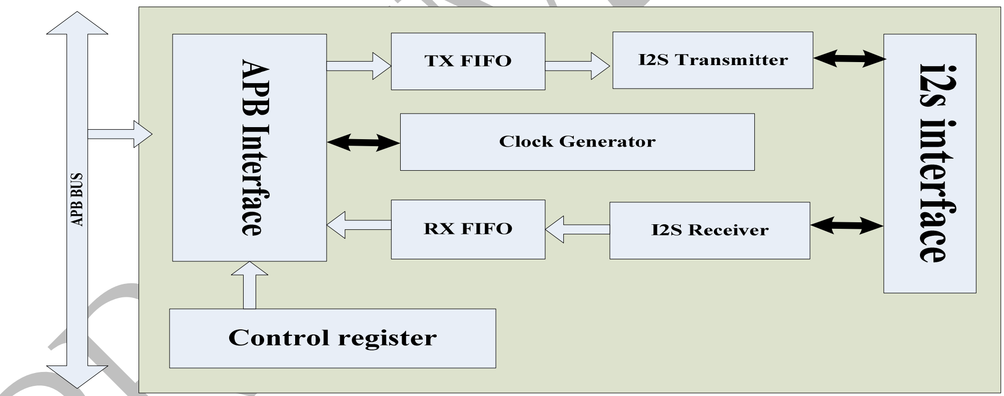


Fig. 27-1 I2S controller design architecture

27.2.2 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.

27.3 Registers

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

27.3.1 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.

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

27.3.2 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
31:20	-	-	Reserved
19:18	RW	0x0	Transmitter devices select. This value is used to select which device of the transmitter is active now (including loop-back mode). 0x0 : device 0 0x1 : device 1 0x2 : device 2 0x3 : device 3
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 0x2 : 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.

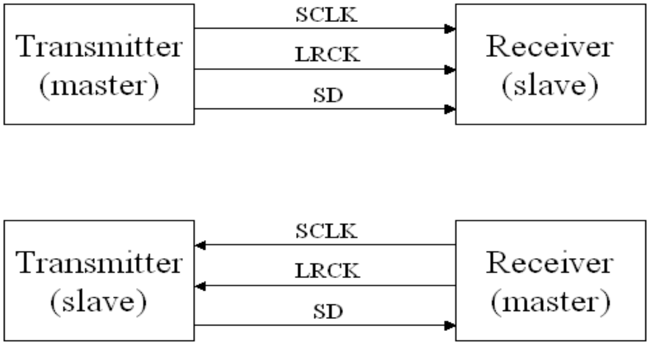
Notes: Attr: **RW** – Read/writable, **R** – read only, **W** – write only

27.4 Functional Description

27.4.1 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.

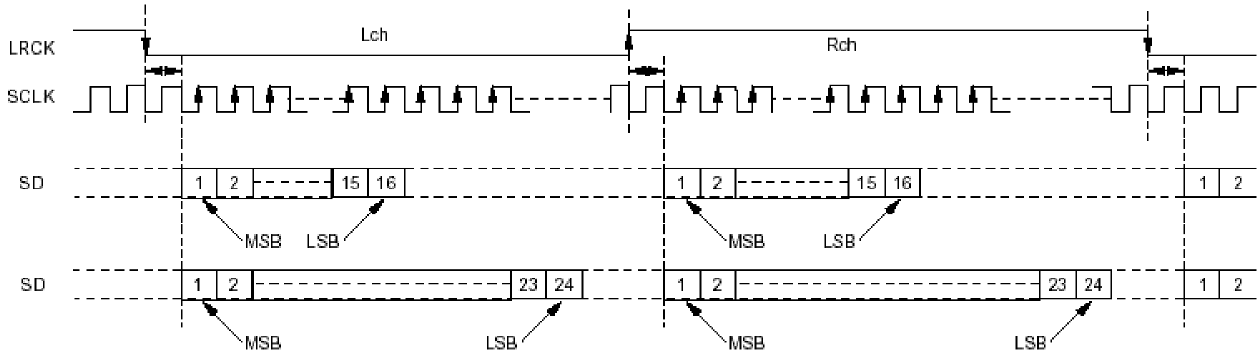


Fig. 27-2 I2S Controller timing format for I2S interface

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.

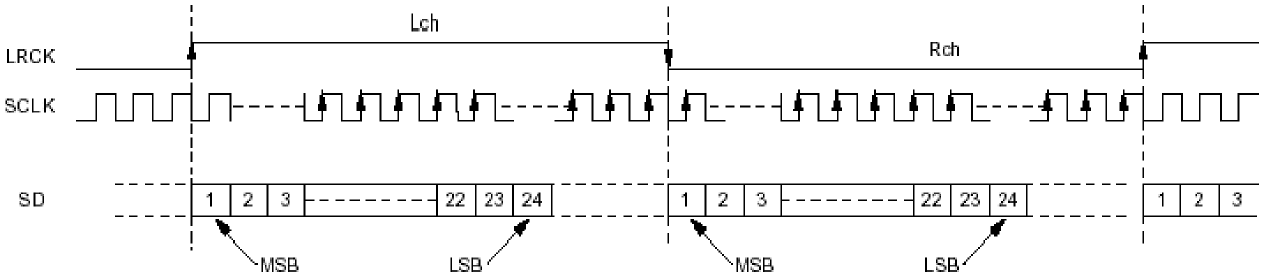


Fig. 27-3 I2S Controller timing format for Left-Justified interface

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.

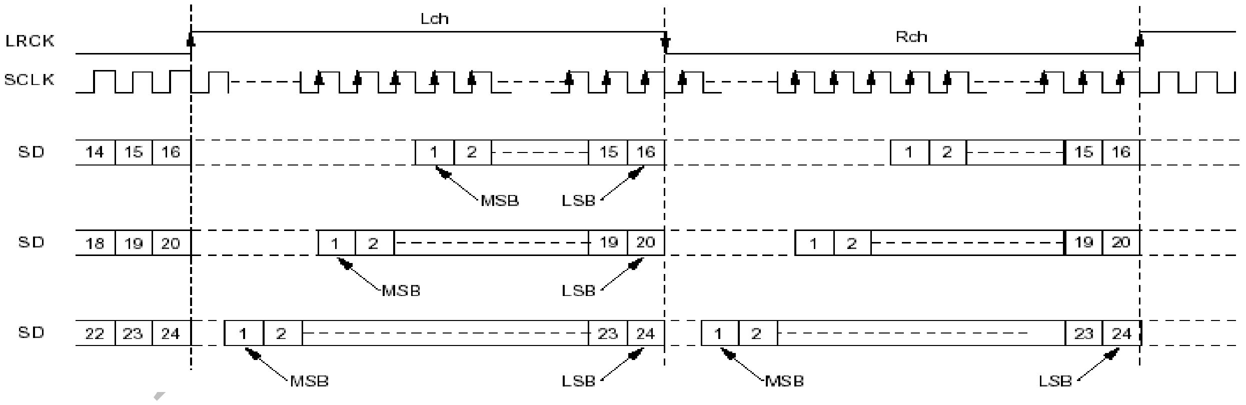


Fig. 27-4 I2S Controller timing format for Right-Justified interface

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.

27.4.2 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.

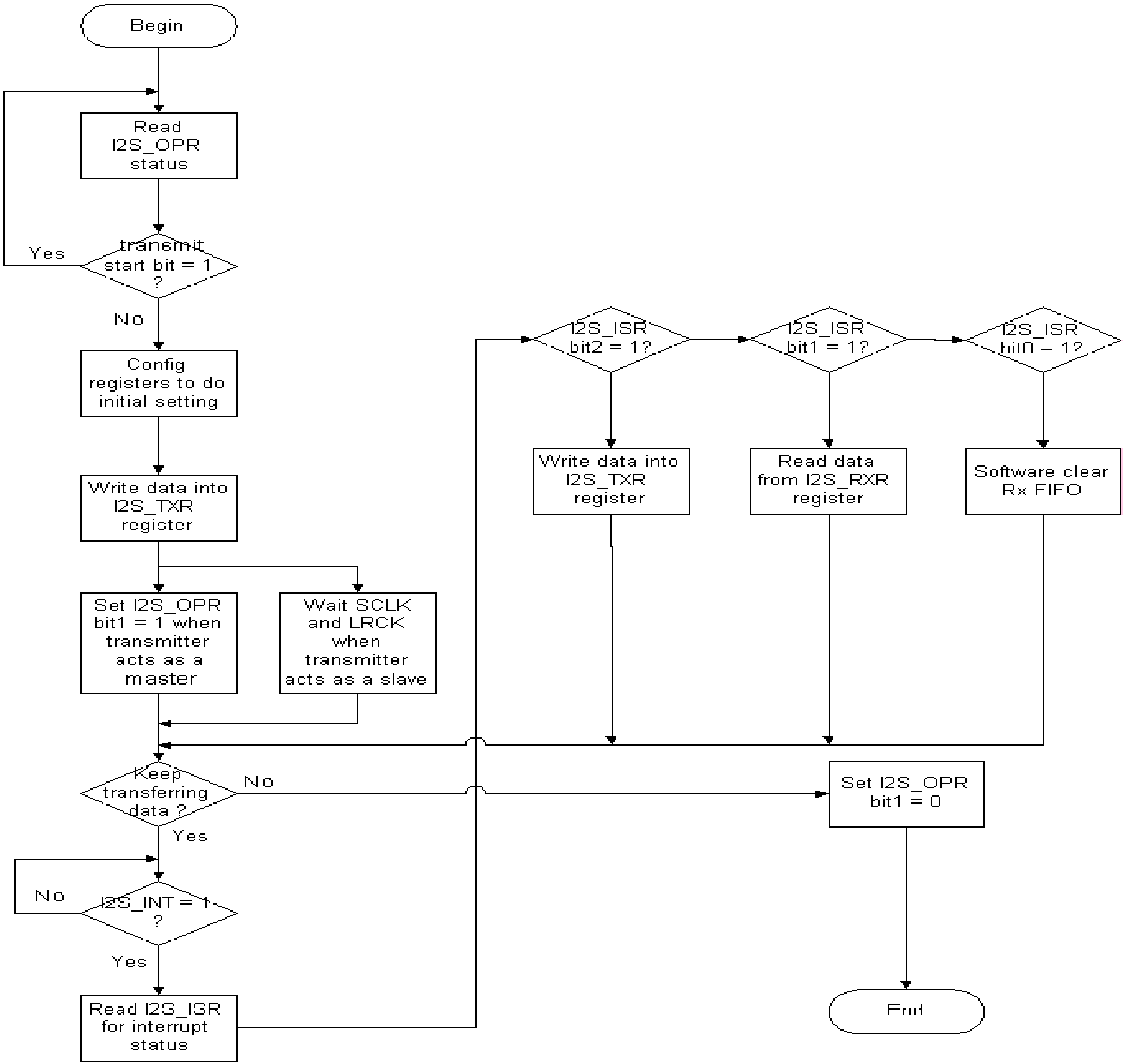

Begin
Read I2S_OPR status
transmit start bit = 1 ?
Yes
No
Config registers to do initial setting
Write data into I2S_TXR register
Set I2S_OPR bit1 = 1 when transmitter acts as a master
Wait SCLK and LRCK when transmitter acts as a slave
Keep transferring data ?
No
Yes
I2S_INT = 1 ?
No
Yes
Read I2S_ISR for interrupt status
I2S_ISR bit2 = 1?
I2S_ISR bit1 = 1?
I2S_ISR bit0 = 1?
Write data into I2S_TXR register
Read data from I2S_RXR register
Software clear Rx FIFO
Set I2S_OPR bit1 = 0
End

Fig. 27-5 I2S Controller TX operation flow chart

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.

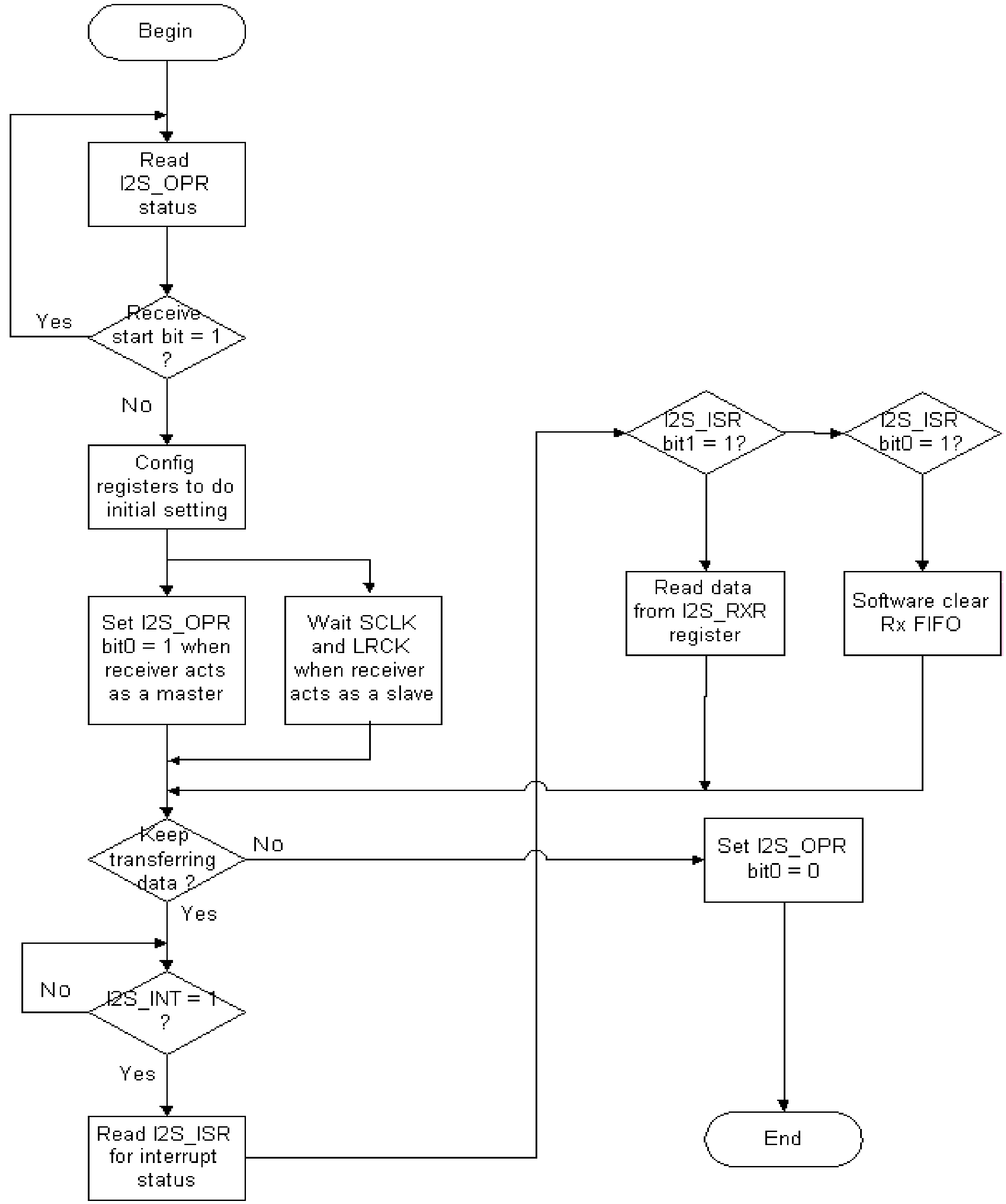

Begin
Read I2S_OPR status
Receive start bit = 1 ?
Yes
No
Config registers to do initial setting
Set I2S_OPR bit0 = 1 when receiver acts as a master
Wait SCLK and LRCK when receiver acts as a slave
Keep transferring data ?
No
Yes
I2S_INT = 1 ?
No
Yes
Read I2S_ISR for interrupt status
I2S_ISR bit1 = 1?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

Fig. 27-6 I2S Controller RX operation flow chart

Chapter 28 PWM Timer

28.1 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.

28.1.1 Key Features

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

28.2 Architecture

28.2.1 Block Diagram

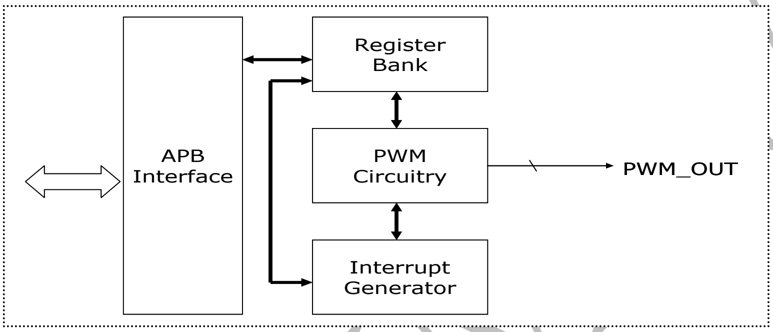


Fig. 28-1 PWM design architecture

28.2.2 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.

28.3 Registers

28.3.1 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

28.3.2 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 LOW reference or capture register

bit	Attr	Reset Value	Description
31:0	RW	0	PWM LOW 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 29 SAR-ADC Controller

29.1 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.

29.1.1 Key Features

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

29.2 Architecture

29.2.1 Block Diagram

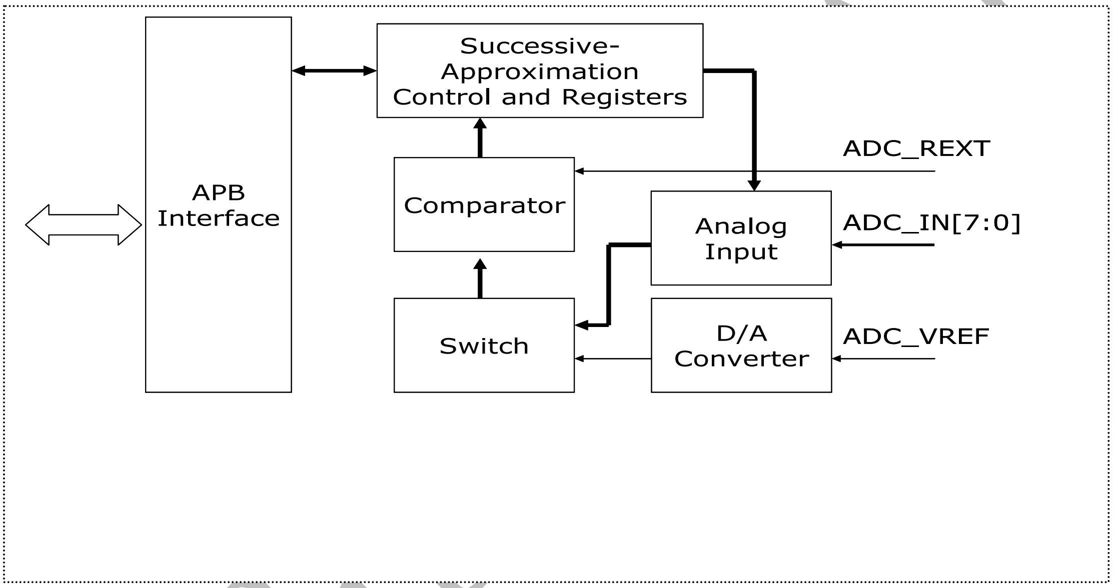


Fig. 29-1 SAR-ADC Controller design architecture

29.2.2 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.

29.3 Registers

29.3.1 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

29.3.2 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

29.4 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 30 GPIO in CPU System

30.1 Design Overview

30.1.1 Overview

GPIO is a programmable General Purpose Programming I/O peripheral. This component is a APB slave device. GPIO controls the output data and direction of external I/O pads. It also can read back the data on external pads using memory-mapped registers.

There are two GPIOs module in CPU system: GPIO_0 module and GPIO_1 module. Each GPIO module has 32 I/O pins, which is divided into 4 groups. For easy using, we name these GPIOs from A to H ports. Pay more attention that only 16 IO ports in group A and group E can support interrupt function, and connected to interrupt controller.

30.1.2 Features

- 32 bits APB bus width
- 32 independently configurable signals
- Eight ports, A to H, which are separately configurable
- Configurable interrupt mode for Port A and Port E
- Separate data registers and data direction registers for each signal
- Software control for each bit of each signal

30.2 Architecture

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

30.2.1 Block Diagram

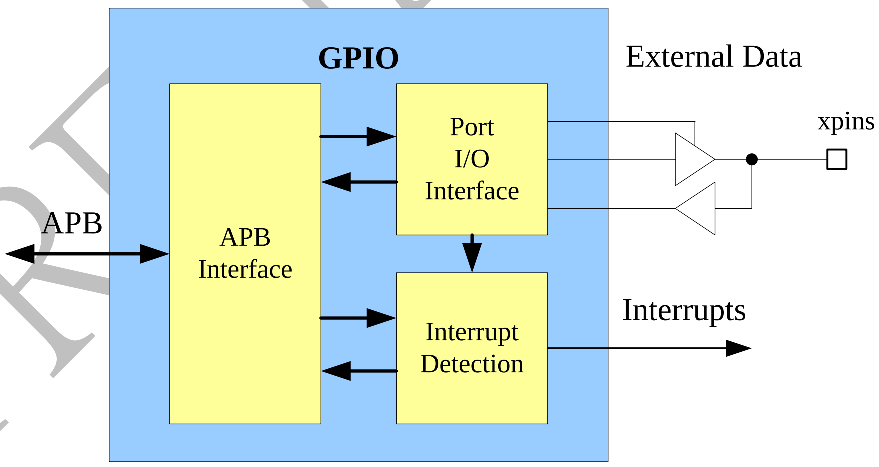


Fig. 30-1 GPIO in CPU System Block Diagram

30.2.2 Block Descriptions

APB Interface

The APB Interface implements the APB slave operation. It’s bus width is 32 bits.

Port I/O Interface

External data Interface to or from I/O pads.

Interrupt Detection

Interrupt interface to or from interrupt controller.

30.3 Registers

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

Following GPIO registers is only for GPIO_0 module. For GPIO_1 module, regsister functions is same and just change base address from 0x1800 8000 to 0x1800 1900.

30.3.1 Registers Summary

Name	Offset	Size	Reset Value	Description
GPIO_SWPORTA_DR	0x0000	W	0x00000000	Port A data register
GPIO_SWPORTA_DDR	0x0004	W	0x00000000	Port A data direction register
GPIO_SWPORTB_DR	0x000C	W	0x00000000	Port B data register
GPIO_SWPORTB_DDR	0x0010	W	0x00000000	Port B data direction register
GPIO_SWPORTC_DR	0x0018	W	0x00000000	Port C data register
GPIO_SWPORTC_DDR	0x001C	W	0x00000000	Port C data direction register
GPIO_SWPORTD_DR	0x0024	W	0x00000000	Port D data register
GPIO_SWPORTD_DDR	0x0028	W	0x00000000	Port D data direction register
GPIO_INTEN	0x0030	W	0x00000000	Port A Interrupt enable register
GPIO_INTMASK	0x0034	W	0x00000000	Port A Interrupt mask register
GPIO_INTTYPE_LEVEL	0x0038	W	0x00000000	Port A Interrupt level register
GPIO_INT_POLARITY	0x003C	W	0x00000000	Port A Interrupt polarity register
GPIO_INT_STATUS	0x0040	W	0x00000000	Interrupt status of port A
GPIO_INT_RAWSTATUS	0x0044	W	0x00000000	Raw Interrupt status of port A
GPIO_DEBOUNCE	0x0048	W	0x00000000	Debounce enable register
GPIO_PORTA_EOI	0x004C	W	0x00000000	Port A clear interrupt register
GPIO_EXT_PORTA	0x0050	W	0x00000000	Port A external port register
GPIO_EXT_PORTB	0x0054	W	0x00000000	Port B external port register
GPIO_EXT_PORTC	0x0058	W	0x00000000	Port C external port register
GPIO_EXT_PORTD	0x005C	W	0x00000000	Port D external port register
GPIO_LS_SYNC	0x0060	W	0x00000000	Level_sensitive synchronization enable register

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

30.3.2 Detail Register Description

GPIO_SWPORTA_DR
Address: Operational Base + offset(0x00)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port A if the corresponding data direction bits for Port A are set to Output mode. The value read back is equal to the last value written to this register.

GPIO_SWPORTA_DDR

Address: Operational Base + offset(0x04)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port A. 0: Input (default) 1: Output

GPIO_SWPORTB_DR
Address: Operational Base + offset(0x0C)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port B if the corresponding data direction bits for Port B are set to Output mode. The value read back is equal to the last value written to this register.

GPIO_SWPORTB_DDR
Address: Operational Base + offset(0x10)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port B. 0: Input (default) 1: Output

GPIO_SWPORTC_DR
Address: Operational Base + offset(0x18)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register are output on the I/O signals for Port C if the corresponding data direction bits for Port C are set to Output mode. The value read back is equal to the last value written to this register.

GPIO_SWPORTC_DDR
Address: Operational Base + offset(0x1C)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port C. 0: Input (default) 1: Output

GPIO_SWPORTD_DR
Address: Operational Base + offset(0x24)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved

7:0	RW	0x00	Values written to this register are output on the I/O signals for Port D if the corresponding data direction bits for Port D are set to Output mode. The value read back is equal to the last value written to this register.

GPIO_SWPORTD_DDR
Address: Operational Base + offset(0x28)
Port A data register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in Port D. 0: Input (default) 1: Output

GPIO_INTEN
Address: Operational Base + offset(0x30)
Interrupt enable register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Allows each bit of Port A to be configured for interrupts. Whenever a 1 is written to a bit of this register, it configures thecorresponding bit on Port A to become an interrupt; otherwise, Port A operates as a normal GPIO signal. Interrupts are disabled on the corresponding bits of Port A if the corresponding data direction register is set to Output. 0: Configure Port A bit as normal GPIO signal (default) 1: Configure Port A bit as interrupt

GPIO_INTMASK
Address: Operational Base + offset(0x34)
Interrupt mask register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls whether an interrupt on Port A can create an interrupt for the interrupt controller by not masking it. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. 0: Interrupt bits are unmasked (default) 1: Mask interrupt

GPIO_INTTYPE_LEVEL
Address: Operational Base + offset(0x38)
Interrupt level register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls the type of interrupt that can occur on Port A. 0: Level-sensitive (default) 1: Edge-sensitive

GPIO_INT_POLARITY
Address: Operational Base + offset(0x3C)
Interrupt polarity register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls the polarity of edge or level sensitivity that can occur on input of Port A. 0: Active-low (default) 1: Active-high

GPIO_INTSTATUS
Address: Operational Base + offset(0x40)
Interrupt status register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	Interrupt status of Port A

GPIO_RAWINTSTATUS
Address: Operational Base + offset(0x44)
Raw Interrupt status register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	Raw interrupt of status of Port A (premasking bits)

GPIO_DEBOUNCE
Address: Operational Base + offset(0x48)
Debounce enable register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. Writing a 1 to a bit in this register enables the debouncing circuitry. A signal must be valid for two periods of an external clock before it is internally processed. 0: No debounce (default) 1: Enable debounce

GPIO_PORTS_EOI
Address: Operational Base + offset(0x4C)
Port A clear interrupt register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	W	0x00	Controls the clearing of edge type interrupts from Port A. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts. 0: No interrupt clear (default) 1: Clear interrupt

GPIO_EXT_PORTA
Address: Operational Base + offset(0x50)
Port A external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port A is configured as Input, then reading this location reads the values on the signal. When the data direction of Port A is set as Output, reading this location reads the data register for Port A.

GPIO_EXT_PORTB
Address: Operational Base + offset(0x54)
Port B external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port B is configured as Input, then reading this location reads the values on the signal. When the data direction of Port B is set as Output, reading this location reads the data register for Port B.

GPIO_EXT_PORTC
Address: Operational Base + offset(0x58)
Port C external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port C is configured as Input, then reading this location reads the values on the signal. When the data direction of Port C is set as Output, reading this location reads the data register for Port C.

GPIO_EXT_PORTD
Address: Operational Base + offset(0x5C)
Port D external port register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	R	0x00	When Port D is configured as Input, then reading this location reads the values on the signal. When the data direction of Port D is set as Output, reading this location reads the data register for Port D.

GPIO_LS_SYNC
Address: Operational Base + offset(0x60)
Level_sensitive synchronization enable register

bit	Attr	Reset Value	Description
31:8	-	-	Reserved
7:0	RW	0x00	Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr. 0: No synchronization to pclk_intr (default) 1: Synchronize to pclk_intr

30.4 Functional Description

30.4.1 Operation

Control Mode(software)

Under software control, the data and direction control for the signal are sourced from the data register (GPIO_SWPORTX_DR) and direction control register (GPIO_SWPORTX_DDR), where X is either A, B, C, or D.

The direction of the external I/O pad is controlled by a write to the Portx data direction register (GPIO_SWPORTX_DDR). The data written to this memory-mapped register gets mapped onto an output signal, GPIO_PORTX_DDR, of the GPIO peripheral. This output signal controls the direction of an external I/O pad.

The data written to the Portx data register (GPIO_SWPORTX_DR) drives the output buffer of the I/O pad. External data are input on the external data signal, GPIO_EXT_PORTX.

Reading the external signal register (GPIO_EXT_PORTX) shows the value on the signal, regardless of the direction. This register is read-only.

Reading External Signals

The data on the GPIO_EXT_PORTX external signal can always be read. The data on the external gpio signal is read by an APB read of the memory-mapped register, GPIO_EXT_PORTX.

An APB read to the GPIO_EXT_PORTX register yields a value equal to that which is on the GPIO_EXT_PORTX signal.

Interrupts

Port A can be programmed to accept external signals as interrupt sources on any of the bits of the signal. The type of interrupt is programmable with one of the following settings:

- Active-high and level
- Active-low and level
- Rising edge
- Falling edge

The interrupts can be masked by programming the GPIO_INTMASK register. The interrupt status can be read before masking (called raw status) and after masking.

The interrupts are combined into a single interrupt output signal, which has the same polarity as the individual interrupts. In order to mask the combined interrupt, all individual interrupts have to be masked. The single combined interrupt does not have its own mask bit. GPIO_STATUS register must be read in the interrupt service routine (ISR) to find the source of the interrupt.

Whenever Port A is configured for interrupts, the data direction must be set to Input. If the data direction register is reprogrammed to Output, then any pending interrupts are not lost. However, no new interrupts are generated.

For edge-detected interrupts, the ISR can clear the interrupt by writing a 1 to the GPIO_PORTA_EOI register for the corresponding bit to disable the interrupt. This write also clears the interrupt status and raw status registers.

Writing to the GPIO_PORTA_EOI register has no effect on level-sensitive interrupts. If level-sensitive interrupts cause the processor to interrupt, then the ISR can poll the GPIO_RAWINT status register until the interrupt source disappears, or it can write to the GPIO_INTMASK register to mask the interrupt before exiting the ISR. If the ISR exits without masking or disabling the interrupt prior to exiting, then the level-sensitive interrupt repeatedly requests an interrupt until the interrupt is cleared at the source.

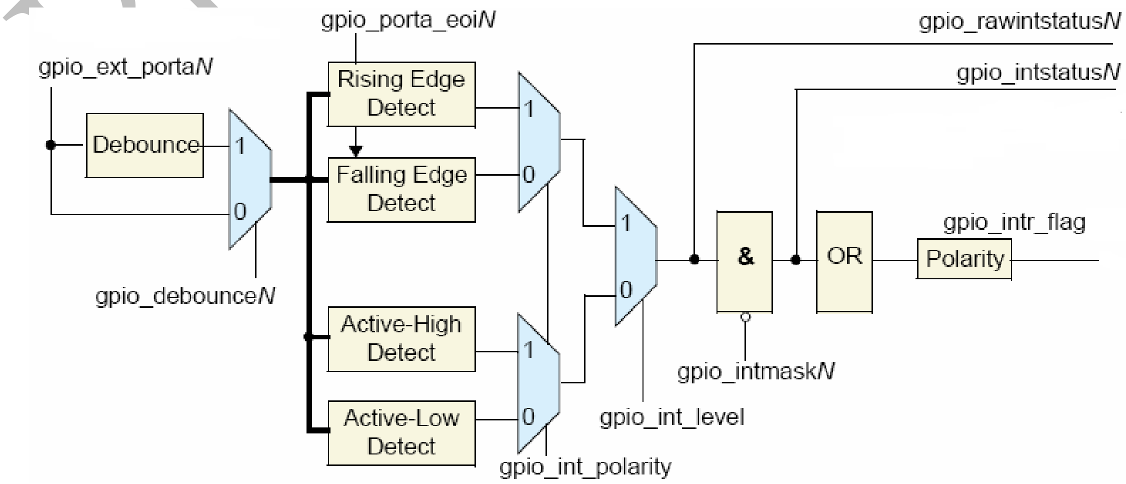


Fig. 30-2 GPIO in CPU System Intrrrupt RTL Block Diagram

Debounce operation

Port A has been configured to include the debounce capability interrupt feature. The external signal can be debounced to remove any spurious glitches that are less than one period of the external debouncing clock.

When input interrupt signals are debounced using a debounce clock(pclk), the signals must be active for a minimum of two cycles of the debounce clock to guarantee that they are registered. Any input pulse widths less than a debounce clock period are bounced. A pulse width between one and two debounce clock widths may or may not propagate, depending on its phase relationship to the debounce clock. If the input pulse spans two rising edges of the debounce clock, it is registered. If it spans only one rising edge, it is not registered.

Synchronization of Interrupt Signals to the System Clock

Interrupt signals are internally synchronized to pclk. Synchronization to pclk must occur for edge-detect signals. With level-sensitive interrupts, synchronization is optional and under software control(GPIO_LS_SYNC).

30.4.2 Programming

Programming Considerations

● Reading from an unused location or unused bits in a particular register always returns zeros. There is no error mechanism in the APB.

● Programming the GPIO registers for interrupt capability, edge-sensitive or level-sensitive interrupts, and interrupt polarity should be completed prior to enabling the interrupts on Port A in order to prevent spurious glitches on the interrupt lines to the interrupt controller.

● Writing to the interrupt clear register clears an edge-detected interrupt and has no effect on a level-sensitive interrupt.

Chapter 31 General Register File in CPU System

31.1 Overview

In CPU system, the General Register File will be used to do static set by software, which is composed of many registers for system control.

31.2 Registers

This section describes the registers for this module.

31.2.1 Registers Summary

Name	Offset	Size	Reset Value	Description
CPU_APB_REG0	0x0000	w	0x0	General controller register0
CPU_APB_REG1	0x0004	w	0x0	General controller register1
CPU_APB_REG2	0x0008	w	0x0	General controller register2
CPU_APB_REG3	0x000c	w	0x0	General controller register3
CPU_APB_REG4	0x0010	w	0x00040000	General controller register4
CPU_APB_REG5	0x0014	w	0x0	General controller register5
CPU_APB_REG6	0x0018	w	0x0	General controller register6
CPU_APB_REG7	0x001c	w	0x0	General controller register7
IOMUX_A_CON	0x0020	w	0x0	IO MUX controller register
IOMUX_B_CON	0x0024	w	0x0	IO MUX controller register
GPIO_AB_PU_CON	0x0028	w	0x55555555	Pull up or down control for GPIO A and B group
GPIO_CD_PU_CON	0x002c	w	0x55555555	Pull up or down control for GPIO C and D group
GPIO_EF_PU_CON	0x0030	w	0xaaaa55aa	Pull up or down control for GPIO E and F group
GPIO_GH_PU_CON	0x0034	w	0xaaaaaaaa	Pull up or down control for GPIO G and H group
OTGPHY_CON0	0x0038	w	0x16fbc963	OTG PHY Control signals
OTGPHY_CON1	0x003c	w	0x00000008	OTG PHY Control signals

31.2.2 Detail Registers Description

CPU_APB_REG0
Address : Base Addr+0x00

bit	Attr	Reset Value	Description
31:11	R	-	Reserved
10	R	0x0	cx_ocm_core_rst status
9	R	0x0	codec pll lock status
8	R	0x0	dsp pll lock status
7	R	0x0	cpu pll lock status
6:4	R	0x0	timer2/1/0 en status
3	R	0x0	Spi slave controller sleep status
2	R	0x0	Spi master controller sleep status
1	R	0x0	utmi_linestate[1]
0	R	0x0	utmi_linestate[0]

CPU_APB_REG1

Address : Base Addr+0x04

bit	Attr	Reset Value	Description
31:0	-	-	Reserved

CPU_APB_REG2

Address : Base Addr+0x08

bit	Attr	Reset Value	Description
31:0	-	-	Reserved

CPU_APB_REG3

Address : Base Addr+0x0c

bit	Attr	Reset Value	Description
31:0	-	-	Reserved

CPU_APB_REG4

Address : Base Addr+0x10

bit	Attr	Reset Value	Description
31	RW	0x0	Switch between lcd dma req4 and sdmmc1 0 : sdmmc1 1 : lcd dma req4
30	RW	0x0	Spi master controller interface type select
29	RW	0x0	arbiter in exp bus enter pause mode control. Active high
28	RW	0x0	arbiter in armd bus enter pause mode control. Active high
27	RW	0x0	host interface enable. Active high
26	RW	0x0	Host inerafce data bus bit control 0 : 8bit 1 : 16bit
25	RW	0x0	LCDC bypass 0 : disable 1 : enable
24	RW	0x0	SDRAM IO voltage 1.8V enable 0 : disable , 2.5V or 3.3V 1 : enable , 1.8V
23	RW	0x0	Demodulator ldpc output data reverse enable 0 : normal 1 : reverse
22	RW	0x0	sdram read pipe control signals. Active high
21	RW	0x0	sdram exit self refresh control. Active high
20	RW	0x0	static memory power down control. Active high
19	RW	0x0	sdram power down control. Active high
18:16	RW	0x100	nor flash data bus width select signals 3'b000 : 16bit 3'b100 : 8bit Others : reserved
15	RW	0x0	mobile sdram controller select signals 0 : sdr sdram 1 : mobile sdram
14:0	RW	0x0	priority set for 5 data ports in sdram controller

CPU_APB_REG5

Address : Base Addr+0x14

bit	Attr	Reset Value	Description
31:13	RW	0x0	Reserved
12	RW	0x0	ARM itcm & dtcm wait select: 0 : zero wait cycle 1 : one wait cycle
11:10	RW	0x0	share mem 1 select signal: 00 : demodulator 01 : DSP l2 mem 10 : CPU l2 mem 11 : reserved
9:8	RW	0x0	share mem 0 select signal: 00 : demodulator 01 : DSP l2 mem 10 : CPU l2 mem 11 : reserved
7	RW	0x0	VIP Vsync valid porality control: 0 : low valid (default) 1 : high valid
6	RW	0x0	LCDC IO tri control 0 : disable , normal 1 : enable , high-z output
5	RW	0x0	CPU to DSP interrupt (nmi) .
4	RW	0x0	DSP boot control. Active high
3	RW	0x0	DSP external wait control . Active high
2	RW	0x0	DSP wake up control . Active high
1	RW	0x0	fiq interrupt control to CPU
0	RW	0x0	remap control

CPU_APB_REG6
Address : Base Addr+0x18

bit	Attr	Reset Value	Description
31:0	RW	0x0	DSP boot vector address

CPU_APB_REG7
Address : Base Addr+0x1c

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

IOMUX_A_CON
Address : Base Addr+0x20

bit	Attr	Reset Value	Description
31	RW	0x0	Reserved
30	RW	0x0	Gpioe_i2c0_sel 0 : i2c0_sda/scl 1 : gpio_e4/e5
29:28	RW	0x0	gpioe_u1ir_i2c1 00 : gpio_e6/e7 01 : uart1_sir_in/sir_out_n 10 : i2c1_sda/scl
27:26	RW	0x0	gpiof1_uart1_cpwm1 00 : gpio_f1 01 : uart1_sout 10 : cx_timer1_pwm

25:24	RW	0x0	gpiof0_uart1_cpwm0 00 : gpio_f0 01 : uart1_sin 10 : cx_timer0_pwm
23	RW	0x0	gpiog_mmc1_sel 0 : gpio_g2/g3/g7 1: sdmmc1_cmd/data0/clkout
22	RW	0x0	gpiog_mmc1d_sel 0 : gpio_g4/g5/g6 1 : sdmmc1_data1/data2/data3
21	RW	0x0	gpioe_spi1_sel 0 : gpio_e1/e2/e3/f7 1 : spi1_clkin/spi1_ss_in_n /spi1_rxd/spi1_txd
20:18	RW	0x0	Reserved
17:16	RW	0x0	Gpio b0_spi0csn1_mmc1pca 00 : gpio_b0 01 : spi0_csn1 10 : sdmmc1_pwr_en
15:14	RW	0x0	gpiog1_uart0_mmc1wpt 00 : gpio_g1 01 : uart0_sout 10 : sdmmc1_write_prt
13:12	RW	0x0	gpiog0_uart0_mmc1det 00 : gpio_g0 01 : uart0_sin 10 : sdmmc1_detect_n
11:10	RW	0x0	gpiof4_apwm2_mmc0wpt 00 : gpio_f4 01 : pwm2 10 : sdmmc0_write_prt
9:8	RW	0x0	gpiof3_apwm1_mmc0detn 00 : gpio_f3 01 : pwm1 10 : sdmmc0_detect_n
7:6	RW	0x0	Gpiob1_smcs1_mmc0pca 00 : gpio_b1 01 : sm_cs1_n 10 : sdmmc0_pwr_en
5	RW	0x0	gpioh_mmc0d_sel 0 : gpio_h2/h3/h4 1 : sdmm0_data1/data2/data3
4	RW	0x0	gpioh_mmc0_sel 0 : gpio_h0/h1/h5 1 : sdmmc0_cmd/data0/clkout
3:2	RW	0x0	Gpio b_spi0_mmc0 00 : gpio_b5/b6/b7 01 : spi0_clkout/spi0_txd/spi0_rxd 10 : sdmmc0_data5/data6/data7
1:0	RW	0x0	Gpio b4_spi0cs0_mmc0d4 00 : gpio_b4 01 : spi0_csn0 10 : sdmmc0_data4

IOMUX_B_CON

Address : Base Addr+0x24

bit	Attr	Reset Value	Description

31:22	RW	0x0	Reserved
21:20	RW	0x0	cxgpio_gpsclk_hsadcclkout 00 : gpio2_24 01 : gps clk 10 : hsadc_clkout
19	RW	0x0	hsadcdata_tscon_sel 0: hsadc_data_i[9:8] 1: ts_fail / ts_valid
18	RW	0x0	Gpio a7_flashcs3_sel 0 : gpio_a7 1 : flash_cs3
17	RW	0x0	Gpio a6_flashcs2_sel 0 : gpio_a6 1 : flash_cs2
16	RW	0x0	Gpio a5_flashcs1_sel 0 : gpio_a5 1 : flash_cs1
15:14	RW	0x0	gpiof5_apwm3_dpwm 00 : gpio_f5 01 : pwm3 10 : demod pwm out
13	RW	0x0	Gpio b3_u0rtsn_sel 0 : gpio_b3 1 : uart0_rts_n
12	RW	0x0	Gpio b2_u0ctsn_sel 0 : gpio_b2 1 : uart0_cts_n
11	RW	0x0	gpiof2_apwm0_sel 0 : gpio_f2 1 : pwm0
10	RW	0x0	Gpiod_lcdc16bit_sel 0 : gpio_d0 ~ gpio_d7 1 : lcdc_data8 ~ lcdc_data15
9	RW	0x0	Gpio c_lcdc24bit_sel 0 : gpio_c2 ~ gpio_c7 1 : lcdc_data18 ~ lcdc_data23
8	RW	0x0	Gpio c_lcdc18bit_sel 0 : gpio_c0/c1 1 : lcdc_data16 ~ lcdc_data17
7	RW	0x0	cxgpio_lcdden_sel 0 : gpio2_26 1 : lcdc_denable
6	RW	0x0	cxgpio_lcdvsync_sel 0 : gpio2_25 1 : lcdc_vsync
5	RW	0x0	cxgpio_hsadc_sel 0 : gpio2_14 ~ gpio2_23 1 : hsadc_data_q[9:0]
4	RW	0x0	cxgpio_host_sel 0 : gpio2_0 ~ gpio2_13 1 : host interface
3	RW	0x0	gpioh7_hsadcclk_sel 0 : gpio_h7 1 : hsadc_clkin
2	RW	0x0	gpioh6_iq_sel 0 : gpio_h6 1 : ext_iq_index

1	RW	0x0	cxgpio_i2s_sel 0 : i2s interface 1 : gpio2_27 ~ gpio2_31
0	RW	0x0	gpiof6_vipclk_sel 0 : gpio_f6 1 : vip clkout

GPIO_AB_PU_CON
Address : Base Addr+0x28

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO A0 [3:2] GPIO A1 … [17:16] GPIO B0 … [31:30] GPIO B7 00 : Normal 01 : Pull UP 10 : Pull Down 11 : Reserved

GPIO_CD_PU_CON
Address : Base Addr+0x2c

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO C0 [3:2] GPIO C1 … [17:16] GPIO D0 … [31:30] GPIO D7 00 : Normal 01 : Pull Up 10 : Pull Down 11 : Reserved

GPIO_EF_PU_CON
Address : Base Addr+0x30

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO E0 [3:2] GPIO E1 … [17:16] GPIO F0 … [31:30] GPIO F7 00 : Normal 01 : Pull Up 10 : Pull Down 11 : Reserved

GPIO_GH_PU_CON
Address : Base Addr+0x34

bit	Attr	Reset Value	Description
31:0	RW	0x0	[1:0] GPIO G0 [3:2] GPIO G1

			… [17:16] GPIO H0 … [31:30] GPIO H7 00 : Normal 01 : Pull Up 10 : Pull Down 11 : Reserved

OTGPHY_CON0

Address : Base Addr+0x38

bit	Attr	Reset Value	Description
31	RW	0x0	Reserved
30	RW	0x0	usbphy_txrise_tune
29:28	RW	01	usbphy_txhsxv_tune
27:24	RW	0110	usbphy_txvref_tune
23:20	RW	1111	usbphy_txfsls_tune
19	RW	1	usbphy_txfreemphasis_tune
18:16	RW	011	usbphy_sqrxtune
15	RW	1	usbphy_txbitstuff_enh
14	RW	1	usbphy_txbitstuff_en
13	RW	0	usbphy_siddq
12	RW	0	usbphy_port_reset
11:10	RW	10	usbphy_refclk_sel
9:8	RW	01	usbphy_refclk_div
7:5	RW	011	usbphy_otg_tune
4	RW	0	usbphy_otg_disable
3:1	RW	001	usbphy_compdistune
0	RW	1	usb phy_common_on_n

OTGPHY_CON1

Address : Base Addr+0x3c

bit	Attr	Reset Value	Description
31:9	RW	0x0	Reserved
8	RW	0	utmi_termselect
7:6	RW	00	utmi_xcvrselect[1:0]
5:4	RW	00	utmi_opmode[1:0]
3	RW	1	utmi_suspend_n
2	RW	0	usbphy_soft_con_sel : 0 : software control usb phy disable 1 : software control usb phy enable
1	RW	0	usbphy_vbus_vld_extsel
0	RW	0	usbphy_vbus_vld_ext

Chapter 32 Port Multiplexer

32.1 Overview

RK28xx has a lot of general purpose IOs which have been described in Chapter 31 and Chapter 32. All of them are set to input mode at reset.

Most of IOs have the multiple functions shared by programmable register set. And can also be pulled-up or pulled-down by reconfigurable register. As for the detailed description for these registers, please refer to register IOMUX_A_CON / IOMUX_B_CON / GPIO_AB_PU_CON / GPIO_CD_PU_CON / GPIO_EF_PU_CON / GPIO_GH_PU_CON in Chapter 34.

32.2 Detailed description for IO MUX

The following table shows the detailed multiplexer for all GPIOs.

Table 32-1 RK28xx IO MUX List

PAD NAME	PORT Name	PAD Direction	Pin Description
CPU GPIO A			
IO_GPIO_A[5]	gpio_a[5]	B Pull Up	gpio
	flash_cs1	O	nand flash cs1
IO_GPIO_A[6]	gpio_a[6]	B Pull Up	
	flash_cs2	O	nand flash cs2
IO_GPIO_A[7]	gpio_a[7]	B Pull Up	
	flash_cs3	O	nand flash cs3
CPU GPIO B			
IO_GPIO_B[0]	gpio_b[0]	B Pull Up	gpio
	spi0_csn1	O	spi0 second chip select
	sdmmc1_pwr_en	O	sdmmc1 power control
IO_GPIO_B[1]	gpio_b[1]	B Pull Up	gpio
	sm_cs1_n	O	nor flash second chip select
	sdmmc0_pwr_en	O	sdmmc0 power control
IO_GPIO_B[2]	gpio_b[2]	B Pull Up	gpio
	uart0_cts_n	I	uart0 modem signal
IO_GPIO_B[3]	gpio_b[3]	B Pull Up	gpio
	uart0_rts_n	O	uart0 modem signal
IO_GPIO_B[4]	gpio_b[4]	B Pull Up	gpio
	spi0_csn0	O	spi0 first chip select
	sdmmc0_data[4]	B	sdmmc0 data bit4
IO_GPIO_B[5]	gpio_b[5]	B Pull Up	gpio
	spi0_clkout	O	spi0 clk out
	sdmmc0_data[5]	B	sdmmc0 data bit5
IO_GPIO_B[6]	gpio_b[6]	B Pull Up	gpio
	spi0_txd	O	spi0 txd
	sdmmc0_data[6]	B	sdmmc0 data bit6
IO_GPIO_B[7]	gpio_b[7]	B Pull Up	gpio
	spi0_rxd	I	spi0 rxd
	sdmmc0_data[7]	B	sdmmc0 data bit7
CPU GPIO C			
IO_GPIO_C[0]	gpio_c[0]	B Pull Up	gpio
	lcdc_data16	O	lcdc data bit16
IO_GPIO_C[1]	gpio_c[1]	B Pull Up	gpio

	lcdc_data17	O	lcdc data bit17
IO_GPIO_C[2]	gpio_c[2]	B Pull Up	gpio
	lcdc_data18	O	lcdc data bit18
IO_GPIO_C[3]	gpio_c[3]	B Pull Up	gpio
	lcdc_data19	O	lcdc data bit19
IO_GPIO_C[4]	gpio_c[4]	B Pull Up	gpio
	lcdc_data20	O	lcdc data bit20
IO_GPIO_C[5]	gpio_c[5]	B Pull Up	gpio
	lcdc_data21	O	lcdc data bit21
IO_GPIO_C[6]	gpio_c[6]	B Pull Up	gpio
	lcdc_data22	O	lcdc data bit22
IO_GPIO_C[7]	gpio_c[7]	B Pull Up	gpio
	lcdc_data23	O	lcdc data bit23
CPU GPIO D			
IO_GPIO_D[0]	gpio_d[0]	B Pull Up	gpio
	lcdc_data8	O	lcdc data bit8
IO_GPIO_D[1]	gpio_d[1]	B Pull Up	gpio
	lcdc_data9	O	lcdc data bit9
IO_GPIO_D[2]	gpio_d[2]	B Pull Up	gpio
	lcdc_data10	O	lcdc data bit10
IO_GPIO_D[3]	gpio_d[3]	B Pull Up	gpio
	lcdc_data11	O	lcdc data bit11
IO_GPIO_D[4]	gpio_d[4]	B Pull Up	gpio
	lcdc_data12	O	lcdc data bit12
IO_GPIO_D[5]	gpio_d[5]	B Pull Up	gpio
	lcdc_data13	O	lcdc data bit13
IO_GPIO_D[6]	gpio_d[6]	B Pull Up	gpio
	lcdc_data14	O	lcdc data bit14
IO_GPIO_D[7]	gpio_d[7]	B Pull Up	gpio
	lcdc_data15	O	lcdc data bit15
CPU GPIO E			
IO_GPIO_E[1]	gpio_e[1]	B Pull Down	gpio
	spi1_clkin	I	spi1 slave mode clock signal
IO_GPIO_E[2]	gpio_e[2]	B Pull Down	gpio
	spi1_ss_n	I	spi1 slave mode select signal
IO_GPIO_E[3]	gpio_e[3]	B Pull Down	gpio
	spi1_rxd	I	spi1 rxd
IO_GPIO_E[4]	i2c0_sda	B Pull Up	i2c0 sda
	gpio_e[4]	B Pull UP	gpio
IO_GPIO_E[5]	i2c0_scl	B Pull UP	i2c0 scl
	gpio_e[5]	B Pull UP	gpio
IO_GPIO_E[6]	gpio_e[6]	B Pull UP	gpio
	uart1_sir_in	I	uart1 IR data in
	i2c1_sda	B Pull UP	i2c1 sda
IO_GPIO_E[7]	gpio_e[7]	B Pull UP	gpio
	uart1_sir_out_n	O	uart1 IR data out
	i2c1_scl	B Pull UP	i2c1 scl
CPU GPIO F			
IO_GPIO_F[0]	gpio_f[0]	B Pull Down	gpio
	uart1_sin	I	uart1 serial data in
	cx_timer0_pwm	O	pwm out from ceva
IO_GPIO_F[1]	gpio_f[1]	B Pull Down	gpio
	uart1_sout	O	uart1 serial data out

	cx_timer1_pwm	O	pwm out from ceva
IO_GPIO_F[2]	gpio_f[2]	B Pull Down	gpio
	pwm0	B	pwm
IO_GPIO_F[3]	gpio_f[3]	B Pull Down	gpio
	pwm1	B	pwm
	sdmmc0_detect_n	I	sdmmc0 detect signal
IO_GPIO_F[4]	gpio_f[4]	B Pull Down	gpio
	pwm2	B	pwm
	sdmmc0_write_prt	I	sdmmc0 write protect
IO_GPIO_F[5]	gpio_f[5]	B Pull Down	gpio
	pwm3	B	pwm
	demod_pwm_out	O	pwm out from demodulator
IO_GPIO_F[6]	gpio_f[6]	B Pull Down	gpio
	vip_clkout	O	sensor clk out
IO_GPIO_F[7]	gpio_f[7]	B Pull Down	gpio
	spi1_txd	O	spi1 txd
CPU GPIO G			
IO_GPIO_G[0]	gpio_g[0]	B Pull Down	gpio
	uart0_sin	I	uart0 serial data in
	sdmmc1_detect_n	I	sdmmc1 card detect
IO_GPIO_G[1]	gpio_g[1]	B Pull Down	gpio
	uart0_sout	O	uart0 serial data out
	sdmmc1_write_prt	I	sdmmc1 card write protect
IO_GPIO_G[2]	gpio_g[2]	B Pull Down	gpio
	sdmmc1_cmd	B	sdmmc1 command
IO_GPIO_G[3]	gpio_g[3]	B Pull Down	gpio
	sdmmc1_data[0]	B	sdmmc1 data bit0
IO_GPIO_G[4]	gpio_g[4]	B Pull Down	gpio
	sdmmc1_data[1]	B	sdmmc1 data bit1
IO_GPIO_G[5]	gpio_g[5]	B Pull Down	gpio
	sdmmc1_data[2]	B	sdmmc1 data bit2
IO_GPIO_G[6]	gpio_g[6]	B Pull Down	gpio
	sdmmc1_data[3]	B	sdmmc1 data bit3
IO_GPIO_G[7]	gpio_g[7]	B Pull Down	gpio
	sdmmc1_clkout	O	sdmmc1 clk out
CPU GPIO H			
IO_GPIO_H[0]	gpio_h[0]	B Pull Down	gpio
	sdmmc0_cmd	B	sdmmc0 command
IO_GPIO_H[1]	gpio_h[1]	B Pull Down	gpio
	sdmmc0_data[0]	B	sdmmc0 data bit0
IO_GPIO_H[2]	gpio_h[2]	B Pull Down	gpio
	sdmmc0_data[1]	B	sdmmc0 data bit1
IO_GPIO_H[3]	gpio_h[3]	B Pull Down	gpio
	sdmmc0_data[2]	B	sdmmc0 data bit2
IO_GPIO_H[4]	gpio_h[4]	B Pull Down	gpio
	sdmmc0_data[3]	B	sdmmc0 data bit3
IO_GPIO_H[5]	gpio_h[5]	B Pull Down	gpio
	sdmmc0_clkout	O	sdmmc0 clock out
IO_GPIO_H[6]	gpio_h[6]	B Pull Down	gpio
	ext_iq_index	I	ext_iq_index from hs_adc module
IO_GPIO_H[7]	gpio_h[7]	B Pull Down	gpio
	hsadc_clkin	I	hsadc clock input for aysnc mode
DSP GPIO			

IO_GPIO2[0]	gpio2[0]	B Pull UP	gpio
	host_data0	B	host interface data bit0
IO_GPIO2[1]	gpio2[1]	B Pull UP	gpio
	host_data1	B	host interface data bit1
IO_GPIO2[2]	gpio2[2]	B Pull UP	gpio
	host_data2	B	host interface data bit2
IO_GPIO2[3]	gpio2[3]	B Pull UP	gpio
	host_data3	B	host interface data bit3
IO_GPIO2[4]	gpio2[4]	B Pull UP	gpio
	host_data4	B	host interface data bit4
IO_GPIO2[5]	gpio2[5]	B Pull UP	gpio
	host_data5	B	host interface data bit5
IO_GPIO2[6]	gpio2[6]	B Pull UP	gpio
	host_data6	B	host interface data bit6
IO_GPIO2[7]	gpio2[7]	B Pull UP	gpio
	host_data7	B	host interface data bit7
IO_GPIO2[8]	gpio2[8]	B Pull UP	gpio
	host_addr0	I	host interface addr bit0
IO_GPIO2[9]	gpio2[9]	B Pull UP	gpio
	host_addr1	I	host interface addr bit1
IO_GPIO2[10]	gpio2[10]	B Pull UP	gpio
	host_csn	I	host interface chip select
IO_GPIO2[11]	gpio2[11]	B Pull UP	gpio
	host_rdn	I	host interface read valid signal
IO_GPIO2[12]	gpio2[12]	B Pull UP	gpio
	host_wrn	I	host interface write valid signal
IO_GPIO2[13]	gpio2[13]	B Pull UP	gpio
	ap2bb_int	O	host interface interrupt from chip to host
IO_GPIO2[14]	gpio2[14]	B Pull UP	gpio
	hsadc_data_q[0]	I	hsadc data bit0 for Q path
IO_GPIO2[15]	gpio2[15]	B Pull UP	gpio
	hsadc_data_q[1]	I	hsadc data bit1 for Q path
IO_GPIO2[16]	gpio2[16]	B Pull Down	gpio
	hsadc_data_q[2]	I	hsadc data bit2 for Q path
IO_GPIO2[17]	gpio2[17]	B Pull Down	gpio
	hsadc_data_q[3]	I	hsadc data bit3 for Q path
IO_GPIO2[18]	gpio2[18]	B Pull Down	gpio
	hsadc_data_q[4]	I	hsadc data bit4 for Q path
IO_GPIO2[19]	gpio2[19]	B Pull Down	gpio
	hsadc_data_q[5]	I	hsadc data bit5 for Q path
IO_GPIO2[20]	gpio2[20]	B Pull Down	gpio
	hsadc_data_q[6]	I	hsadc data bit6 for Q path
IO_GPIO2[21]	gpio2[21]	B Pull Down	gpio
	hsadc_data_q[7]	I	hsadc data bit7 for Q path
IO_GPIO2[22]	gpio2[22]	B Pull Down	gpio
	hsadc_data_q[8]	I	hsadc data bit8 for Q path
IO_GPIO2[23]	gpio2[23]	B Pull Down	gpio
	hsadc_data_q[9]	I	hsadc data bit9 for Q path
IO_GPIO2[24]	gpio2[24]	B Pull Down	gpio
	gps clk	I Pull Down	clock input for gps application

	hsadc_clkout	O	clock out to hsadc analog
IO_GPIO2[25]	gpio2[25]	B Pull Down	gpio
	lcdc_vsync	O	lcdc vertical sync signal
IO_GPIO2[26]	gpio2[26]	B Pull Down	gpio
	lcdc_denable	O	lcdc data valid signal
IO_GPIO2[27]	i2s_sdi	I Pull Down	i2s sdi from codec
	gpio2[27]	B Pull Down	gpio
IO_GPIO2[28]	i2s_sdo	O	i2s sdo to codec
	gpio2[28]	B Pull Down	gpio
IO_GPIO2[29]	i2s_clk	O	i2s clock out to codec
	gpio2[29]	B Pull Down	gpio
IO_GPIO2[30]	i2s_lrck	B Pull Down	i2s lrck
	gpio2[30]	B Pull Down	gpio
IO_GPIO2[31]	i2s_sclk	B Pull Down	i2s serial clock
	gpio2[31]	B Pull Down	gpio

Notes : B --- Bidirectional IO
I --- Input IO
O --- Output IO

32.3 Detailed description for LCD port

Table 32-2 RK28xx LCD port MUX List

PIN NAME	18bit MCU	16bit MCU	24bit RGB	18bit RGB	8bit RGB	CCIR656
LCD_D0	DB0	DB0	R0	R0	D0	D0
LCD_D1	DB1	DB1	R1	R1	D1	D1
LCD_D2	DB2	DB2	R2	R2	D2	D2
LCD_D3	DB3	DB3	R3	R3	D3	D3
LCD_D4	DB4	DB4	R4	R4	D4	D4
LCD_D5	DB5	DB5	R5	R5	D5	D5
LCD_D6	DB6	DB6	R6	G0	D6	D6
LCD_D7	DB7	DB7	R7	G1	D7	D7
LCD_D8	DB8	DB8	G0	G2		
LCD_D9	DB9	DB9	G1	G3		
LCD_D10	DB10	DB10	G2	G4		
LCD_D11	DB11	DB11	G3	G5		
LCD_D12	DB12	DB12	G4	B0		
LCD_D13	DB13	DB13	G5	B1		
LCD_D14	DB14	DB14	G6	B2		
LCD_D15	DB15	DB15	G7	B3		
LCD_D16	DB16		B0	B4		
LCD_D17	DB17		B1	B5		
LCD_D18			B2			
LCD_D19			B3			
LCD_D20			B4			
LCD_D21			B5			
LCD_D22			B6			
LCD_D23			B7			
LCD_VSYNC /MCU_CS	CS	CS	VSYNC	VSYNC	VSYNC	
LCD_HSYNC/ MCU_WR	WR	WR	HSYNC	HSYNC	HSYNC	
LCD_DEN			(DEN)	(DEN)	(DEN)	
LCD_CLK /MCU_RS	RS	RS	DOT_CLK	DOT_CLK	DOT_CLK	DOT_CLK

Chapter 33 Hardware Information

33.1 Oscillator Connection

RK28xx will use two oscillators, one is for input of three on-chip PLLs, for USB OTG PHY, and for I2S main clock, which should be 24MHz, another is for RTC function, which should be 32.768 KHz. The design for oscillator pad has been optimized for stability and minimum jitter, and characterized to allow a variation of 4pF to 18pF on both XI and XO pins for crystal stability. In the Fig. 39-1, the variation range for C value is 4pF to 18pF.

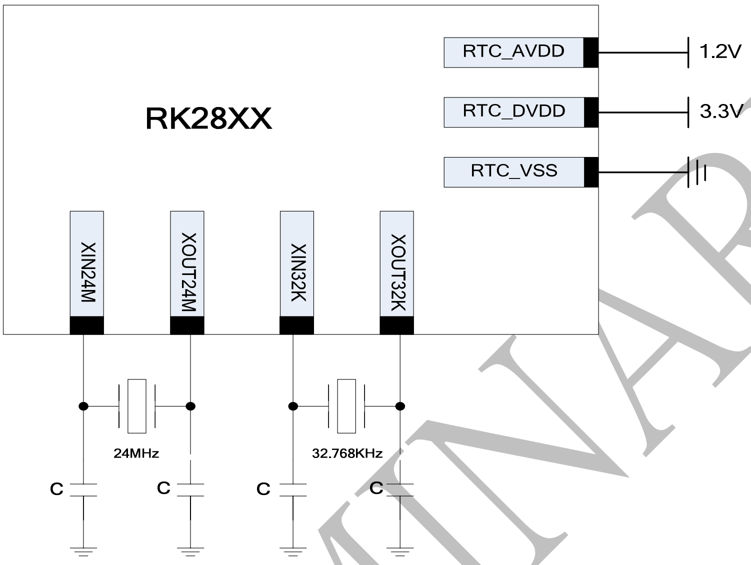


Fig. 33-1 RK28xx external oscillator connection diagram

If no using internal RTC function of RK28xx, please connect 32.768 KHz oscillators pin as following，and all of the RTC power PIN must be supplied.

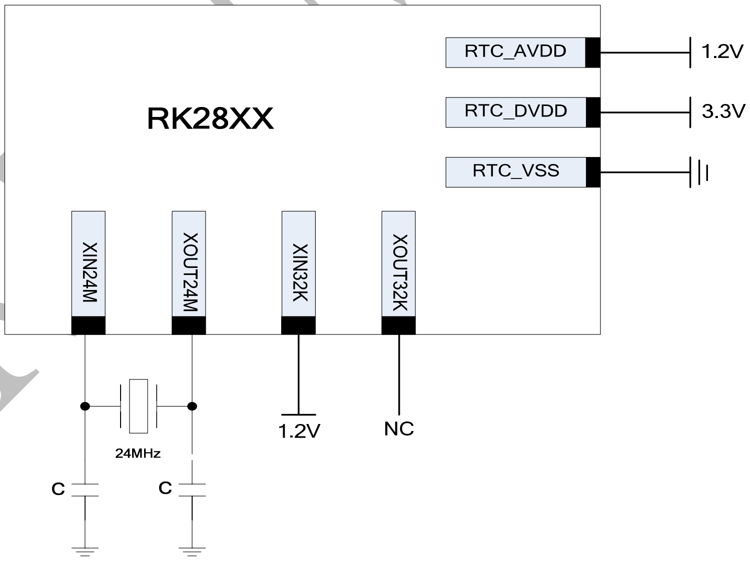


Fig. 4-2 RK28xx RTC power connection diagram

33.2 USB PHY Connection

USB2.0 OTG PHY is used in RK28xx for USB host, USB device and otg functions. The following figure shows external connection for USB PHY interface.

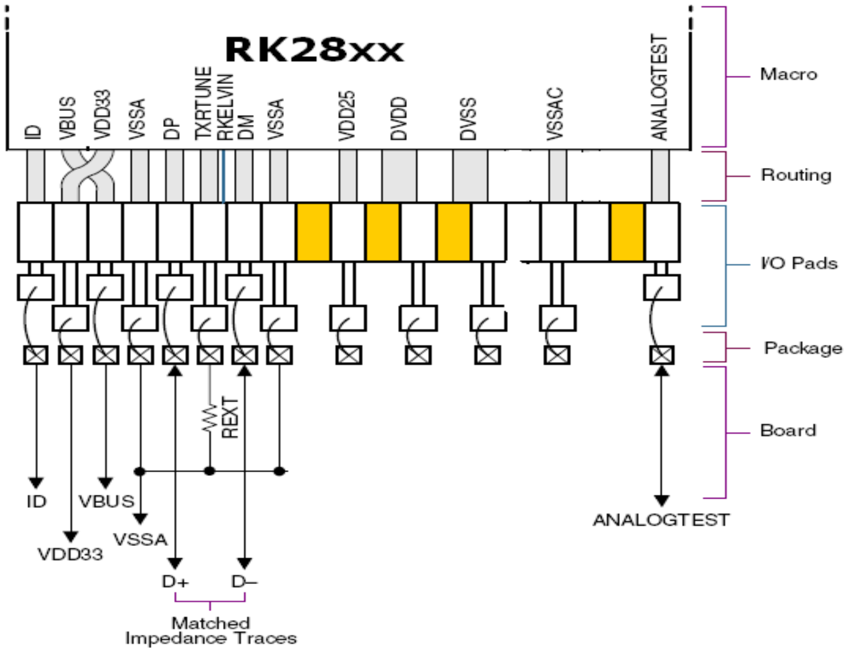


Fig. 33-2 RK28xx USB PHY connection diagram

In the above diagram, some parameters and its viariant will be shown in the following table.

External resistor (REXT)	44.2 Ω (± 1%)
Analog power supplies	3.3 V (+ 10%, – 10%) at the macro pins with respect to VSSA and VSSAC 2.5 V (+ 10%, – 10%) at the macro pins with respect to VSSA and VSSAC
Digital power supply	1.2 V (+ 10%, – 12.5%) at the macro pins with respect to DVSS
Junction temperature	–40º C through +125º C

33.3 Power up Sequence for power supply

For IO and core power supply of RK28xx, there are no power sequence requirements, since IO is 3-state when core power is not valid.

33.4 Power on reset Descriptions

The following figure shows power-on-reset sequence and relative clock behavior. When npor (power-on-reset) is released after stabilization of oscillator clock xin24m. After about T1 timing length, power supply for on-chip PLLs will be in stable state and pll_rstn (internal reset signal for PLL) is released. Then after (T2-T1) timing length, chip_rstn (internal reset signal for chip logic) is released. Then clock for IP module inside chip will be valid . After about 15 clocks , ip_rstn (internal reset signal for all IPs) will be released, which can meet some special requirements for some IPs , " reset signal will be kept valid no less than 15 clock cycles" .

Notes : T1 is about 5us ; T2 is about 139us

Another, RK28xx can filter out 5 clock cycles for low pulse of npor; the clock cycle is xin24m clock, so about 208ns low pulse of npor will not be recognized as valid power-on-reset signal for RK28xx.

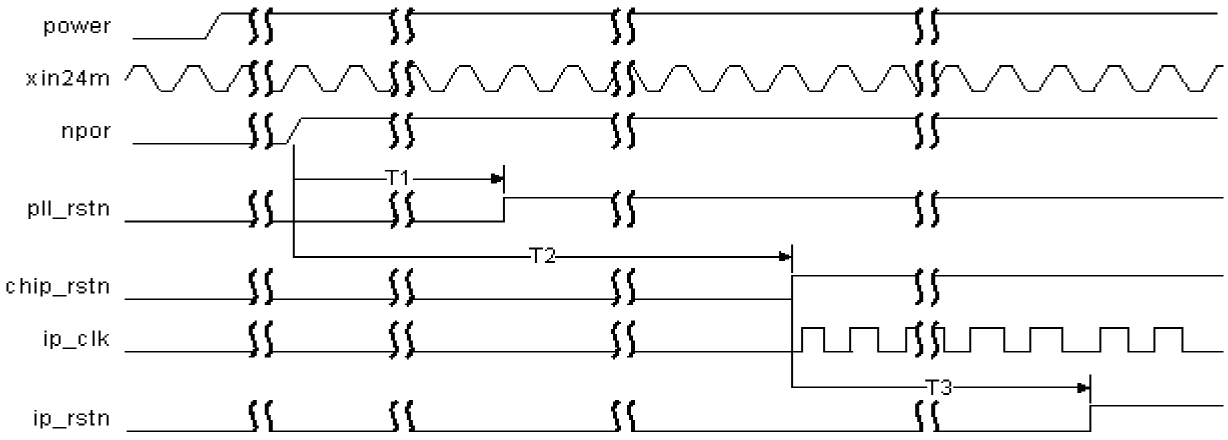


Fig. 33-3 RK28xx reset sequence timing waveform

Chapter 34 Electrical Specification

34.1 Recommended Operating Conditions

TBD

34.2 Electrical Characteristics

TBD

Appendix A – ARM926EJS16K16K

TBD

Appendix B – Refer documents

- **RK28xx USB OTG Controller.pdf**
- **RK28xx DSP sub-system.pdf**
- **RK28xx Real Time Clock.pdf**
- **RK28xx NAND Flash Controller.pdf**
- **RK28xx Efuse controller.pdf**