Rockchip
RKNanoD
Technical Reference Manual

Revision 1.0
May.2015

Revision History

Date	Revision	Description
2015-5-19	1.0	Update
2015-03-2	0.2	Update chapter 1,3,4,5,9,15,17,18 for some mistake
2015-01-26	0.1	Initial Release

Warranty Disclaimer

Rockchip Electronics Co.,Ltd makes no warranty, representation or guarantee (expressed, implied, statutory, or otherwise) by or with respect to anything in this document, and shall not be liable for any implied warranties of non-infringement, merchantability or fitness for a particular purpose or for any indirect, special or consequential damages.

Information furnished is believed to be accurate and reliable. However, Rockchip Electronics Co.,Ltd assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use.

Rockchip Electronics Co.,Ltd's products are not designed, intended, or authorized for using as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Rockchip Electronics Co.,Ltd's product could create a situation where personal injury or death may occur, should buyer purchase or use Rockchip Electronics Co.,Ltd's products for any such unintended or unauthorized application, buyers shall indemnify and hold Rockchip Electronics Co.,Ltd and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Rockchip Electronics Co.,Ltd was negligent regarding the design or manufacture of the part.

Copyright and Patent Right

Information in this document is provided solely to enable system and software implementers to use Rockchip Electronics Co.,Ltd 's products. There are no expressed and patent or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Rockchip Electronics Co.,Ltd does not convey any license under its copyright and patent rights nor the rights of others.
All copyright and patent rights referenced in this document belong to their respective owners and shall be subject to corresponding copyright and patent licensing requirements.

Trademarks

Rockchip and Rockchip™ logo and the name of Rockchip Electronics Co.,Ltd's products are trademarks of Rockchip Electronics Co.,Ltd. and are exclusively owned by Rockchip Electronics Co.,Ltd. References to other companies and their products use trademarks owned by the respective companies and are for reference purpose only.

Confidentiality

The information contained herein (including any attachments) is confidential. The recipient hereby acknowledges the confidentiality of this document, and except for the specific purpose, this document shall not be disclosed to any third party.

Reverse engineering or disassembly is prohibited.

ROCKCHIP ELECTRONICS CO.,LTD. RESERVES THE RIGHT TO MAKE CHANGES IN ITS PRODUCTS OR PRODUCT SPECIFICATIONS WITH THE INTENT TO IMPROVE FUNCTION OR DESIGN AT ANY TIME AND WITHOUT NOTICE AND IS NOT REQUIRED TO UNDATE THIS DOCUMENTATION TO REFLECT SUCH CHANGES.

Copyright © 2015 Rockchip Electronics Co., Ltd.

Table of Content

Figure Index

Table Index

Chapter 1 Introduction

RKnanoD is a ARM Cortex-M3 based microcontroller for Wireless Audio, MP3 player and IOT applications.

RKnanoD includes two M3 cores , up to 1M Bytes Ram, internal power management unit, high quality audio codec, dedicated hardware MP3 decode accelerator, hardware lossless audio decode accelerator and rich peripheral interface. RKnanoD can support Wi-Fi and Bluetooth protocol without external memory, support 24 bits 192k Hz sample rate lossless audio decoding with low power consumption, and support three power modes.

1.1 Features

The features listed below which may or may not be present in actual product, may be subject to the third party licensing requirements. Please contact Rockchip for actual product feature configurations and licensing requirements.

- **Processor**
  - Dual ARM Cortex-M3 core
    - A Thumb instruction set subset
    - Banked Stack Pointer (SP) only
    - Hardware divide instructions, SDIV and UDIV (Thumb 32-bit instructions)
    - Handler and Thread modes
    - Thumb and Debug states
    - Interruptible-continued LDM/STM, PUSH/POP for low interrupt latency
    - Automatic processor state saving and restoration for low latency Interrupt Service Routine (ISR) entry and exit
    - Support for ARMv6 unaligned accesses
  - Nested Vectored Interrupt Controller (NVIC)
    - 32-level priority of interrupt
    - Dynamic reprioritization of interrupts
    - Priority grouping. This enables selection of pre-empting interrupt levels and non-pre-empting interrupt levels
    - Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts
    - Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.
  - Mail box
    - Support dual-core system: MCU core and MPU core
    - Support APB interface
    - Support four mailbox elements, each element includes one data word, one command word register and one flag bit that can represent one interrupt
    - Four interrupts to MCU core Four interrupts to MPU core

Memory Organization

- 16KB boot ROM
- 64KB PMU SRAM for low power sleep mode
- 320KB IRAM and 256KB DRAM for Core 0
- 128KB IRAM and 256KB DRAM for Core 1

- 64KB/bank clock-gate control for reduce power consumption Power Management Unit
  - Multiple configurable work modes to save power by different frequency or automatically clock gating control or power domain on/off control
  - 2 voltage domains and 3 separate power domains, which can be power up/down by software based on different application scenes

- **CRU (clock & reset unit)**
  - Support clock gating control for individual components
  - One oscillator with 24MHz clock input and 1 embedded general purpose PLL
  - Support global soft-reset control for whole SOC, also individual soft-reset for every components

- **Hardware Accelerator for MP3 decode**
  - MP3 imdct36 calculation module
  - MP3 subband synthesize module

- **Memory Interface**
  - SD/MMC controller
    - SD/MMC SPI mode/1bit mode/4bit mode
    - Support Multi Media Card Specification Version 4.41
    - Support SD Memory Card Specification Version 2.0
    - Cards Clock Rate up to PCLK, Re-scaling the SD/MMC clock (PCLK) with the 8-bits pre-scale register in SCU block
    - Support FIFO over-run and under-run prevention by stopping card clock automatically
    - Support CRC generation and error detection

  - eMMC Interface
    - Support MMC4.41 protocol
    - Support FIFO over-run and under-run prevention by stopping card clock automatically
    - Support CRC generation and error detection
    - 8bits data bus width

- **DISPLAY interface**
  - Support source data format: RGB565, YUV420
  - Support UV swap
  - Support YUV2RGB
  - Support BT601 limited range
  - Support BT709 limited range
  - Support BT601 full range
  - Support allegro dither down for RGB888 to RGB565
  - Support RGB565 display data format
  - Support display data swap
  - Support max output resolution 400x400
  - Built-in i8080 MCU interface
  - Support EPD T-CON / E-INK

- **DMA Controller**

- Two DMA Controllers in chip
  - DMAC1 Support 6 DMA channels
  - DMAC2 Support 2 DMA channels
  - Support incremental and fixed addressing mode
  - Support hardware and software trigger DMA transfer mode
  - Support error interrupt, transport-complete interrupt
  - When transport data is not align with source burst, the last data will be transported in single burst mode
  - Support LLP mode and auto-reload

USB interface

- USB 2.0 OTG controller and PHY
- Operates in High-Speed and Full-Speed mode
- Support Session Request Protocol(SRP) and Host Negotiation Protocol(HNP)
- Support 6 endpoints , one control endpoint, two IN/OUT endpoints, one IN endpoint
- Support 4 channels at Host mode, support bulk transfer

Low speed Peripheral interface

- I2C controller
  - Support 3 I2C controllers
  - Supports master modes of I2C bus
  - 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 2 I2S controllers
  - Support mono/stereo audio file
  - Support 16 ~ 32 bits audio data transfer
  - Support audio sample rate up to 192 KHz
  - Support I2S, Left-Justified and Right-Justified digital serial data format
- PWM
  - 5 on-chip PWMs with interrupt-based operation
  - Programmable counter and duty cycle
  - Chained timer for long period purpose
  - Support single counter mode and reload mode
  - Configurable polarity
  - Support interrupt output
- SPI master
  - 2 on-chip SPIs
  - Serial-master operation – Enables serial communication with serial-slave peripheral devices
  - DMA Controller Interface – Enables interface to a DMA controller using a handshaking interface for transfer requests
  - Support interrupt interface to interrupt controller, and independently masking of interrupts
  - One hardware slave-select lines
  - Dynamic control of the serial bit rate of the data transfer
- GPIO

- 3 groups of GPIO (GPIO0~GPIO2) , 32 GPIOs per group
- All GPIOs can be used to generate interrupt to CPU
- All pull-up GPIOs are software-programmable for pull-up resistor or not
- All pull-down GPIOs are software-programmable for pull-down resistor or not
- All GPIOs are always in input direction in default after power-on-reset

- Timer
  - 2 on-chip 64bits Timers in SoC with interrupt-based operation
  - Provide two operation modes: free-running and user-defined count
  - Support timer work state checkable
- UART
  - 6 on-chip UARTs
  - 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.

- SDIO interface
  - Compatible with SDIO 2.0 protocol
  - 4bits data bus widths

- Analog IP interface
  - AUDIO-DAC
    - High Digital to Analog Convert SNR.
    - High Analog to Digital Convert SNR.
    - Differential analog input microphone with boost pre-amplify and low-noise microphone bias.
    - Stereo line-in input.
    - Stereo line output.
    - Internal PLL.
    - Stereo virtual-ground headphone amplifier with ultra low power.
    - One 24bit/8k~192K I2S/PCM interface for stereo DAC and ADC.
    - ALC (Automatic Level Control) in ADC path and DRC (Dynamic Range control) in DAC path.
    - High-pass filter in ADC path.
    - Soft pop noise suppression.

  - SAR-ADC(Successive Approximation Register)
    - 8-channel single-ended 10-bit SAR analog-to-digital converter
    - Maximum conversion rate up to 100KSPS with 1MHz A/D converter clock
    - Power supply is 3.3V (±10%) for analog interface, power dissipation is less than 900uW

1.2 Block Diagram

The following diagram shows the basic block diagram.

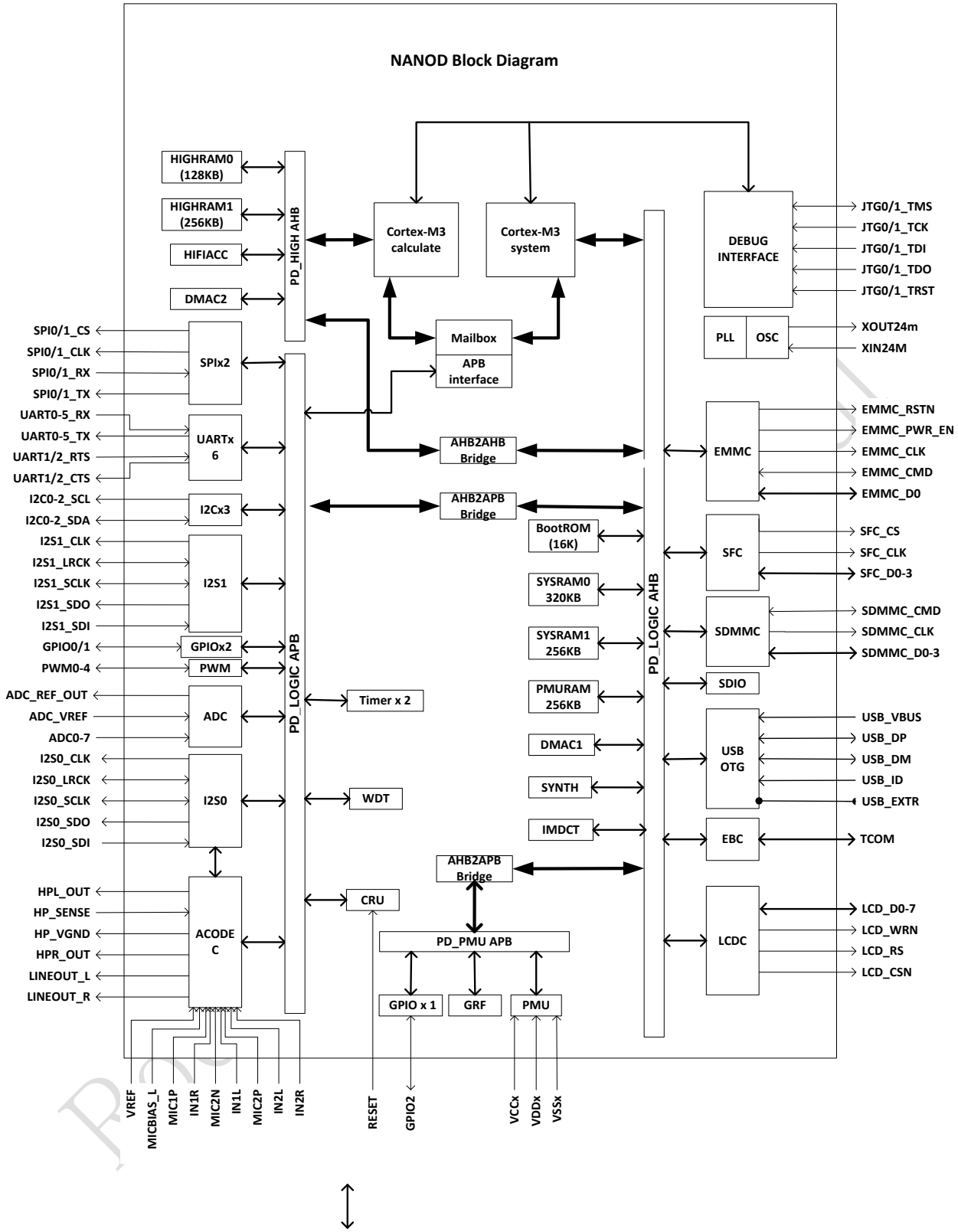


Fig 1-1 Block Diagram

Chapter 2 System Overview

2.1 Address Mapping

The chip supports boot from internal BOOTROM, which supports remap function by software programming. Remap is controlled by GRF_SOC_CON0[8].

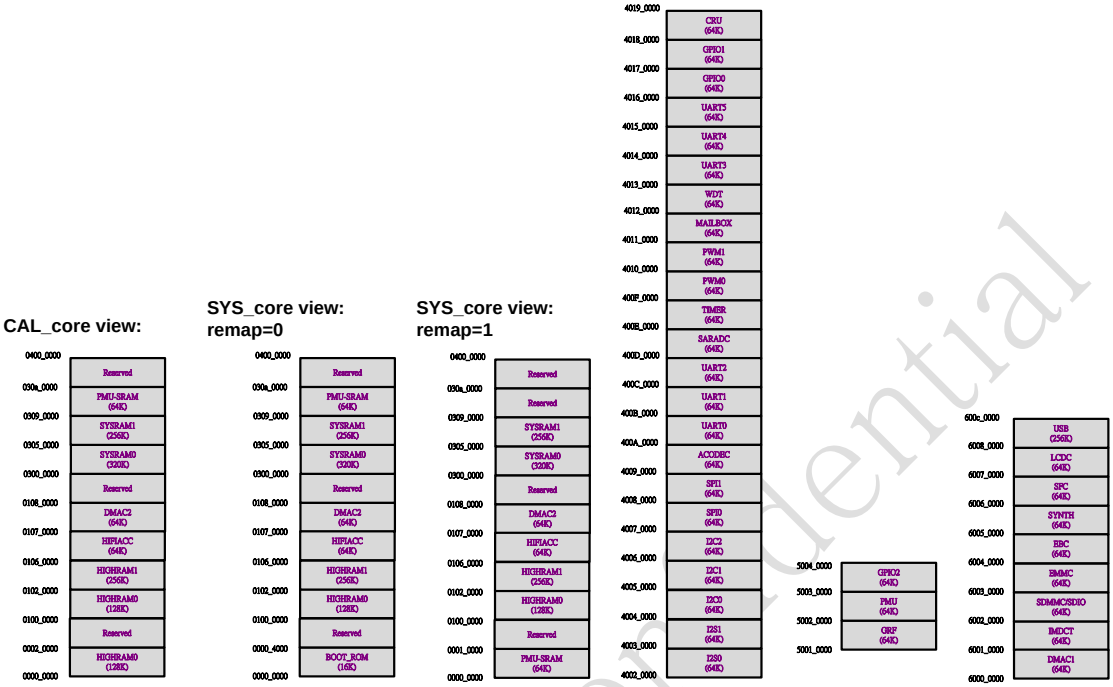


Fig 2-1 Address Mapping

2.2 System Boot

The chip provides system boot from off-chip devices such as SDMMC card, SPInor or nand, and eMMC memory. When boot code is not ready in these devices, also provide system code download into them by USB OTG interface. All of the boot code will be stored in internal BOOTROM. The following is the whole boot procedure for boot code.
The following features are supports.

- Support system boot from the following device:
  - Boot from SDMMC Card
  - Boot from eMMC flash
  - Boot from SPI Nand/Nor flash
  - Boot from USB

2.3 System Interrupt connection

There is a Nested Vectored Interrupt Controller (NVIC) in CPU processor, which has general interrupt sources for internal blocks or external devices. Each interrupts triggered type is high level, not programmable. The detailed interrupt sources connection is in the following table 2-1.

Table 2-1Interrupt connection list

int0	sfc_int
int1	synth_int
int2	ebc_int
int3	emmc_int
int4	sdmmc_int
int5	usbc_int

int6	dmac_int
int7	imdct_int
int8	wdt_int
int9	mailbox0_int
int10	mailbox1_int
int11	mailbox2_int
int12	mailbox3_int
int13	reserved
int14	reserved
int15	reserved
int16	pwm1_int
int17	pwm0_int
int18	timer1_int
int19	timer0_int
int20	saradc_int
int21	uart5_int
int22	uart4_int
int23	uart3_int
int24	uart2_int
int25	uart1_int
int26	uart0_int
int27	spi1_int
int28	spi0_int
int29	i2c2_int
int30	i2c1_int
int31	i2c0_int
int32	i2s1_int
int33	i2s0_int
int34	hifiacc_int
int35	pmu_int
int36	gpio2_int
int37	gpio1_int
int38	gpio0_int
int39	lcdc_int
int40	dma2_int

2.4 System DMA hardware request connection

The chip provides two DMA controllers: DMAC1 in pd_logic and DMAC2 in pd_high. 15 hardware request ports are used in DMAC1 and 2 hardware request ports are used in DMAC2.The trigger type for each of them is high level, not programmable.

Table 2-2 DMAC1 and DMAC2 Hardware request connection list

Req Number	Source	Polarity
0	i2s0 tx	High level
1	i2s0 rx	High level
2	i2s1 tx	High level
3	i2s1 rx	High level
4	spi0 tx	High level
5	spi0 rx	High level

6	spi1 tx	High level
7	spi1 rx	High level
8	uart0/2 tx	High level
9	uart0/2 rx	High level
10	uart1 tx	High level
11	uart1 rx	High level
12	sdmmc	High level
13	emmc	High level
14	vop	High level
15	Reserved	High level

Req Number	Source	Polarity
0	hifi rx	High level
1	hifi tx	High level

In DMAC1 hardware request 5 and 6, the hardware request can be selected between UART0 and UART2. The control bit is in GRF_INTER_CON0[1].

Chapter 3 Clock and reset unit

3.1 Overview

The CRU is an APB slave module that is designed for generating all of the internal and system clocks, resets of chip. CRU generates system clock from PLL output clock or external clock source, and generates system reset from external power-on-reset, watchdog timer reset or software reset.

CRU supports the following features:

- Compliance to the AMBA APB interface
- Embedded one SOC PLL
- Support only one crystal
- Flexible selection of clock source
- Supports the respective gating of all clocks
- Supports the respective software reset of all modules

3.2 Block Diagram

The CRU comprises with:

- One SOC PLL
- Register configuration unit
- Clock generate unit
- Reset generate unit

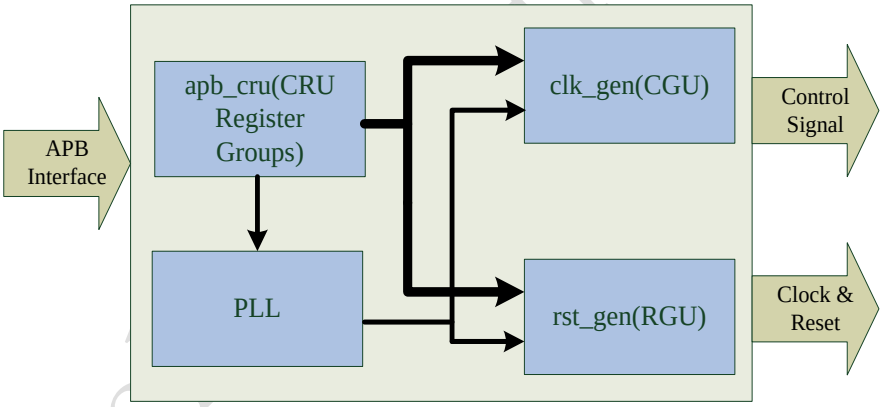


Fig 3-1 CRU Architecture

3.3 System Clock Solution

The following diagrams show clock architecture (mux and divider information).

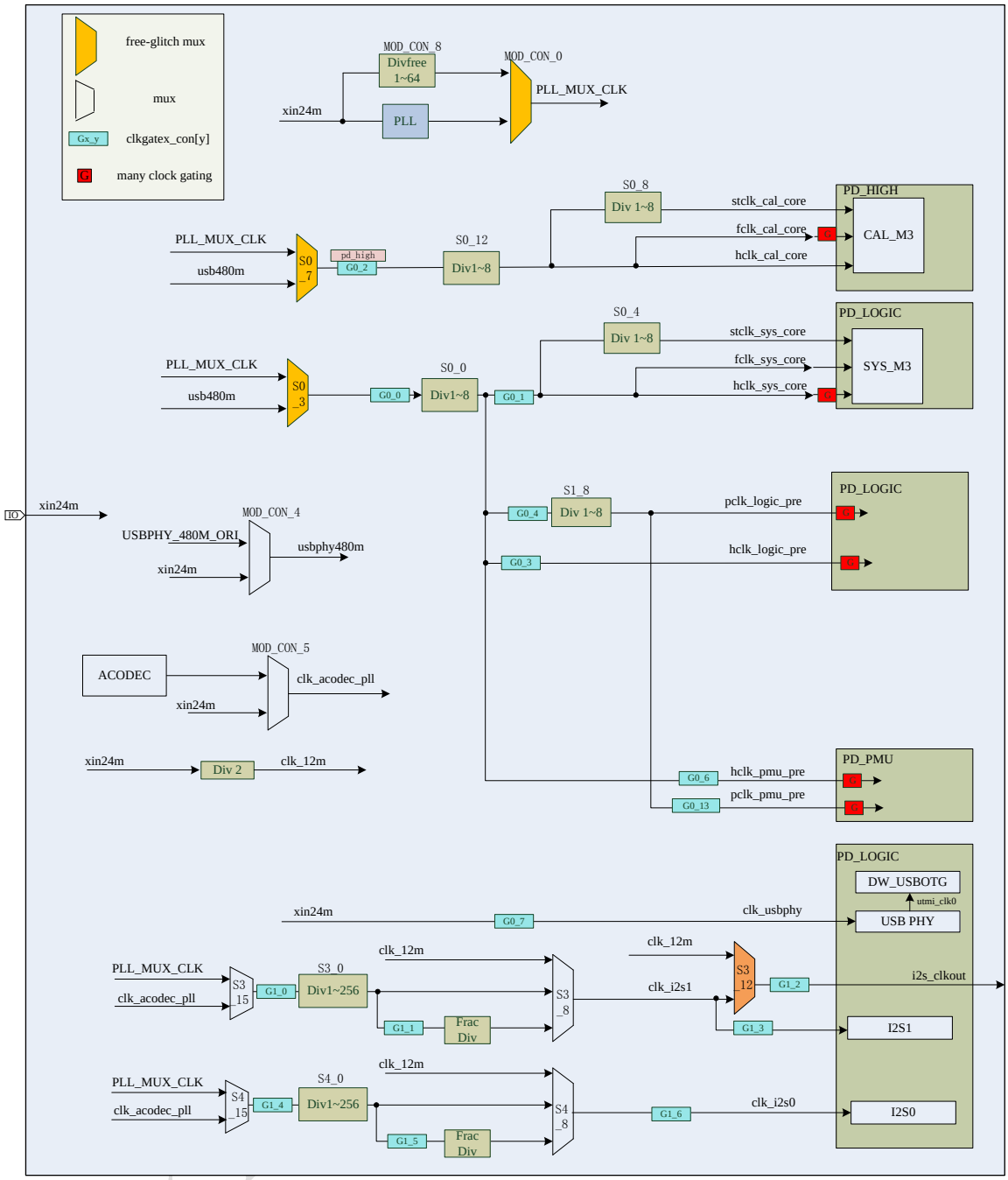


Fig 3-2 Chip Clock Architecture Diagram 1

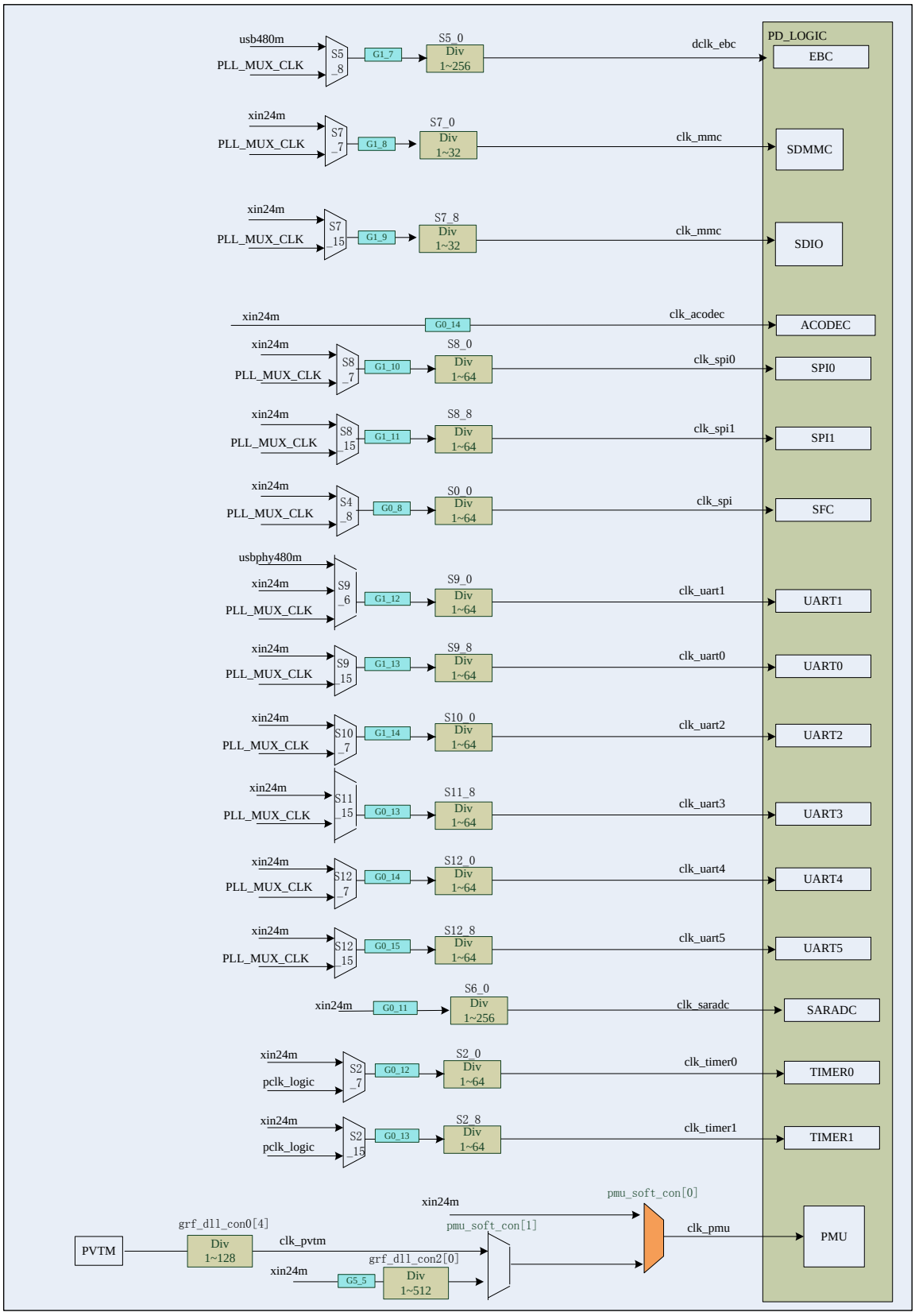


Fig 3-3 Chip Clock Architecture Diagram 2

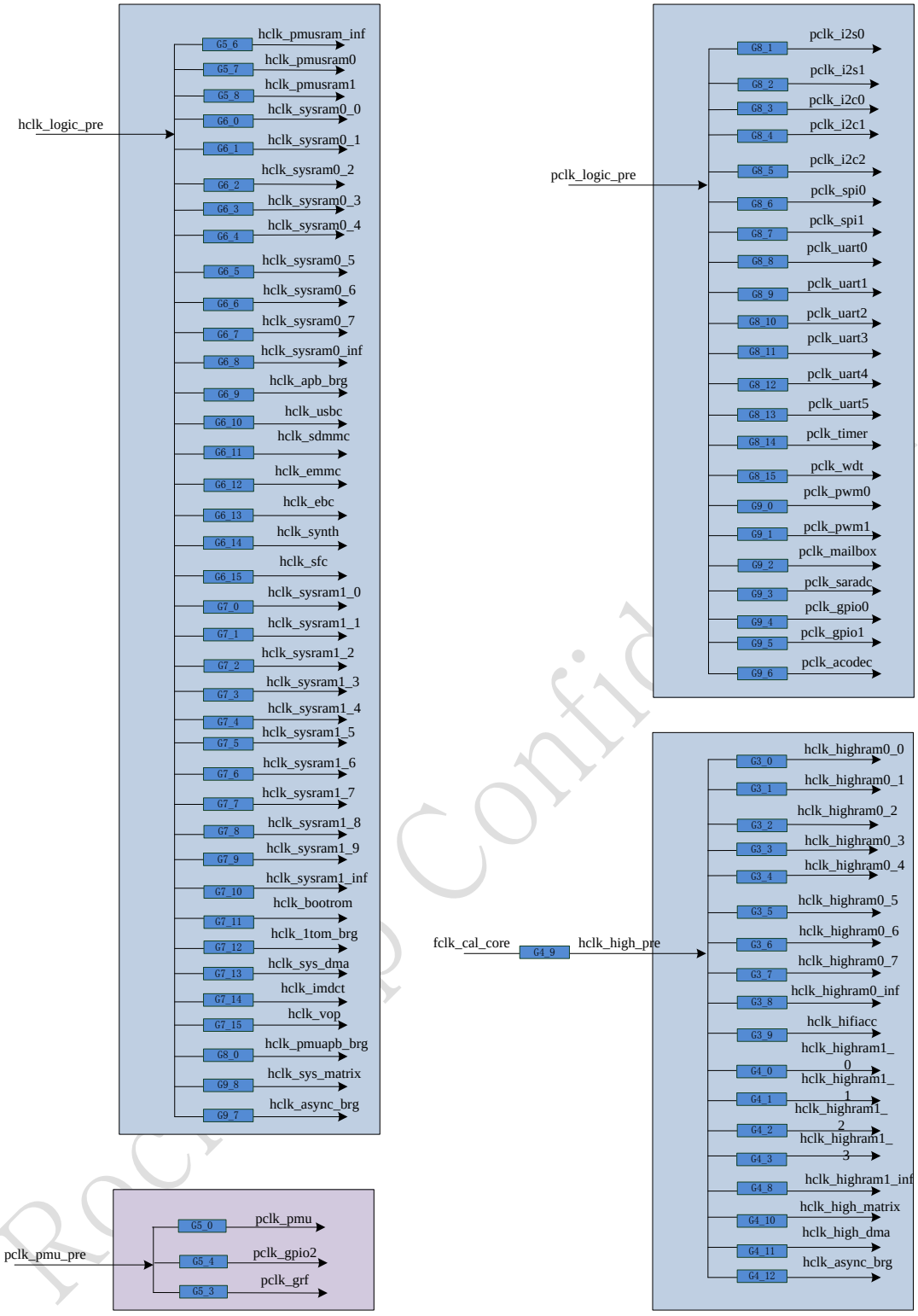


Fig 3-4 Chip Clock Architecture Diagram 3

Description about input clock

The source of input clock in upper diagrams is listed as following Table.

Table 3-1 Input clock description in clock architecture diagram

Input Clock	Source
xin24m	External crystal oscillator (24MHz)

3.4 System Reset Solution

The following diagrams show reset architecture in this device.

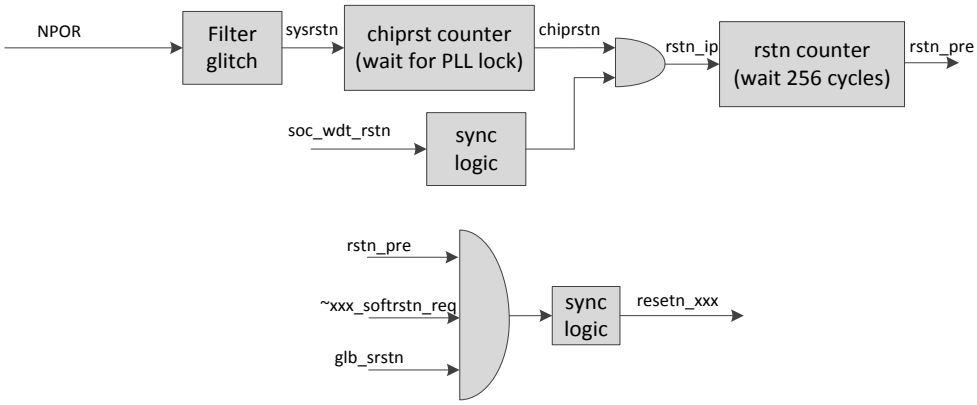


Fig 3-5 Reset Architecture Diagram

Reset source of each reset signal includes hardware reset(NPOR), soc watch dog reset(soc_wdt_rstn), software reset request(xxx_softrstn_req), global software reset(glb_srstn).

The 'xxx' of resetn_xxx and xxx_softrstn_req is the module name.

Soc_wdt_rstn is the reset from watch-dog IP in the SoC.

Glb_srstn is the global software reset by programming CRU register. When register CRU_GLB_SRST_FST_VALUE is set as 0xfdb9, glb_srstn will be asserted. The software reset will be self-clear by hardware. Glb_srstn will reset the all logic except GRF and all GPIOs.

3.5 Function Description

There are one PLL in the chip, and it supports only one 24MHz crystal oscillator. The PLL can be set to slow mode or deep slow mode. When power on or changing PLL setting, we must force PLL into slow mode to ensure output stable clock.

To maximize the flexibility, some of clocks can select divider source from 24MHz and PLL. To provide some specific frequency, another solution is integrated: fractional divider. In order to be sure the performance for divided clock, there is some usage limit, we can only get low frequency and divider factor must be larger than 20.

All clocks can be software gated and all reset can be software generated.

3.6 PLL Introduction

Overview

The PLL is a general purpose, high-performance PLL-based clock generator. The PLL is a multi-function, general purpose frequency synthesizer.

Block diagram

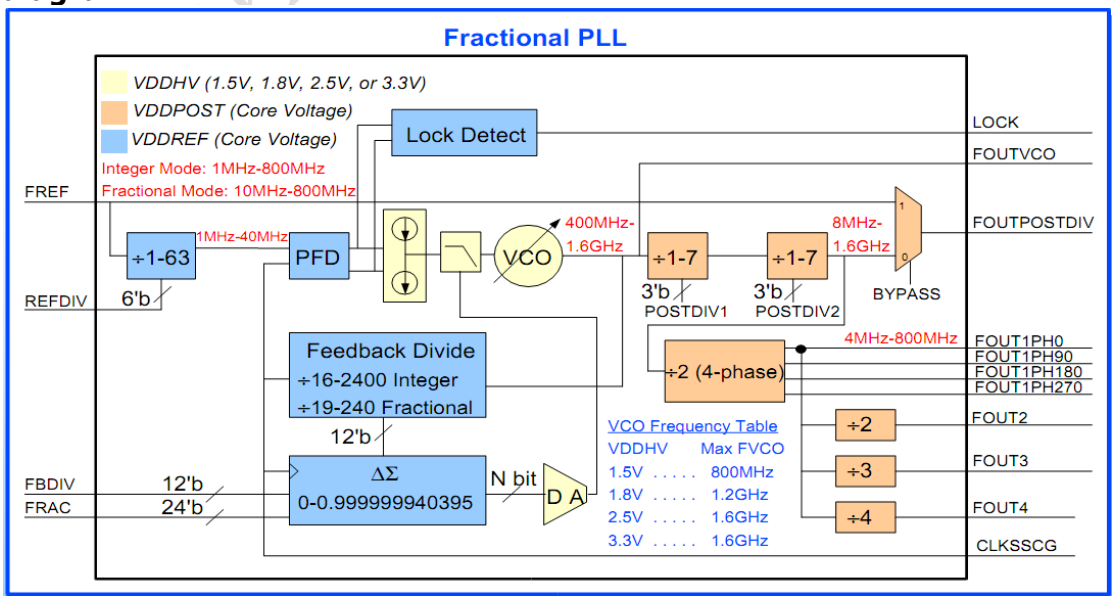


Fig 3-6PLL Block Diagram

How to calculate the PLL

The Fractional PLL output frequency can be calculated using some simple formulas. These formulas also embedded within the Fractional PLL Verilog model:

If DSMPD = 1 (DSM is disabled, "integer mode")

FOUTVCO = FREF / REFDIV * FBDIV

FOUTPOSTDIV = FOUTVCO / POSTDIV1 / POSTDIV2

If DSMPD = 0 (DSM is enabled, "fractional mode")

FOUTVCO = FREF / REFDIV * (FBDIV + FRAC / 2^{24})

FOUTPOSTDIV = FOUTVCO / POSTDIV1 / POSTDIV2

Where:

FOUTVCO = Fractional PLL non-divided output frequency

FOUTPOSTDIV = Fractional PLL divided output frequency (output of second post divider)

FREF = Fractional PLL input reference frequency

REFDIV = Fractional PLL input reference clock divider

FVCO = Frequency of internal VCO

FBDIV = Integer value programmed into feedback divide

FRAC = Fractional value programmed into DSM

Changing the PLL Programming

In most cases the PLL programming can be changed on-the-fly and the PLL will simply slew to the new frequency. However, certain changes have the potential to cause glitches on the PLL output clocks. These changes include:

- Switching into or out of BYPASS mode may cause a glitch on FOUTPOSTDIV
- Changing POSTDIV1 or POSTDIV2 may cause a short pulse with width equal to as little as one VCO period on FOUTPOSTDIV
- Changing POSTDIV could cause a shortened pulse on FOUT1PH* or FOUT2/3/4
- Asserting PD or FOUTPOSTDIVPD may cause a glitch on FOUTPOSTDIV
- Asserting PD or FOUTPOSTDIVPD may cause a glitch on FOUTPOSTDIV

3.7 Register Description

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

3.7.1 Registers Summary

Name	Offset	Size	Reset Value	Description
CRU_APLL_CON0	0x00000	W	0x00004064	ARM PLL control register0
CRU_APLL_CON1	0x00004	W	0x00001042	ARM PLL control register1
CRU_APLL_CON2	0x00008	W	0x00000001	ARM PLL control register2
CRU_MODE_CON	0x00010	W	0x00000000	System work mode control register
CRU_CLKSEL0_CON	0x00014	W	0x00000331	Internal clock select and divide register
CRU_CLKSEL1_CON	0x00018	W	0x00000100	Internal clock select and divide register
CRU_CLKSEL2_CON	0x0001c	W	0x00000000	Internal clock select and divide register
CRU_CLKSEL3_CON	0x00020	W	0x00000007	Internal clock select and divide register
CRU_CLKSEL4_CON	0x00024	W	0x00000007	Internal clock select and divide register

Name	Offset	Size	Reset Value	Description
CRU_CLKSEL5_CON	0x00028	W	0x0000000f	Internal clock select and divide register
CRU_CLKSEL6_CON	0x0002c	W	0x0000001f	Internal clock select and divide register
CRU_CLKSEL7_CON	0x00030	W	0x00000707	Internal clock select and divide register
CRU_CLKSEL8_CON	0x00034	W	0x00000707	Internal clock select and divide register
CRU_CLKSEL9_CON	0x00038	W	0x00000707	Internal clock select and divide register
CRU_CLKSEL10_CON	0x0003c	W	0x00000307	Internal clock select and divide register
CRU_CLKSEL11_CON	0x00040	W	0x00000700	Internal clock select and divide register
CRU_CLKSEL12_CON	0x00044	W	0x00000707	Internal clock select and divide register
CRU_CLK_FRACDIV_CON0	0x00050	W	0x0bb8ea60	Internal clock frac divide register0
CRU_CLK_FRACDIV_CON1	0x00054	W	0x0bb8ea60	Internal clock frac divide register1
CRU_CLKGATE0_CON	0x00080	W	0x00000000	Internal clock gating control register
CRU_CLKGATE1_CON	0x00084	W	0x00000000	Internal clock gating control register
CRU_CLKGATE2_CON	0x00088	W	0x00000000	Internal clock gating control register
CRU_CLKGATE3_CON	0x0008c	W	0x00000000	Internal clock gating control register
CRU_CLKGATE4_CON	0x00090	W	0x00000000	Internal clock gating control register
CRU_CLKGATE5_CON	0x00094	W	0x00000000	Internal clock gating control register
CRU_CLKGATE6_CON	0x00098	W	0x00000000	Internal clock gating control register
CRU_CLKGATE7_CON	0x0009c	W	0x00000000	Internal clock gating control register
CRU_CLKGATE8_CON	0x000a0	W	0x00000000	Internal clock gating control register
CRU_CLKGATE9_CON	0x000a4	W	0x00000000	Internal clock gating control register
CRU_SOFTRST0_CON	0x000c0	W	0x00000000	Internal software reset control register0
CRU_SOFTRST1_CON	0x000c4	W	0x00000000	Internal software reset control register0

Name	Offset	Size	Reset Value	Description
CRU_SOFTRST2_CON	0x000c8	W	0x00000000	Internal software reset control register0
CRU_SOFTRST3_CON	0x000cc	W	0x00000004	Internal software reset control register0
CRU_STCLK_CON0	0x000e0	W	0x00000001	stick clk counter0
CRU_STCLK_CON1	0x000e4	W	0x00000001	stick clk counter1
CRU_GLB_SRST_FST_VALUE	0x000f4	W	0x00000000	The first global software reset config value
CRU_GLB_CNT_TH	0x000f8	W	0x00000064	global reset wait counter threshold

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

3.7.2 Detail Register Description

CRU_APLL_CON0

Address: Operational Base + offset (0x00000)
ARM PLL control register0

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	bit_write_mask bit_write_mask control corresponding (bit_write_mask - 16) configuration bit 0: mask 1: unmask
15	RW	0x0	bp pll bypass
14:12	RW	0x4	postdiv1 PLL factor postdiv1
11:0	RW	0x064	fbdiv PLL factor fbdiv

CRU_APLL_CON1

Address: Operational Base + offset (0x00004)
ARM PLL control register1

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	bit_write_mask bit_write_mask control corresponding (bit_write_mask - 16) configuration bit 0: mask 1: unmask
15	RW	0x0	rstmode PLL Reset select 0 : internal reset 1 : software reset
14	RW	0x0	powr_down PLL Software pd 0 : normal 1 : power_down

Bit	Attr	Reset Value	Description
13	RO	0x0	reserved
12	RW	0x1	dsmpd when 1, PLL work at integer mode when 0, PLL work at frac mode
11	RO	0x0	reserved
10	RO	0x0	lock PLL lock status
9	RO	0x0	reserved
8:6	RW	0x1	postdiv2 PLL factor postdiv2
5:0	RW	0x02	refdiv PLL factor refdiv

CRU_APLL_CON2
Address: Operational Base + offset (0x00008)
ARM PLL control register2

Bit	Attr	Reset Value	Description
31:28	RO	0x0	reserved
27	RW	0x0	fout4phasepd 4 phase clock power down，active high
26	RW	0x0	foutvcopd buffered VCO clock power down，active high
25	RW	0x0	foutpostdivpd post divide power down，active high
24	RW	0x0	dacpd PLL cancellation DAC power down, active high
23:0	RW	0x000001	frac PLL factor frac

CRU_MODE_CON
Address: Operational Base + offset (0x00010)
System work mode control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:14	RO	0x0	reserved
13:8	RW	0x00	div_con_24m 24m div control select clk=clk_src/(div_con+1)
7:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5	RW	0x0	acodec_pll_sel acodec pll clock select 0: clock from external 24MHz OSC 1: clock from acodec pll output
4	RW	0x0	usb480m_sel usbphy 480M clock select 0: clock from external 24MHz OSC 1: clock from usbphy 480M output
3:1	RO	0x0	reserved
0	RW	0x0	apll_work_mode ARM PLL work mode select 0: Slow mode, clock from external 24MHz OSC 1: Normal mode, clock from PLL output

CRU_CLKSEL0_CON
Address: Operational Base + offset (0x00014)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	cal_core_src_sel cal core source clock select 1'b0: select PLL 1'b1: select usbphy480M
14:12	RW	0x0	cal_core_div_con cal core clock divider frequency clk=clk_src/(div_con+1)
11	RO	0x0	reserved
10:8	RW	0x3	cal_stclk_div_con core 1 stclk divider frequency clk=clk_src/(div_con+1)
7	RO	0x0	reserved
6:4	RW	0x3	sys_stclk_div_con system core stclk divider frequency clk=clk_src/(div_con+1)
3	RW	0x0	sys_core_src_sel system core source clock select 1'b0: select PLL 1'b1: select usbphy480M

Bit	Attr	Reset Value	Description
2:0	RW	0x1	sys_core_div_con system core clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL1_CON
Address: Operational Base + offset (0x00018)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9:8	RW	0x1	pclk_logic_div_con pd_logic apb clock divider frequency 2'd0: div1 2'd1: div2 2'd2: div4 2'd3: div8
7:0	RO	0x0	reserved

CRU_CLKSEL2_CON
Address: Operational Base + offset (0x0001c)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	timer1_pll_sel timer1 clock source selection 1'b0: select 24M 1'b1: select pclk_logic
14:13	RO	0x0	reserved
12:8	RW	0x00	timer1_div_con timer0 clock divider frequency clk=clk_src/(div_con+1)
7	RW	0x0	timer0_pll_sel timer0 clock source selection 1'b0: select 24M 1'b1: select pclk_logic

Bit	Attr	Reset Value	Description
6:5	RO	0x0	reserved
4:0	RW	0x00	timer0_div_con timer0 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL3_CON
Address: Operational Base + offset (0x00020)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	i2s1_pll_sel i2s1 output clock selection 1'b0: select soc pll 1'b1: select acodec pll
14:13	RO	0x0	reserved
12	RW	0x0	i2s1_out_sel i2s1 output clock selection 1'b0: select clk_i2s1 1'b1: select 12M
11:10	RO	0x0	reserved
9:8	RW	0x0	i2s1_clk_sel i2s1 source selection 2'b00: select pll clock 2'b01: select frac_div output 2'b10: select 12M
7	RO	0x0	reserved
6:0	RW	0x07	i2s1_div_con Control I2S1 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL4_CON
Address: Operational Base + offset (0x00024)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

Bit	Attr	Reset Value	Description
15	RW	0x0	i2s0_pll_sel i2s0 output clock selection 1'b0: select soc pll 1'b1: select acodec pll
14:10	RO	0x0	reserved
9:8	RW	0x0	i2s0_clk_sel i2s0 source selection 2'b00: select pll clock 2'b01: select frac_div output 2'b10: select 12M
7	RO	0x0	reserved
6:0	RW	0x07	i2s0_div_con Control I2S0 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL5_CON
Address: Operational Base + offset (0x00028)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:9	RO	0x0	reserved
8	RW	0x0	ebc_dclk_pll_sel ebc dclk pll source selection 1'b0: select PLL 1'b1: select usbphy 480M
7:0	RW	0x0f	ebc_dclk_div_con Control ebc clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL6_CON
Address: Operational Base + offset (0x0002c)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:0	RW	0x1f	saradc_div_con Control saradc clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL7_CON
Address: Operational Base + offset (0x00030)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	emmc_pll_sel emmc pll source selection 1'b0: select PLL 1'b1: select 24m
14:13	RO	0x0	reserved
12:8	RW	0x07	emmc_div_con emmc clock divider frequency clk=clk_src/(div_con+1)
7	RW	0x0	sdmmc_clk_pll_sel sdmmc clock pll source selection 1'b0: select PLL 1'b1: select 24m
6:5	RO	0x0	reserved
4:0	RW	0x07	sdmmc_div_con Control sdmmc clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL8_CON
Address: Operational Base + offset (0x00034)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	spi1_clk_pll_sel spi1 pll source selection 1'b0: select PLL 1'b1: select 24m

Bit	Attr	Reset Value	Description
14	RO	0x0	reserved
13:8	RW	0x07	spi1_div_con spi1 clock divider frequency clk=clk_src/(div_con+1)
7	RW	0x0	spi0_clk_pll_sel spi0 clock pll source selection 1'b0: select PLL 1'b1: select 24m
6	RO	0x0	reserved
5:0	RW	0x07	spi0_div_con spi0 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL9_CON
Address: Operational Base + offset (0x00038)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	uart0_pll_sel uart0 pll source selection 1'b0: select PLL 1'b1: select 24m
14	RO	0x0	reserved
13:8	RW	0x07	uart0_div_con uart0 clock divider frequency clk=clk_src/(div_con+1)
7:6	RW	0x0	uart1_pll_sel uart1 source selection 2'b00: select PLL 2'b01: select 24m 2'b10: select usb480m
5:0	RW	0x07	uart1_div_con Control uart1 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL10_CON
Address: Operational Base + offset (0x0003c)
Internal clock select and divide register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	sfc_pll_sel sfc pll source selection 1'b0: select PLL 1'b1: select 24m
14:13	RO	0x0	reserved
12:8	RW	0x03	sfc_div_con sfc clock divider frequency clk=clk_src/(div_con+1)
7	RW	0x0	uart2_pll_sel uart2 pll source selection 1'b0: select PLL 1'b1: select 24m
6	RO	0x0	reserved
5:0	RW	0x07	uart2_div_con Control uart2 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL11_CON
Address: Operational Base + offset (0x00040)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	uart3_pll_sel uart3 source selection 1'b0: select PLL 1'b1: select 24m
14	RO	0x0	reserved
13:8	RW	0x07	uart3_div_con uart3 clock divider frequency clk=clk_src/(div_con+1)

Bit	Attr	Reset Value	Description
7:5	RW	0x0	core_clk_pll_sel core clock pll source selection 3'b000: select clk_acodec 3'b001: select clk_pvtm 3'b010: select clk_sys_core 3'b011: select clk_cal_core 3'b100: select clk_sdmmc 3'b101: select pclk_logic 3'b110: select clk_i2s0 3'b111: select clk_i2s1
4:0	RW	0x00	test_div_con Control test output clock divider frequency clk=clk_src/(div_con+1)

CRU_CLKSEL12_CON
Address: Operational Base + offset (0x00044)
Internal clock select and divide register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	uart5_pll_sel uart5 pll source selection 1'b0: select PLL 1'b1: select 24m
14	RO	0x0	reserved
13:8	RW	0x07	uart5_div_con uart5 clock divider frequency clk=clk_src/(div_con+1)
7	RW	0x0	uart4_pll_sel uart4 source selection 1'b0: select PLL 1'b1: select 24m
6	RO	0x0	reserved
5:0	RW	0x07	uart4_div_con uart4 clock divider frequency clk=clk_src/(div_con+1)

CRU_CLK_FRACDIV_CON0
Address: Operational Base + offset (0x00050)
Internal clock frac divide register0

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x0bb8ea60	i2s0_frac_factor Control i2s0 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

CRU_CLK_FRACDIV_CON1
Address: Operational Base + offset (0x00054)
Internal clock frac divide register1

Bit	Attr	Reset Value	Description
31:0	RW	0x0bb8ea60	i2s1_frac_factor Control i2s1 fraction divider frequency High 16-bit for numerator Low 16-bit for denominator

CRU_CLKGATE0_CON
Address: Operational Base + offset (0x00080)
Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RO	0x0	reserved
14	RW	0x0	clk_acodec_gate_en clock acodec disable. When HIGH, disable clock
13	RW	0x0	pclk_pmu_gate_en pd_pmu apb clock disable. When HIGH, disable clock
12	RW	0x0	clk_timer1_gate_en clock timer1 disable. When HIGH, disable clock
11	RW	0x0	clk_timer0_gate_en clock timer0 disable. When HIGH, disable clock
10	RW	0x0	clk_saradc_gate_en clock saradc disable. When HIGH, disable clock
9	RO	0x0	reserved
8	RW	0x0	clk_sfc_gate_en clock sfc disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
7	RW	0x0	clk_usbphy_gate_en clock usbphy disable. When HIGH, disable clock
6	RW	0x0	hclk_pmu_gate_en pd_pmu ahb clock disable. When HIGH, disable clock
5	RO	0x0	reserved
4	RW	0x0	pclk_logic_gate_en pd_logic ahb clock disable. When HIGH, disable clock
3	RW	0x0	hclk_logic_gate_en pd_logic ahb clock disable. When HIGH, disable clock
2	RW	0x0	hclk_cal_core_gate_en hclk_cal_core disable. When HIGH, disable clock
1	RW	0x0	hclk_sys_core_gate_en hclk_sys_core disable. When HIGH, disable clock
0	RW	0x0	clk_sys_core_gate_en clock system core disable. When HIGH, disable clock

CRU_CLKGATE1_CON
Address: Operational Base + offset (0x00084)
Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	clk_test_gate_en clk test out disable. When HIGH, disable clock
14	RW	0x0	clk_uart2_gate_en clk uart2 disable. When HIGH, disable clock
13	RW	0x0	clk_uart1_gate_en clk uart1 source disable. When HIGH, disable clock
12	RW	0x0	clk_uart0_gate_en clk uart0 disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
11	RW	0x0	clk_spi1_gate_en clk spi1 source disable. When HIGH, disable clock
10	RW	0x0	clk_spi0_gate_en clk spi0 source disable. When HIGH, disable clock
9	RW	0x0	clk_emmc_gate_en clk emmc disable. When HIGH, disable clock
8	RW	0x0	clk_sdmmc_gate_en clk sdmmc disable. When HIGH, disable clock
7	RW	0x0	dclk_ebc_src_gate_en clk ebc source disable. When HIGH, disable clock
6	RW	0x0	clk_i2s_gate_en clk i2s source disable. When HIGH, disable clock
5	RW	0x0	clk_i2s_frac_src_gate_en clk i2s frac div source disable. When HIGH, disable clock
4	RW	0x0	clk_i2s0_src_gate_en clk i2s0 source disable. When HIGH, disable clock
3	RW	0x0	clk_i2s1_gate_en clk i2s disable. When HIGH, disable clock
2	RW	0x0	clk_i2s1_out_gate_en clk i2s output disable. When HIGH, disable clock
1	RW	0x0	clk_i2s1_frac_src_gate_en clk i2s1 frac div source disable. When HIGH, disable clock
0	RW	0x0	clk_i2s1_src_gate_en clk i2s1 source disable. When HIGH, disable clock

CRU_CLKGATE2_CON

Address: Operational Base + offset (0x00088)

Internal clock gating control register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:3	RO	0x0	reserved
2	RW	0x0	clk_uart5_gate_en clk uart5 disable. When HIGH, disable clock
1	RW	0x0	clk_uart4_gate_en clk uart4 disable. When HIGH, disable clock
0	RW	0x0	clk_uart3_gate_en clk uart3 disable. When HIGH, disable clock

CRU_CLKGATE3_CON
Address: Operational Base + offset (0x0008c)
Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9	RW	0x0	hclk_hifi_gate_en hclk_hifi disable. When HIGH, disable clock
8	RW	0x0	hdram_interface_gate_en hdram interface disable. When HIGH, disable clock
7	RW	0x0	hdram7_gate_en hdram7 disable. When HIGH, disable clock
6	RW	0x0	hdram6_gate_en hdram6 disable. When HIGH, disable clock
5	RW	0x0	hdram5_gate_en hdram5 disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
4	RW	0x0	hdram4_gate_en hdram4 disable. When HIGH, disable clock
3	RW	0x0	hdram3_gate_en hdram3 disable. When HIGH, disable clock
2	RW	0x0	hdram2_gate_en hdram2 disable. When HIGH, disable clock
1	RW	0x0	hdram1_gate_en hdram1 disable. When HIGH, disable clock
0	RW	0x0	hdram0_gate_en hdram0 disable. When HIGH, disable clock

CRU_CLKGATE4_CON

Address: Operational Base + offset (0x00090)

Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:13	RO	0x0	reserved
12	RW	0x0	hclk_async_brg_gate_en pd_high async bridge hclk disable. When HIGH, disable clock
11	RW	0x0	hclk_high_dma_gate_en pd_high DMA hclk disable. When HIGH, disable clock
10	RW	0x0	hclk_high_matrix_gate_en pd_high bus_matrix disable. When HIGH, disable clock
9	RW	0x0	hclk_high_gate_en hclk_high_pre disable. When HIGH, disable clock
8	RW	0x0	hiram_interface_gate_en hiram interface disable. When HIGH, disable clock
7:4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3	RW	0x0	hiram3_gate_en hiram3 disable. When HIGH, disable clock
2	RW	0x0	hiram2_gate_en hiram2 disable. When HIGH, disable clock
1	RW	0x0	hiram1_gate_en hiram1 disable. When HIGH, disable clock
0	RW	0x0	hiram0_gate_en hiram0 disable. When HIGH, disable clock

CRU_CLKGATE5_CON

Address: Operational Base + offset (0x00094)

Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:11	RO	0x0	reserved
10	RW	0x0	pvtm_clk_gate_en clk_pvtm disable. When HIGH, disable clock
9	RO	0x0	reserved
8	RW	0x0	pmu_ram1_gate_en pmu_ram1 disable. When HIGH, disable clock
7	RW	0x0	pmu_ram0_gate_en pmu_ram0 disable. When HIGH, disable clock
6	RW	0x0	pmu_ram_interface_gate_en pmu_ram interface disable. When HIGH, disable clock
5	RW	0x0	clk24m_gate_en clk24m disable. When HIGH, disable clock
4	RW	0x0	pclk_gpio2_gate_en pclk_gpio2 disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
3	RW	0x0	pclk_grf_gate_en pclk_grf disable. When HIGH, disable clock
2:1	RO	0x0	reserved
0	RW	0x0	pclk_pmu_gate_en pmu apb bus clock disable. When HIGH, disable clock

CRU_CLKGATE6_CON

Address: Operational Base + offset (0x00098)

Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	hclk_sfc_gate_en hclk_sfc disable. When HIGH, disable clock
14	RW	0x0	hclk_synth_gate_en hclk_synth disable. When HIGH, disable clock
13	RW	0x0	hclk_ebc_gate_en hclk_ebc disable. When HIGH, disable clock
12	RW	0x0	hclk_emmc_gate_en hclk_emmc disable. When HIGH, disable clock
11	RW	0x0	hclk_sdmmc_gate_en hclk_sdmmc disable. When HIGH, disable clock
10	RW	0x0	hclk_usbc_gate_en hclk usb controller disable. When HIGH, disable clock
9	RW	0x0	hclk_apb_brg_gate_en hclk_apb_brg disable. When HIGH, disable clock
8	RW	0x0	dram_interface_gate_en dram interface disable. When HIGH, disable clock
7	RW	0x0	dram7_gate_en dram7 disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
6	RW	0x0	dram6_gate_en dram6 disable. When HIGH, disable clock
5	RW	0x0	dram5_gate_en dram5 disable. When HIGH, disable clock
4	RW	0x0	dram4_gate_en dram4 disable. When HIGH, disable clock
3	RW	0x0	dram3_gate_en dram3 disable. When HIGH, disable clock
2	RW	0x0	dram2_gate_en dram2 disable. When HIGH, disable clock
1	RW	0x0	dram1_gate_en dram1 disable. When HIGH, disable clock
0	RW	0x0	dram0_gate_en dram0 disable. When HIGH, disable clock

CRU_CLKGATE7_CON
Address: Operational Base + offset (0x0009c)
Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	hclk_lcdc_gate_en hclk_lcdc disable. When HIGH, disable clock
14	RW	0x0	hclk_imdct_gate_en hclk_imdct disable. When HIGH, disable clock
13	RW	0x0	hclk_dma_gate_en hclk_dma disable. When HIGH, disable clock
12	RW	0x0	hclk_1tom_brg_gate_en hclk_1tom_brg disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
11	RW	0x0	hclk_ahb_brg_gate_en hclk_ahb_brg disable. When HIGH, disable clock
10	RW	0x0	hclk_bus_matrix_gate_en hclk_bus_matrix disable. When HIGH, disable clock
9	RW	0x0	hclk_bootrom_gate_en hclk_bootrom disable. When HIGH, disable clock
8	RW	0x0	iram_interface_gate_en iram interface disable. When HIGH, disable clock
7	RW	0x0	iram7_gate_en iram7 disable. When HIGH, disable clock
6	RW	0x0	iram6_gate_en iram6 disable. When HIGH, disable clock
5	RW	0x0	iram5_gate_en iram5 disable. When HIGH, disable clock
4	RW	0x0	iram4_gate_en iram4 disable. When HIGH, disable clock
3	RW	0x0	iram3_gate_en iram3 disable. When HIGH, disable clock
2	RW	0x0	iram2_gate_en iram2 disable. When HIGH, disable clock
1	RW	0x0	iram1_gate_en iram1 disable. When HIGH, disable clock
0	RW	0x0	iram0_gate_en iram0 disable. When HIGH, disable clock

CRU_CLKGATE8_CON

Address: Operational Base + offset (0x000a0)

Internal clock gating control register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	pclk_wdt_gate_en pclk_wdt disable. When HIGH, disable clock
14	RW	0x0	pclk_timer_gate_en pclk_timer disable. When HIGH, disable clock
13	RW	0x0	pclk_uart5_gate_en pclk_uart5 disable. When HIGH, disable clock
12	RW	0x0	pclk_uart4_gate_en pclk_uart4 disable. When HIGH, disable clock
11	RW	0x0	pclk_uart3_gate_en pclk_uart3 disable. When HIGH, disable clock
10	RW	0x0	pclk_uart2_gate_en pclk_uart2 disable. When HIGH, disable clock
9	RW	0x0	pclk_uart1_gate_en pclk_uart1 disable. When HIGH, disable clock
8	RW	0x0	pclk_uart0_gate_en pclk_uart0 disable. When HIGH, disable clock
7	RW	0x0	pclk_spi1_gate_en pclk_spi1 disable. When HIGH, disable clock
6	RW	0x0	pclk_spi0_gate_en pclk_spi0 disable. When HIGH, disable clock
5	RW	0x0	pclk_i2c2_gate_en pclk_i2c2 disable. When HIGH, disable clock
4	RW	0x0	pclk_i2c1_gate_en pclk_i2c1 disable. When HIGH, disable clock
3	RW	0x0	pclk_i2c0_gate_en pclk_i2c0 disable. When HIGH, disable clock

Bit	Attr	Reset Value	Description
2	RW	0x0	pclk_i2s1_gate_en pclk_i2s1 disable. When HIGH, disable clock
1	RW	0x0	pclk_i2s0_gate_en pclk_i2s0 disable. When HIGH, disable clock
0	RW	0x0	hclk_pmu_apb_brg_gate_en apb bridge to pd_pmu hclk disable. When HIGH, disable clock

CRU_CLKGATE9_CON

Address: Operational Base + offset (0x000a4)

Internal clock gating control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:9	RO	0x0	reserved
8	RW	0x0	pclk_lgc_matrix_gate_en pd_logic bus matrix pclk disable. When HIGH, disable clock
7	RW	0x0	pclk_async_brg_gate_en pclk_async_brg disable. When HIGH, disable clock
6	RW	0x0	pclk_acodec_gate_en pclk_acodec disable. When HIGH, disable clock
5	RW	0x0	pclk_gpio1_gate_en pclk_gpio1 disable. When HIGH, disable clock
4	RW	0x0	pclk_gpio0_gate_en pclk_gpio0 disable. When HIGH, disable clock
3	RW	0x0	pclk_saradc_gate_en pclk_saradc disable. When HIGH, disable clock
2	RW	0x0	pclk_mailbox_gate_en pclk_mailbox disable. When HIGH, disable clock
1	RW	0x0	pclk_pwm1_gate_en pclk_pwm1 disable. When HIGH, disable clock

0	RW	0x0	pclk_pwm0_gate_en pclk_pwm0 disable. When HIGH, disable clock

CRU_SOFTRST0_CON

Address: Operational Base + offset (0x000c0)

Internal software reset control register0

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15	RW	0x0	synth_srst_sel synth software reset 1'b0: not reset 1'b1: reset
14	RW	0x0	imdct_srst_sel imdct software reset 1'b0: not reset 1'b1: reset
13	RW	0x0	lcdc_srst_sel lcdc software reset 1'b0: not reset 1'b1: reset
12	RW	0x0	sysram0_srst_sel sysram0 software reset 1'b0: not reset 1'b1: reset
11	RW	0x0	sysram1_srst_sel sysram1 software reset 1'b0: not reset 1'b1: reset
10	RW	0x0	bootrom_srst_sel boot rom software reset 1'b0: not reset 1'b1: reset
9	RW	0x0	uart2_srst_sel uart2 software reset 1'b0: not reset 1'b1: reset
8	RW	0x0	uart1_srst_sel uart1 software reset 1'b0: not reset 1'b1: reset

Bit	Attr	Reset Value	Description
7	RW	0x0	uart0_srst_sel uart0 software reset 1'b0: not reset 1'b1: reset
6	RW	0x0	spi1_srst_sel spi1 software reset 1'b0: not reset 1'b1: reset
5	RW	0x0	spi0_srst_sel spi0 software reset 1'b0: not reset 1'b1: reset
4	RW	0x0	i2s1_srst_sel i2s1 software reset 1'b0: not reset 1'b1: reset
3	RW	0x0	i2s0_srst_sel i2s0 software reset 1'b0: not reset 1'b1: reset
2	RW	0x0	usbglb_srst_sel usb glb software reset 1'b0: not reset 1'b1: reset
1	RW	0x0	usbphy_srst_sel usb phy software reset 1'b0: not reset 1'b1: reset
0	RW	0x0	usbotg_srst_sel usb otg ctrl software reset 1'b0: not reset 1'b1: reset

CRU_SOFTRST1_CON
Address: Operational Base + offset (0x000c4)
Internal software reset control register0

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit

Bit	Attr	Reset Value	Description
15	RW	0x0	wdt_srst_sel wdt software reset 1'b0: not reset 1'b1: reset
14	RW	0x0	mailbox_srst_sel mailbox software reset 1'b0: not reset 1'b1: reset
13	RW	0x0	ebc_srst_sel ebc software reset 1'b0: not reset 1'b1: reset
12	RW	0x0	i2c2_srst_sel i2c2 software reset 1'b0: not reset 1'b1: reset
11	RW	0x0	i2c1_srst_sel i2c1 software reset 1'b0: not reset 1'b1: reset
10	RW	0x0	i2c0_srst_sel i2c0 software reset 1'b0: not reset 1'b1: reset
9	RW	0x0	sfc_srst_sel sfc software reset 1'b0: not reset 1'b1: reset
8	RW	0x0	pwm1_srst_sel pwm1 software reset 1'b0: not reset 1'b1: reset
7	RW	0x0	pwm0_srst_sel pwm0 software reset 1'b0: not reset 1'b1: reset
6	RW	0x0	saradc_srst_sel saradc software reset 1'b0: not reset 1'b1: reset
5	RW	0x0	timer1_srst_sel timer1 software reset 1'b0: not reset 1'b1: reset

Bit	Attr	Reset Value	Description
4	RW	0x0	timer0_srst_sel timer0 software reset 1'b0: not reset 1'b1: reset
3	RW	0x0	sysmatrix_srst_sel sys_matrix software reset 1'b0: not reset 1'b1: reset
2	RW	0x0	sdio_srst_sel sdio software reset 1'b0: not reset 1'b1: reset
1	RW	0x0	sdmmc_srst_sel sdmmc software reset 1'b0: not reset 1'b1: reset
0	RW	0x0	sysdma_srst_sel sys_dma software reset 1'b0: not reset 1'b1: reset

CRU_SOFTRST2_CON
Address: Operational Base + offset (0x000c8)
Internal software reset control register0

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9	RW	0x0	gpio1_srst_sel gpio1 software reset 1'b0: not reset 1'b1: reset
8	RW	0x0	gpio0_srst_sel pgio0 software reset 1'b0: not reset 1'b1: reset
7	RW	0x0	uart5_srst_sel uart5 software reset 1'b0: not reset 1'b1: reset

Bit	Attr	Reset Value	Description
6	RW	0x0	uart4_srst_sel uart4 software reset 1'b0: not reset 1'b1: reset
5	RW	0x0	uart3_srst_sel uart3 software reset 1'b0: not reset 1'b1: reset
4	RO	0x0	reserved
3	RW	0x0	hifi_srst_sel hifi software reset 1'b0: not reset 1'b1: reset
2	RW	0x0	highram0_srst_sel highram0 software reset 1'b0: not reset 1'b1: reset
1	RW	0x0	highram1_srst_sel highram1 software reset 1'b0: not reset 1'b1: reset
0	RW	0x0	high_matrix_srst_sel pd_high matrix software reset 1'b0: not reset 1'b1: reset

CRU_SOFTRST3_CON

Address: Operational Base + offset (0x000cc)

Internal software reset control register0

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:8	RO	0x0	reserved
7	RW	0x0	dma2_srst_sel dma2 software reset 1'b0: not reset 1'b1: reset
6	RW	0x0	async_brg_srst_sel async_brg software reset 1'b0: not reset 1'b1: reset

Bit	Attr	Reset Value	Description
5	RW	0x0	pmu_srst_sel acodec software reset 1'b0: not reset 1'b1: reset
4	RO	0x0	reserved
3	RW	0x0	brg_to_pmu_srst_sel bridge to pmu software reset 1'b0: not reset 1'b1: reset
2	RW	0x1	cal_core_srst_sel cal_core software reset 1'b0: not reset 1'b1: reset
1	RW	0x0	sys_core_srst_sel sys_core software reset 1'b0: not reset 1'b1: reset
0	RW	0x0	acodec_srst_sel acodec software reset 1'b0: not reset 1'b1: reset

CRU_STCLK_CON0
Address: Operational Base + offset (0x000e0)
stick clk counter0

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:0	RW	0x000001	stclk_counter Field0000 Abstract system core stick clk counter

CRU_STCLK_CON1
Address: Operational Base + offset (0x000e4)
stick clk counter1

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:0	RW	0x000001	stclk_counter Field0000 Abstract Field0000 Description

CRU_GLB_SRST_FST_VALUE
Address: Operational Base + offset (0x000f4)
The first global software reset config value

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
15:0	RW	0x0000	glb_srst_fst_value The first global software reset config value If config 0xfdb9, it will generate first global software reset.

CRU_GLB_CNT_TH
Address: Operational Base + offset (0x000f8)
global reset wait counter threshold

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:10	RO	0x0	reserved
9:0	RW	0x064	glb_rst_cnt_th Global soft reset counter threshold

3.8 Timing Diagram

Power on reset timing is shown as follow:

Fig 3-7Chip Power On Reset Timing Diagram

NPOR is hardware reset signal from out-chip, which is filtered glitch to obtain signal sysrstn. To make PLLs work normally, the PLL reset signal (pllrstn) must maintain high for more than 1us, and PLLs start to lock when pllrstn deassert, and the PLL max lock time is 1500 PLL REFCLK cycles. And then the system will wait about 138us, and then deactive reset signal chiprstn. The signal chiprstn is used to generate output clocks in CRU. After CRU start output clocks, the system waits again for 512cycles (21.3us) to deactive signal rstn_pre, which is used to generate power on reset of all IP.

3.9 Application Notes

3.10 PLL usage

A. PLL output frequency configuration

FBDIV,POSTDIV1, BYPASS can be configured by programming CRU_APLL_CON0, CRU_DPLL_CON0, CRU_CPLL_CON0 and CRU_GPLL_CON0.

DSMPD, REFDIV, POSTDIV2 can be configured by programming CRU_APLL_CON1, CRU_DPLL_CON1, CRU_CPLL_CON1 and CRU_GPLL_CON1.

FRAC can be configured by programming CRU_APLL_CON2, CRU_DPLL_CON2,

CRU_CPLL_CON2 and CRU_GPLL_CON2.

(1) If DSMPD = 1 (DSM is disabled, "integer mode")

FOUTVCO = FREF / REFDIV * FBDIV

FOUTPOSTDIV = FOUTVCO / POSTDIV1 / POSTDIV2

When FREF is 24MHz, and if 700MHz FOUTPOSTDIV is needed. The configuration can be:

DSMPD = 1
REFDIV = 6
FBDIV = 175
POSTDIV1=1
POSTDIV2=1

And then

FOUTVCO = FREF / REFDIV * FBDIV = 24/6*175=700
FOUTPOSTDIV = FOUTVCO / POSTDIV1 / POSTDIV2=700/1/1=700

(2) If DSMPD = 0 (DSM is enabled, "fractional mode")

FOUTVCO = FREF / REFDIV * (FBDIV + FRAC / 2^{24})
FOUTPOSTDIV = FOUTVCO / POSTDIV1 / POSTDIV2

When FREF is 24MHz, and if 491.52MHz FOUTPOSTDIV is needed. The configuration can be:

DSMPD = 0
REFDIV = 1
FBDIV = 40
FRAC = 24'hf5c28f
POSTDIV1=2
POSTDIV2=1

And then

FOUTVCO = FREF / REFDIV * (FBDIV + FRAC / 2^{24}) = 24/1*(40+24'hf5c28f /2^{24}) = 983.04
FOUTPOSTDIV = FOUTVCO / POSTDIV1 / POSTDIV2=983.04/2/1=491.52

B. PLL frequency range requirement

All the value range requirement can be check in Fig. 3-6.

FREF(Input Frequency Range in Integer Mode):	1MHz to 800MHz
FREF(Input Frequency Range in Fractional Mode):	10MHz to 800MHz
FREF/REFDIV(The divided reference frequency):	1 to 40MHz
FOUTVCO:	400NHz to 1.6GHz
FOUTPOSTDIV:	8MHz to 1.6GHz
FBDIV(Integer Mode):	16~1600
FBDIV(Fractional Mode):	19~160

C. PLL setting consideration

- The divided reference frequency (FREF/REFDIV) should be less than 40MHz
- If the POSTDIV value is changed during operation a short pulse (glitch) may occur on FOUTPOSTDIV. The minimum width of the short pulse will be equal to twice the period of the VCO. Therefore, if the circuitry clocked by the PLL is sensitive to short pulses, the new divide value should be re-timed so that it is synchronous with the rising edge of the output clock (FOUTPOSTDIV). Glitches cannot occur on any of the other outputs.
- For lowest power operation, the minimum VCO and FREF frequencies should be used. For minimum jitter operation, the highest VCO and FREF frequencies should be used. The normal operating range for the VCO is described above in.
- The supply rejection will be worse at the low end of the VCO range so care should be taken to keep the supply clean for low power applications.
- The feedback divider is not capable of dividing by all possible settings due to the use of a power-saving architecture. The following settings are valid for FBDIV:

- DSMPD=1 (Integer Mode):
  12, 13, 14, 16-4095 (practical value is limited to 3200, 2400, or 1600 (FVCOMAX / FREFMIN))
- DSMPD=0 (Fractional Mode):
  19-4091 (practical value is limited to 320, 240, or 160 (FVCOMAX / FREFMIN))

- The PD input places the PLL into the lowest power mode. In this case, all analog circuits are turned off and FREF will be "ignored". The FOUTPOSTDIV and FOUTVCO pins are forced to logic low (0V).
- The BYPASS pin controls a mux which selects FREF to be passed to the FOUTPOSTDIV when active high. However, the PLL continues to run as it normally would if bypass were low. This is a useful feature for PLL testing since the clock path can be verified without the PLL being required to work. Also, the effect that the PLL induced supply noise has on the output buffering can be evaluated. It is not recommended to switch between BYPASS mode and normal mode for regular chip operation since this may result in a glitch. Also, FOUTPOSTDIVPD should be set low if the PLL is to be used in BYPASS mode.

3.10.1 PLL frequency change and lock check

The PLL programming support changed on-the-fly and the PLL will simply slew to the new frequency.
PLL lock state can be check in CRU_APLL_CON1[10] register. The lock state is high when both original hardware PLL lock and PLL counter lock are high. The PLL counter lock initial value is CRU_GLB_CNT_TH[31:16].
The max lock time is 1500 REF_CLK.

PLL locking consists of three phases.
● Phase 1 is control voltage slewing. During this phase one of the clocks (reference or divide) is much faster than the other, and the PLL frequency adjusts almost continuously. When locking from power down, the divide clock is initially very slow and steadily increases frequency.
● Phase 2 is small signal phase acquisition. During this phase, the internal up/down signals alternate semi-chaotically as the phase slowly adjusts until the two signals are aligned. The duration of this phase depends on the loop bandwidth and is faster with higher bandwidth. Bandwidth can be estimated as FREF / REFDIV / 20 for integer mode and FREF /REFDIV / 40 for fractional mode. The duration of small signal locking is about 1/Bandwidth.
● Phase 3 is the digital cycle count. After the last cycle slip is detected, an internal counter waits 256 FREF / REFDIV cycles before the lock signal goes high. This is frequently the dominant factor in lock time – especially for slower reference clock signals or large reference divide settings. This time can be calculated as 256*REFDIV/FREF.

3.10.2 Fractional divider usage

To get specific frequency, clocks of I2S can be generated by fractional divider.
Generally, you must set that denominator is 20 times larger than numerator to generate precise clock frequency. So the fractional divider applies only to generate low frequency clock like I2S.

3.10.3 Global software reset

CRU_GLB_SRST_FST_VALUE[15:0] can be programmed as 0xfdb9 to assert the global software reset glb_srstn. The software reset is self-deasserted by hardware.
Glb_srstn resets all logic except GRF and GPIOs.

Chapter 4 General register file(GRF)

4.1 Overview

The general register file will be used to do static set by software, which is composed of many registers for system control.

4.1.1 Features

The function of general register file is：

- IOMUX control
- Control the state of GPIO in power-down mode
- GPIO PAD pull down and pull up control
- Used for common system control
- Used to record the system state

4.2 GRF Register Description

4.2.1 Register Summary

Name	Offset	Size	Reset Value	Description
GRF_GPIO0A_IOMUX	0x0000	W	0x00000000	GPIO0A iomux control
GRF_GPIO0B_IOMUX	0x0004	W	0x00000000	GPIO0B iomux control
GRF_GPIO0C_IOMUX	0x0008	W	0x00000000	GPIO0C iomux control
GRF_GPIO0D_IOMUX	0x000c	W	0x00000000	GPIO0D iomux control
GRF_GPIO0A_PULL	0x0010	W	0x00000000	GPIO0_A[7:0] bits PULL Control Register
GRF_GPIO0B_PULL	0x0014	W	0x00000000	GPIO0_B[7:0] bits PULL Control Register
GRF_GPIO0C_PULL	0x0018	W	0x00000000	GPIO0_C[7:0] bits PULL Control Register
GRF_GPIO0D_PULL	0x001c	W	0x00000000	GPIO0_D[7:0] bits PULL Control Register
GRF_GPIO1A_IOMUX	0x0020	W	0x00000000	GPIO1A iomux control
GRF_GPIO1B_IOMUX	0x0024	W	0x00000000	GPIO1B iomux control
GRF_GPIO1C_IOMUX	0x0028	W	0x00000000	GPIO1C iomux control
GRF_GPIO1D_IOMUX	0x002c	W	0x00000000	GPIO1D iomux control
GRF_GPIO1A_PULL	0x0030	W	0x00000000	GPIO1_A[7:0] bits PULL Control Register
GRF_GPIO1B_PULL	0x0034	W	0x00000000	GPIO1_B[7:0] bits PULL Control Register
GRF_GPIO1C_PULL	0x0038	W	0x00000000	GPIO1_C[7:0] bits PULL Control Register
GRF_GPIO1D_PULL	0x003c	W	0x00000000	GPIO1_D[7:0] bits PULL Control Register
GRF_GPIO2A_IOMUX	0x0040	W	0x00000000	GPIO2A iomux control
GRF_GPIO2B_IOMUX	0x0044	W	0x00000000	GPIO2B iomux control
GRF_GPIO2C_IOMUX	0x0048	W	0x00000000	GPIO2C iomux control
GRF_GPIO2D_IOMUX	0x004c	W	0x00000000	GPIO2D iomux control
GRF_GPIO2A_PULL	0x0050	W	0x00000000	GPIO2_A[7:0] bits PULL Control Register

Name	Offset	Size	Reset Value	Description
GRF_GPIO2B_PULL	0x0054	W	0x00000000	GPIO2_B[7:0] bits PULL Control Register
GRF_GPIO2C_PULL	0x0058	W	0x00000000	GPIO2_C[7:0] bits PULL Control Register
GRF_GPIO2D_PULL	0x005c	W	0x00000000	GPIO2_D[7:0] bits PULL Control Register
GRF_PVTM_CON0	0x0060	W	0x00000000	
GRF_PVTM_CON1	0x0064	W	0x00000000	
GRF_PVTM_CON2	0x0068	W	0x00000000	
GRF_PVTM_STATUS0	0x006c	W	0x00000000	
GRF_PVTM_STATUS1	0x0070	W	0x00000000	
GRF_USBPHY_CON0	0x0080	W	0x00008518	usbphy control register
GRF_USBPHY_CON1	0x0084	W	0x0000e001	usbphy control register
GRF_USBPHY_CON2	0x0088	W	0x000082a7	usbphy control register
GRF_USBPHY_CON3	0x008c	W	0x00000200	usbphy control register
GRF_USBPHY_CON4	0x0090	W	0x00000002	usbphy control register
GRF_USBPHY_CON5	0x0094	W	0x00000000	usbphy control register
GRF_USBPHY_CON6	0x0098	W	0x00000004	usbphy control register
GRF_USBPHY_CON7	0x009c	W	0x000068c0	usbphy control register
GRF_USBPHY_CON8	0x00a0	W	0x00000020	usbphy control register
GRF_USBPHY_CON9	0x00a4	W	0x00000000	usbphy control register
GRF_USBPHY_CON10	0x00a8	W	0x00000000	usbphy control register
GRF_USBPHY_CON11	0x00ac	W	0x00000000	usbphy control register
GRF_UOC_CON0	0x00b0	W	0x00000000	
GRF_UOC_CON1	0x00b4	W	0x00000000	
GRF_UOC_CON2	0x00b8	W	0x00000000	
GRF_IOMUX_CON	0x00bc	W	0x00000000	
GRF_INTER_CON0	0x00c4	W	0x00000001	
GRF_GRF_VREF_CON	0x00cc	W	0x00000018	
GRF_SOC_STATUS0	0x00e0	W	0x00000000	
GRF_SOC_STATUS1	0x00e4	W	0x00000000	
GRF_SOC_USB_STATUS	0x00e8	W	0x00000000	
GRF_PRJ_ID	0x00f8	W	0xdddddddd	Project ID Register
GRF_CPU_ID	0x00fc	W	0x00000000	The id of cpu

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

4.2.2 Detailed Register Description

Operational Base Address of GRF is 0x50010000.

GRF_GPIO0A_IOMUX

Address: Operational Base + offset (0x0000)

GPIO0A iomux control

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio0a7_sel GPIO0A[7] iomux select 2'b11: reserve 2'b10: sfc_d1 2'b01: emmc_d4 2'b00: gpio
13:12	RW	0x0	gpio0a6_sel GPIO0A[6] iomux select 2'b11: i2c0c_scl 2'b10: sfc_d2 2'b01: emmc_d3 2'b00: gpio
11:10	RW	0x0	gpio0a5_sel GPIO0A[5] iomux select 2'b11: i2c0c_sda 2'b10: sfc_d3 2'b01: emmc_d2 2'b00: gpio
9:8	RW	0x0	gpio0a4_sel GPIO0A[4] iomux select 2'b11: uart2c_rts 2'b10: i2s1b_sdi 2'b01: emmc_d1 2'b00: gpio
7:6	RW	0x0	gpio0a3_sel GPIO0A[3] iomux select 2'b11: uart2c_cts 2'b10: i2s1b_sdo 2'b01: emmc_d0 2'b00: gpio

Bit	Attr	Reset Value	Description
5:4	RW	0x0	gpio0a2_sel GPIO0A[2] iomux select 2'b11: uart2c_rx 2'b10: i2s1b_sclk 2'b01: emmc_cmd 2'b00: gpio
3:2	RW	0x0	gpio0a1_sel GPIO0A[1] iomux select 2'b11: uart2c_tx 2'b10: i2s1b_lrck 2'b01: emmc_clk 2'b00: gpio
1:0	RW	0x0	gpio0a0_sel GPIO0A[0] iomux select 2'b11: reserve 2'b10: i2s1b_clk 2'b01: emmc_pwren 2'b00: gpio

GRF_GPIO0B_IOMUX
Address: Operational Base + offset (0x0004)
GPIO0B iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio0b7_sel GPIO0B[7] iomux select 2'b11: ebc_gdoe 2'b10: i2s0_clk 2'b01: lcd_csn 2'b00: gpio

Bit	Attr	Reset Value	Description
13:12	RW	0x0	gpio0b6_sel GPIO0B[6] iomux select 2'b11: ebc_gdsp 2'b10: i2s0_lrck 2'b01: uart2a_tx 2'b00: gpio
11:10	RW	0x0	gpio0b5_sel GPIO0B[5] iomux select 2'b11: ebc_gdclk 2'b10: i2s0_sclk 2'b01: uart2a_rx 2'b00: gpio
9:8	RW	0x0	gpio0b4_sel GPIO0B[4] iomux select 2'b11: ebc_sdclk 2'b10: i2s0_sdo 2'b01: uart2a_cts 2'b00: gpio
7:6	RW	0x0	gpio0b3_sel GPIO0B[3] iomux select 2'b11: ebc_vcom 2'b10: i2s0_sdi 2'b01: uart2a_rts 2'b00: gpio
5:4	RW	0x0	gpio0b2_sel GPIO0B[2] iomux select 2'b11: jtg1_trst 2'b10: sfc_cs 2'b01: emmc_d7 2'b00: gpio
3:2	RW	0x0	gpio0b1_sel GPIO0B[1] iomux select 2'b11: jtg1_tdo 2'b10: sfc_clk 2'b01: emmc_d6 2'b00: gpio
1:0	RW	0x0	gpio0b0_sel GPIO0B[0] iomux select 2'b11: jtg1_tdi 2'b10: sfc_d0 2'b01: emmc_d5 2'b00: gpio

GRF_GPIO0C_IOMUX
Address: Operational Base + offset (0x0008)

GPIO0C iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio0c7_sel GPIO0C[7] iomux select 2'b11: ebc_sddo7 2'b10: uart2b_cts 2'b01: lcd_d7 2'b00: gpio
13:12	RW	0x0	gpio0c6_sel GPIO0C[5] iomux select 2'b11: ebc_sddo6 2'b10: uart2b_rts 2'b01: lcd_d6 2'b00: gpio
11:10	RW	0x0	gpio0c5_sel GPIO0C[5] iomux select 2'b11: ebc_sddo5 2'b10: uart2b_tx 2'b01: lcd_d5 2'b00: gpio
9:8	RW	0x0	gpio0c4_sel GPIO0C[4] iomux select 2'b11: ebc_sddo4 2'b10: uart2b_rx 2'b01: lcd_d4 2'b00: gpio
7:6	RW	0x0	gpio0c3_sel GPIO0C[3] iomux select 2'b11: ebc_sddo3 2'b10: spi0a_cs 2'b01: lcd_d3 2'b00: gpio

Bit	Attr	Reset Value	Description
5:4	RW	0x0	gpio0c2_sel GPIO0C[2] iomux select 2'b11: ebc_sddo2 2'b10: spi0a_clk 2'b01: lcd_d2 2'b00: gpio
3:2	RW	0x0	gpio0c1_sel GPIO0C[1] iomux select 2'b11: ebc_sddo1 2'b10: spi0a_rx 2'b01: lcd_d1 2'b00: gpio
1:0	RW	0x0	gpio0c0_sel GPIO0C[0] iomux select 2'b11: ebc_sddo0 2'b10: spi0a_tx 2'b01: lcd_d0 2'b00: gpio

GRF_GPIO0D_IOMUX
Address: Operational Base + offset (0x000c)
GPIO0D iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio0d7_sel GPIO0D[6] iomux select
13:12	RW	0x0	gpio0d6_sel GPIO0D[6] iomux select
11:10	RW	0x0	gpio0d5_sel GPIO0D[5] iomux select

Bit	Attr	Reset Value	Description
9:8	RW	0x0	gpio0d4_sel GPIO0D[4] iomux select
7:6	RW	0x0	gpio0d3_sel GPIO0D[3] iomux select
5:4	RW	0x0	gpio0d2_sel GPIO0D[2] iomux select
3:2	RW	0x0	gpio0d1_sel GPIO0D[1] iomux select 2'b11: ebc_sdoe 2'b10: i2c2b_sda 2'b01: lcd_rs 2'b00: gpio
1:0	RW	0x0	gpio0d0_sel GPIO0D[0] iomux select 2'b11: ebc_sdle 2'b10: i2c2b_scl 2'b01: lcd_wrn 2'b00: gpio

GRF_GPIO0A_PULL
Address: Operational Base + offset (0x0010)
GPIO0_A[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO0B_PULL
Address: Operational Base + offset (0x0014)
GPIO0_B[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved

Bit	Attr	Reset Value	Description
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO0C_PULL
Address: Operational Base + offset (0x0018)
GPIO0_C[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO0D_PULL
Address: Operational Base + offset (0x001c)
GPIO0_D[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable

Bit	Attr	Reset Value	Description
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO1A_IOMUX
Address: Operational Base + offset (0x0020)
GPIO1A iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio1a7_sel GPIO1A[7] iomux select 2'b11: uart4_tx 2'b10: spi1a_rx 2'b01: sdmmc_d0 2'b00: gpio
13:12	RW	0x0	gpio1a6_sel GPIO1A[6] iomux select 2'b11: uart3_rx 2'b10: spi1a_clk 2'b01: sdmmc_clk 2'b00: gpio

Bit	Attr	Reset Value	Description
11:10	RW	0x0	gpio1a5_sel GPIO1A[5] iomux select 2'b11: uart3_tx 2'b10: spi1a_cs 2'b01: sdmmc_cmd 2'b00: gpio
9:8	RW	0x0	gpio1a4_sel GPIO1A[4] iomux select 2'b11: ebc_sdce4 2'b10: uart1b_rts 2'b01: i2s1a_sdi 2'b00: gpio
7:6	RW	0x0	gpio1a3_sel GPIO1A[3] iomux select 2'b11: ebc_sdce1 2'b10: uart1b_cts 2'b01: i2s1a_sdo 2'b00: gpio
5:4	RW	0x0	gpio1a2_sel GPIO1A[2] iomux select 2'b11: ebc_sdce2 2'b10: uart1b_tx 2'b01: i2s1a_sclk 2'b00: gpio
3:2	RW	0x0	gpio1a1_sel GPIO1A[1] iomux select 2'b11: ebc_sdshr 2'b10: uart1b_rx 2'b01: i2s1a_lrck 2'b00: gpio
1:0	RW	0x0	gpio1a0_sel GPIO1A[0] iomux select 2'b11: reserve 2'b10: ebc_gdrl 2'b01: i2s1a_clk 2'b00: gpio

GRF_GPIO1B_IOMUX
Address: Operational Base + offset (0x0024)
GPIO1B iomux control

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio1b7_sel GPIO1B[7] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
13:12	RW	0x0	gpio1b6_sel GPIO1B[6] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
11:10	RW	0x0	gpio1b5_sel GPIO1B[5] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
9:8	RW	0x0	gpio1b4_sel GPIO1B[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
7:6	RW	0x0	gpio1b3_sel GPIO1B[3] iomux select 2'b11: reserve 2'b10: reserve 2'b01: emmc_rstn 2'b00: gpio

Bit	Attr	Reset Value	Description
5:4	RW	0x0	gpio1b2_sel GPIO1B[2] iomux select 2'b11: uart5_rx 2'b10: i2c1b_sda 2'b01: sdmmc_d3 2'b00: gpio
3:2	RW	0x0	gpio1b1_sel GPIO1B[1] iomux select 2'b11: uart5_tx 2'b10: i2c1b_scl 2'b01: sdmmc_d2 2'b00: gpio
1:0	RW	0x0	gpio1b0_sel GPIO1B[0] iomux select 2'b11: uart4_rx 2'b10: spi1a_tx 2'b01: sdmmc_d1 2'b00: gpio

GRF_GPIO1C_IOMUX
Address: Operational Base + offset (0x0028)
GPIO1C iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio1c7_sel GPIO1C[7] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio

Bit	Attr	Reset Value	Description
13:12	RW	0x0	gpio1c6_sel GPIO1C[6] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
11:10	RW	0x0	gpio1c5_sel GPIO1C[5] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
9:8	RW	0x0	gpio1c4_sel GPIO1C[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
7:6	RW	0x0	gpio1c3_sel GPIO1C[3] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
5:4	RW	0x0	gpio1c2_sel GPIO1C[2] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
3:2	RW	0x0	gpio1c1_sel GPIO1C[1] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
1:0	RW	0x0	gpio1c0_sel GPIO1C[0] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio

GRF_GPIO1D_IOMUX
Address: Operational Base + offset (0x002c)
GPIO1D iomux control

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio1d7_sel GPIO1D[7] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
13:12	RW	0x0	gpio1d6_sel GPIO1D[6] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
11:10	RW	0x0	gpio1d5_sel GPIO1D[5] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
9:8	RW	0x0	gpio1d4_sel GPIO1D[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
7:6	RW	0x0	gpio1d3_sel GPIO1D[3] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
5:4	RW	0x0	gpio1d2_sel GPIO1D[2] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
3:2	RW	0x0	gpio1d1_sel GPIO1D[1] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio

Bit	Attr	Reset Value	Description
1:0	RW	0x0	gpio1d0_sel GPIO1D[0] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio

GRF_GPIO1A_PULL
Address: Operational Base + offset (0x0030)
GPIO1_A[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO1B_PULL
Address: Operational Base + offset (0x0034)
GPIO1_B[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO1C_PULL
Address: Operational Base + offset (0x0038)
GPIO1_C[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO1D_PULL
Address: Operational Base + offset (0x003c)
GPIO1_D[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO2A_IOMUX
Address: Operational Base + offset (0x0040)
GPIO2A iomux control

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio2a7_sel GPIO2A[7] iomux select 2'b11: reserve 2'b10: reserve 2'b01: pmu_idel 2'b00: gpio
13:12	RW	0x0	gpio2a6_sel GPIO2A[6] iomux select 2'b11: pmu_st0 2'b10: jtg0_trst 2'b01: i2c2c_sda 2'b00: gpio
11:10	RW	0x0	gpio2a5_sel GPIO2A[5] iomux select 2'b11: pmu_st1 2'b10: jtg0_tdo 2'b01: i2c2c_scl 2'b00: gpio
9:8	RW	0x0	gpio2a4_sel GPIO2A[4] iomux select 2'b11: pmu_st2 2'b10: jtg0_tdi 2'b01: pwm0_out 2'b00: gpio
7:6	RW	0x0	gpio2a3_sel GPIO2A[3] iomux select 2'b11: ebc_gdpwr1 2'b10: clk_obs 2'b01: pwm1_out 2'b00: gpio

Bit	Attr	Reset Value	Description
5:4	RW	0x0	gpio2a2_sel GPIO2A[2] iomux select 2'b11: pmu_st3 2'b10: ebc_gdpwr2 2'b01: pwm2_out 2'b00: gpio
3:2	RW	0x0	gpio2a1_sel GPIO2A[1] iomux select 2'b11: ebc_sdce0 2'b10: i2c2a_scl 2'b01: pwm3_out 2'b00: gpio
1:0	RW	0x0	gpio2a0_sel GPIO2A[0] iomux select 2'b11: ebc_gdpwr0 2'b10: i2c2a_sda 2'b01: pwm4_out 2'b00: gpio

GRF_GPIO2B_IOMUX
Address: Operational Base + offset (0x0044)
GPIO2B iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio2b7_sel GPIO2B[7] iomux select 2'b11: spi1b_rx 2'b10: jtg0_tck 2'b01: uart1a_ctx 2'b00: gpio

Bit	Attr	Reset Value	Description
13:12	RW	0x0	gpio2b6_sel GPIO2B[6] iomux select 2'b11: spi1b_cs 2'b10: jtg0_tms 2'b01: uart1a_rts 2'b00: gpio
11:10	RW	0x0	gpio2b5_sel GPIO2B[5] iomux select 2'b11: i2c1c_scl 2'b10: jtg1_tck 2'b01: uart0a_tx 2'b00: gpio
9:8	RW	0x0	gpio2b4_sel GPIO2B[4] iomux select 2'b11: i2c1c_sda 2'b10: jtg1_tms 2'b01: uart0a_rx 2'b00: gpio
7:6	RW	0x0	gpio2b3_sel GPIO2B[3] iomux select 2'b11: ebc_border0 2'b10: spi0b_rx 2'b01: i2c0a_sda 2'b00: gpio
5:4	RW	0x0	gpio2b2_sel GPIO2B[2] iomux select 2'b11: ebc_sdce5 2'b10: spi0b_tx 2'b01: i2c0a_scl 2'b00: gpio
3:2	RW	0x0	gpio2b1_sel GPIO2B[1] iomux select 2'b11: ebc_border1 2'b10: spi0b_clk 2'b01: i2c1a_scl 2'b00: gpio
1:0	RW	0x0	gpio2b0_sel GPIO2B[0] iomux select 2'b11: ebc_sdce3 2'b10: spi0b_cs 2'b01: i2c1a_sda 2'b00: gpio

GRF_GPIO2C_IOMUX
Address: Operational Base + offset (0x0048)

GPIO2C iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio2c7_sel GPIO2C[7] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
13:12	RW	0x0	gpio2c6_sel GPIO2C[6] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
11:10	RW	0x0	gpio2c5_sel GPIO2C[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
9:8	RW	0x0	gpio2c4_sel GPIO2C[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
7:6	RW	0x0	gpio2c3_sel GPIO2C[3] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
5:4	RW	0x0	gpio2c2_sel GPIO2C[2] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
3:2	RW	0x0	gpio2c1_sel GPIO2C[1] iomux select 2'b11: spi1b_tx 2'b10: i2c0b_scl 2'b01: uart1a_rx 2'b00: gpio

Bit	Attr	Reset Value	Description
1:0	RW	0x0	gpio2c0_sel GPIO2c[0] iomux select 2'b11: spi1b_clk 2'b10: i2c0b_sda 2'b01: uart1a_tx 2'b00: gpio

GRF_GPIO2D_IOMUX
Address: Operational Base + offset (0x004c)
GPIO2D iomux control

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:14	RW	0x0	gpio2d7_sel GPIO2D[7] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
13:12	RW	0x0	gpio2d6_sel GPIO2D[6] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
11:10	RW	0x0	gpio2d5_sel GPIO2D[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
9:8	RW	0x0	gpio2d4_sel GPIO2D[4] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio

Bit	Attr	Reset Value	Description
7:6	RW	0x0	gpio2d3_sel GPIO2D[3] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
5:4	RW	0x0	gpio2d2_sel GPIO2D[2] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
3:2	RW	0x0	gpio2d1_sel GPIO2D[1] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio
1:0	RW	0x0	gpio2d0_sel GPIO2D[0] iomux select 2'b11: reserve 2'b10: reserve 2'b01: reserve 2'b00: gpio

GRF_GPIO2A_PULL
Address: Operational Base + offset (0x0050)
GPIO2_A[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO2B_PULL
Address: Operational Base + offset (0x0054)
GPIO2_B[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved

Bit	Attr	Reset Value	Description
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO2C_PULL
Address: Operational Base + offset (0x0058)
GPIO2_C[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_GPIO2D_PULL
Address: Operational Base + offset (0x005c)
GPIO2_D[7:0] bits PULL Control Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RW	0x00	write_enable write enable flag for corresponding low 16bits, one for each bit. 1 : write enable 0 : write disable

Bit	Attr	Reset Value	Description
15:8	RO	0x0	reserved
7:0	RW	0x00	pull Values written to this register independently control Pull-up/ Pull-down or not for the corresponding data bit in GPIOi[7:0] bits. 0: Pull-up/Pull-down enable , PAD type will decide to be up or down , not related with this value 1: not pull-up/pull-down

GRF_PVTM_CON0
Address: Operational Base + offset (0x0060)

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:11	RO	0x0	reserved
10:4	RW	0x00	pvtm_div_con pvtm clock divider frequency clk=clk_src/(div_con+1)
3:2	RO	0x0	reserved
1	RW	0x0	pvtm_func_osc_en pd_core PVT monitor oscillator enable 1'b1: enable 1'b0: disable
0	RW	0x0	pvtm_func_start pd_core PVT monitor start control posed edge active(write 0, delay 1us, then write 1, delay 1us)

GRF_PVTM_CON1
Address: Operational Base + offset (0x0064)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pvtm_func_cal_cnt funcpvtm calculator counter

GRF_PVTM_CON2
Address: Operational Base + offset (0x0068)

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:9	RO	0x0	reserved
8:0	RW	0x000	clk_24m_div_con low 24m clock divider frequency 24MHz clock divider, work in PD PMU. generate the clk_low, which is one of the source clk of U_PMU pclkclk=clk_src/(div_con+1)

GRF_PVTM_STATUS0
Address: Operational Base + offset (0x006c)

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	pvtm_func_freq_done funcpvtm frequency calculate done status

GRF_PVTM_STATUS1
Address: Operational Base + offset (0x0070)

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	pvtm_func_freq_cnt pvtm frequency count

GRF_USBPHY_CON0
Address: Operational Base + offset (0x0080)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:13	RW	0x4	squel_trigger_con bit 2 ~ bit 0 of squel_trigger_con. 0000:112.5mV 1001:162.5mV 1011:175mV 1100:150mV(default) 1110:125mV
12:11	RW	0x0	non_driving Registers for non-driving state control. non-driving state is controlled by op-mode by default, when bit[11] is configured with "1", user can control non-driving state trough bit[12].
10:8	RW	0x5	tx_clk_phase_con USB Tx Clock phase configure, 000 represent the earliest phase , and 111 the latest, single step delay is 256ps
7:5	RW	0x0	rx_clk_phase_con USB Rx Clock phase configure,000 represent the earliest phase , and 111 the latest, single step delay is 256ps
4:3	RW	0x3	fls_eye_height FS/LS eye height configure , 00 represent the largest slew rate , 11 represent the smallest slew rate

Bit	Attr	Reset Value	Description
2:0	RW	0x0	hs_eye_diag_adjust HS eye diagram adjust, open HS pre-emphasize function to increase HS slew rate, only used when large cap loading is Attached. 001:open pre-emphasize in sof or eop state 010: open pre-emphasize in chirp state 100: open pre-emphasize in non-chirp state 111: always open pre-emphasize other combinations : reserved

GRF_USBPHY_CON1
Address: Operational Base + offset (0x0084)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:13	RW	0x7	hs_eye_height bit2 ~ bit 0 of hs_eye_height. HS eye height tuning ,more zeros represent bigger eye, more ones represent smaller eye
12:1	RO	0x0	reserved
0	RW	0x1	squel_trigger_con bit 3 of squel_trigger_con. 0000:112.5mV 1001:162.5mV 1011:175mV 1100:150mV(default) 1110:125mV

GRF_USBPHY_CON2
Address: Operational Base + offset (0x0088)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15	RW	0x1	odt_compensation bit 0 of odt_compensation. ODT Compensation voltage reference 000:200mV 001:187.5mV(default) 010:225mV 110:175mV 111:162.5mV
14:13	RW	0x0	battery_charging_related Battery charging related registers, keeping the default value is greatly appreciated.
12	RW	0x0	bg_vref_adj BG output voltage reference adjust, keeping the default value is greatly appreciated.
11:10	RW	0x0	auto_compensation_bypass auto compensation bypass , "11" will bypass current and ODT compensation, customers can set the driver strength and current manually. For larger HS eye height, customer can give more "0" for hs_eye_height; For larger HS/FS/LS slew rate, give more "1" for hfs_driver_strength.
9:5	RW	0x15	hfs_driver_strength HS/FS driver strength tuning , "11111" represent the largest slew rate and "10000" represents the smallest slew rate

Bit	Attr	Reset Value	Description
4:0	RW	0x07	hs_eye_height bit7 ~ bit 3 of hs_eye_height. HS eye height tuning ,more zeros represent bigger eye, more ones represent smaller eye

GRF_USBPHY_CON3
Address: Operational Base + offset (0x008c)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15	RO	0x0	reserved
14	RW	0x0	vol_toleran_det_con 5V tolerance detection function controlling bit trough registers, only active when bit[65] is set "1".
13:10	RO	0x0	reserved
9	RW	0x1	odt_auto_refresh A port ODT auto refresh bypass, active low, this register should only be used when auto_compensation_bypass were set to "11". In bypass mode , customer can configure driver strength through hfs_driver_strength.
8	RW	0x0	bg_out_voltage_adjust BG output voltage reference adjust, keeping the default value is greatly appreciated.

Bit	Attr	Reset Value	Description
7:5	RW	0x0	compen_current_ref compensation current tuning reference 000:200mV(default) 001:187.5mV 010:225mV 110:175mV 111:162.5mV
4:2	RW	0x0	bias_current_ref bias current tuning reference 000:400mV(default) 001:362.5mV 010:350mV 101:425mV 111:450mV
1:0	RW	0x0	odt_compensation bit 2 ~ bit 1 of odt_compensation. ODT Compensation voltage reference 000:200mV 001:187.5mV(default) 010:225mV 110:175mV 111:162.5mV

GRF_USBPHY_CON4
Address: Operational Base + offset (0x0090)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:2	RO	0x0	reserved

Bit	Attr	Reset Value	Description
1	RW	0x1	bypass_5v_tolerance_det Bypass 5V tolerance detection function, active high
0	RO	0x0	reserved

GRF_USBPHY_CON5
Address: Operational Base + offset (0x0094)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:0	RO	0x0	reserved

GRF_USBPHY_CON6
Address: Operational Base + offset (0x0098)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;

Bit	Attr	Reset Value	Description
15:13	RW	0x0	session_end_con session_end reference tuning
12:10	RW	0x0	b_session_con B_sessionvalid reference tuning
9:7	RW	0x0	a_session_con A_sessionvalid reference tuning
6	RW	0x0	force_vbus_valid force output vbus_valid asserted, active high
5	RW	0x0	force_session_end_val force output session_end asserted, active high
4	RW	0x0	force_b_session_val force output B_sessionvalid asserted, active high
3	RW	0x0	force_a_session_val force output A_sessionvalid asserted, active high
2	RW	0x1	turn_off_diff_receiver Turn off differential receiver in suspend mode to save power, active low.
1:0	RO	0x0	reserved

GRF_USBPHY_CON7
Address: Operational Base + offset (0x009c)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	vbus_vol_det vbus voltage level detection function power down, active high.

Bit	Attr	Reset Value	Description
14:11	RW	0xd	host_discon_con HOST disconnect detection trigger point configure, only used in HOST mode 0000: 575mV 0001: 600mV 1001:625mV 1101:650mV(default)
10:8	RO	0x0	reserved
7	RW	0x1	bypass_squelch_trigger bypass squelch trigger point auto configure in chirp modes , active high
6	RW	0x1	half_bit_pre_empha_en half bit pre-emphasize enable, active high. "1" represent half bit pre-emphasis , "0" for full bit
5:3	RO	0x0	reserved
2:0	RW	0x0	vbus_valid_con vbus_valid reference tuning

GRF_USBPHY_CON8
Address: Operational Base + offset (0x00a0)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:11	RO	0x0	reserved
10:8	RW	0x0	Tx_HS_driver_str Tx HS driver strength configure, more "1" represents larger slew rate and eye height

Bit	Attr	Reset Value	Description
7:5	RW	0x1	Tx_HS_pre_emphasize_str TX HS pre_emphasize strength configure, 111 represents the strongest, 000 the weakest.
4:0	RW	0x00	battery_charging_related Battery charging related registers, keeping the default value is greatly appreciated.

GRF_USBPHY_CON9
Address: Operational Base + offset (0x00a4)
usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0x0000	reserve

GRF_USBPHY_CON10
Address: Operational Base + offset (0x00a8)
usb phy control register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0x0000	reserve

GRF_USBPHY_CON11
Address: Operational Base + offset (0x00ac)
Usb phy control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:0	RW	0x0000	reserve

GRF_UOC_CON0
Address: Operational Base + offset (0x00b0)

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:3	RO	0x0	reserved
2	RW	0x0	bypasssel0 1'b1: bypass function OTG, INNO_USB_PHY connect to UART0 0'b0: do not bypass NOTE: otgdisable0, bypassdmem0 and bypasssel0 should all be high, then function OTG bypassed, and INNO_USB_PHY connect to UART0
1	RW	0x0	bypassdmen0 1'b1: INNO_USB_PHY connect to UART0 0'b0: do not bypass NOTE: otgdisable0, bypassdmem0 and bypasssel0 should all be high, then OTG function bypassed, and INNO_USB_PHY connect to UART0
0	RW	0x0	otgdisable0 1'b1: INNO_USB_PHY connect to UART0 0'b0: do not bypass NOTE: otgdisable0, bypassdmem0 and bypasssel0 should all be high, then OTG function bypassed, and INNO_USB_PHY connect to UART0

GRF_UOC_CON1
Address: Operational Base + offset (0x00b4)

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:11	RO	0x0	reserved
10	RW	0x0	UTMIOTG_IDDIG USB Plug Indicator Output
9	RW	0x0	UTMIOTG_IDDIG_SEL 1'b1: select UOC_CON1[10](UTMIOTG_IDDIG) as USB controller's input signal "utmiotg_iddig" 1'b0: select "utmiotg_iddig" from USB OTG PHY as USB controller's input signal "iddig"
8:7	RO	0x0	reserved
6	RW	0x0	UTMI_TERMSELECT Termination Select between FS/LS and HS Terminations
5:4	RW	0x0	UTMI_XCVRSELECT Transceiver Select between FS/LS and HS Transceivers
3:2	RW	0x0	UTMI_OPMODE operational mode selector between various modes
1	RW	0x0	UTMI_SUSPEND_N suspend mode enable. 1'b0: suspend 1'b1: normal

Bit	Attr	Reset Value	Description
0	RW	0x0	USB_CTRL_SEL Data path selector of USB controller. 1'b1: select bit6 to bit1 (UTMI_SUSPEND_N, UTMI_OPMODE, UTMI_XCVRSELECT, UTMI_TERMSELECT) as the corresponding input signals of USB controller. 1'b0: select UTMI_SUSPEND_N, UTMI_OPMODE, UTMI_XCVRSELECT, UTMI_TERMSELECT from USB PHY as the corresponding input signals of USB controller.

GRF_UOC_CON2

Address: Operational Base + offset (0x00b8)

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:7	RO	0x0	reserved
6	RW	0x0	utmi_dischrgbus VBUS discharging enable 1'b1: disable 1'b0: enable
5	RW	0x0	utmi_chrgbus VBUS charging enable 1'b1: enable 1'b0: disable
4	RW	0x0	utmi_drvvbus connect to INNO_USB_PHY
3	RW	0x0	utmi_idpullup signal that enable analog ID line sampling 1'b1: enable 1'b0: disable

Bit	Attr	Reset Value	Description
2	RW	0x0	utmi_dmpulldown Enable DMINUS Pull Down resistor 1'b1: enable 1'b0: disable
1	RW	0x0	utmi_dppulldown Enable DPLUS Pull Down resistor 1'b1: enable 1'b0: disable
0	RW	0x0	usbphy_commonon configure PLL clock output in suspend mode

GRF_IOMUX_CON

Address: Operational Base + offset (0x00bc)

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:13	RO	0x0	reserved
12	RW	0x0	i2s0_sel 1'b0: i2s0 is connected to pad 1'b1: i2s0 is connected to acodec
11:10	RW	0x0	uart2_sel 2'b0: uart2a 2'b1: uart2b 2'b2: uart2c 2'b3: reserve
9	RW	0x0	uart1_sel 1'b0: uart1a 1'b1: uart1b
8	RO	0x0	reserved
7	RW	0x0	spi1_sel 1'b0: spi1a 1'b1: spi1b

Bit	Attr	Reset Value	Description
6	RW	0x0	spi0_sel 1'b0: spi0a 1'b1: spi0b
5:4	RW	0x0	i2c2_sel 2'b0: i2c2a 2'b1: i2c2b 2'b2: i2c2c 2'b3: reserve
3:2	RW	0x0	i2c1_sel 2'b0: i2c1a 2'b1: i2c1b 2'b2: i2c1c 2'b3: reserve
1:0	RW	0x0	i2c0_sel 2'b0: i2c0a 2'b1: i2c0b 2'b2: i2c0c 2'b3: reserve

GRF_INTER_CON0

Address: Operational Base + offset (0x00c4)

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15	RW	0x0	caldram_resp_cycle cal dram multicycle enable cal dram multi-cycle ctrl register 1'b1: enable 1'b0: disable

Bit	Attr	Reset Value	Description
14	RW	0x0	caliram_resp_cycle calirammulticycle enable caliram multi-cycle ctrl register 1'b1: enable 1'b0: disable
13	RW	0x0	sysdram_resp_cycle sys dram multicycle enable system dram multi-cycle ctrl register 1'b1: enable 1'b0: disable
12	RW	0x0	sysiram_resp_cycle sys irammulticycle enable system iram multi-cycle ctrl register 1'b1: enable 1'b0: disable
11:9	RO	0x0	reserved
8	RW	0x0	noc_remap noc_remap: 0: remap BOOT_ROM to 0000_0000 1: remap PMU-SRAM to 0000_0000
7	RO	0x0	reserved
6	RW	0x0	pmu_resp_cycle pmusrammulticycle enable pmusram multi-cycle ctrl register 1'b1: enable 1'b0: disable
5	RW	0x0	btrom_resp_cycle boot rom multi-cycle ctrl register 1'b1: enable 1'b0: disable
4:2	RO	0x0	reserved
1	RW	0x0	dma_uart02_sel 1'b0: uart0 1'b1: uart2
0	RW	0x1	force_jtag if enalbed, jtag0 will be connected to pad 1'b1: enable 1'b0: disable

GRF_GRF_VREF_CON

Address: Operational Base + offset (0x00cc)

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	RW	0x0000	write_enable bit0~bit15 write enable When bit 16=1, bit 0 can be written by software. When bit 16=0, bit 0 cannot be written by software; When bit 17=1, bit 1 can be written by software. When bit 17=0, bit 1 cannot be written by software; ..... When bit 31=1, bit 15 can be written by software. When bit 31=0, bit 15 cannot be written by software;
15:7	RO	0x0	reserved
6:4	RW	0x1	grf_vreg_vbg_sel Select the VBG voltage which should has the best temperature characteristics
3:1	RW	0x4	grf_vref_trim The 2.5V output voltage trim register, the turning range is 2.4V~2.6V. 000-> 2.4V 111-> 2.6V
0	RW	0x1	grf_vref_pd When high the ADC REF is power down

GRF_SOC_STATUS0

Address: Operational Base + offset (0x00e0)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x0	syscore_halted In halting debug mode. HALTED remains asserted while the core is in debug.

Bit	Attr	Reset Value	Description
2	RW	0x0	syscore_lockup Reads as one if the core is running (not halted) and a lockup condition is present. LOCKUP gives immediate indication of seriously errant kernel software. This is the result of the core being locked up because of an unrecoverable exception following the activation of the processor's built in system state protection hardware. For more information about the ARMv7-M architectural lock up conditions see the ARMv7-M Architecture Reference Manual.
1	RW	0x0	syscore_sleepdeep sys core Sleep deep bit: 1 = indicates to the system that Cortex-M3 clock can be stopped. Setting this bit causes the SLEEPDEEP port to be asserted when the processor can be stopped. 0 = not OK to turn off system clock.
0	RO	0x0	syscore_sleeping 1'b1: sys core is sleeping 1'b0: sys core is not sleeping

GRF_SOC_STATUS1

Address: Operational Base + offset (0x00e4)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x0	calcore_halted In halting debug mode. HALTED remains asserted while the core is in debug.
2	RW	0x0	calcore_lockup Reads as one if the core is running (not halted) and a lockup condition is present. LOCKUP gives immediate indication of seriously errant kernel software. This is the result of the core being locked up because of an unrecoverable exception following the activation of the processor's built in system state protection hardware. For more information about the ARMv7-M architectural lock up conditions see the ARMv7-M Architecture Reference Manual.

Bit	Attr	Reset Value	Description
1	RW	0x0	calcore_sleepdeep sys core Sleep deep bit: 1 = indicates to the system that Cortex-M3 clock can be stopped. Setting this bit causes the SLEEPDEEP port to be asserted when the processor can be stopped. 0 = not OK to turn off system clock.
0	RO	0x0	calcore_sleeping 1'b1: sys core is sleeping 1'b0: sys core is not sleeping

GRF_SOC_USB_STATUS
Address: Operational Base + offset (0x00e8)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x0	utmisrp_sessend
2	RW	0x0	utmiotg_avalid
1	RW	0x0	utmisrp_bvalid
0	RO	0x0	utmiotg_vbusvalid

GRF_PRJ_ID
Address: Operational Base + offset (0x00f8)
Project ID Register

Bit	Attr	Reset Value	Description
31:0	RO	0xdddddddd	id

GRF_CPU_ID
Address: Operational Base + offset (0x00fc)
The id of cpu

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	cpu_id The id of cpu 0: core 0 1: core 1

Chapter 5 Power Management Unit (PMU)

5.1 Overview

In order to meet high performance and low power requirements, a power management unit is designed for saving power when the chip in low power mode. The PMU is dedicated for managing the power of the whole chip.

5.1.1 Features

- Support 2 voltage domains including VD_LOGIC, VD_PMU
- Support 3 separate power domains including pd_pmu, pd_logic, pd_high, which can be power up/down by software based on different application scenes.
- In low power mode, the PMU could power up/down VD_LOGIC by hardware
- Support global interrupt disable in low power mode
- Support pd_pmu clock switch to 24M_div_outor PVTM clock in low power mode
- A group of configurable counter in PMU for HW control.
- Support GPIO or GPIO interrupt as wakeup source for low power mode

5.2 Block Diagram

5.2.1 power domain partition

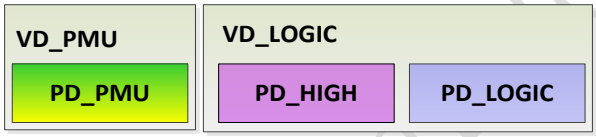


Fig 5-1 Power Domain Partition

The above diagram describes the power domain and voltage domain partition, and the following table lists all the power domains.

Table 5-1 Power Domain and Voltage Domain Summary

Voltage Domain	Power Domain	Description
VD_PMU	PD_PMU	Including PMU, GRF and GPIO2.
VD_LOGIC	PD_HIGH	Including core 1, HIFIACC and HIRAM, HDRAM.
	PD_LOGIC	Including system CPU, internal SRAM, ROM and all the peripheral IP.

5.2.2 PMU block diagram

The following figure is the PMU block diagram. The PMU includes the 3 following parts:

- APB interface and register, which can accept the system configuration.
- Low Power State Control, which generates low power controlling signals.
- Power Switch Control, which control all power domain switch

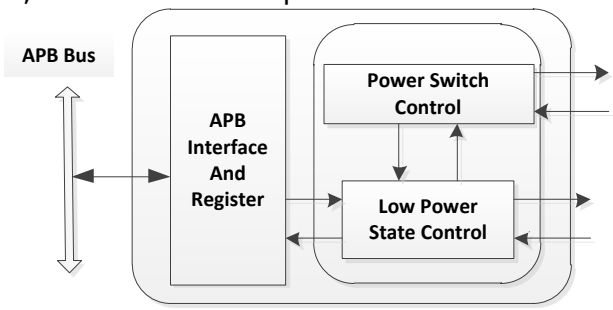


Fig 5-2 PMU Bock Diagram

5.3 Function Description

5.3.1 Normal Mode

First of all, we define two modes of power for chip, normal mode and low power mode.
In normal mode, the PMU can power off/on all power domain by setting PMU_PWRDN_CON

register.

5.3.2 Low Power Mode

PMU can work in the Low Power Mode by setting bit[0] of PMU_PWRMODE_CON register. After setting the register, PMU would enter the Low Power mode. In the low power mode, pmu will auto power on/off the specified power domain, send idle req to specified power domain, disable/enable config bus clock and so on. All of above are configurable by setting corresponding registers.

Table 5-2 Low Power State

Num	Hardware Flow	Description of Flow	Control bit
0	Normal	Normal status	
1	CLOCK_LF	Switch to 24M_div or pvtm	Bit[3] of PMU_PWRMODE_CON
2	LOGIC_PWRDN	VD_LOGIC power down	Bit[2] of PMU_PWRMODE_CON
3	WAIT WAKEUP	Wait wakeup	
4	WAIT 24M	Wait 24MHz stable	Bit[6] of PMU_PWRMODE_CON
5	CLOCK HF	Switch to 24MHz	Bit[3] of PMU_PWRMODE_CON
6	LOGIC_PWRUP	VD_LOGIC power up	

The Low Power mode have three steps:

- Enter Low Power mode, there are some sub-steps in the enter step, every sub-step can be enable/disable by setting the corresponding register.
- Wait wakeup, there is GPIO2 and GPIO2_int source by setting PMU_WAKEUP_CFG0/PMU_WAKEUP_CFG1/PMU_WAKEUP_CFG2 registers.
- Exit Low Power mode, the sub-step are executed depend on whether they were executed in entering low power step.

5.3.3 Wakeup source

There are two wakeup source, GPIO2 and GPIO2_int which can trigger PMU from low power mode to normal mode.

5.4 Register Description

5.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
PMU_PMU_WAKEUP_CFG0	0x00000	W	0x00000000	pmu wakeup configure register 0
PMU_PMU_WAKEUP_CFG1	0x00004	W	0x00000000	pmu wakeup configure register 1
PMU_PMU_WAKEUP_CFG2	0x00008	W	0x00000000	pmu wakeup configure register 2
PMU_PMU_PWRDN_CON	0x0000c	W	0x00000000	pmu power down configure register
PMU_PMU_PWRDN_ST	0x00010	W	0x00000000	pmu power down status register
PMU_PMU_PWRMODE_CON	0x00014	W	0x00000000	pmu power mode configure register
PMU_PMU_OSC_CNT	0x0001c	W	0x00005dc0	pmu osc counter register
PMU_CORE_PWRDWN_CNT	0x00020	W	0x00005dc0	core power down count register
PMU_CORE_PWRUP_CNT	0x00024	W	0x00005dc0	core power UP count register
PMU_SOFT_CON	0x00028	W	0x00000000	Internal software control register
PMU_PMU_PLLLOCK_CNT	0x0002c	W	0x00005dc0	pmu pll lock count register
PMU_PMU_INT_CON	0x00030	W	0x00000000	pmu interrupt configure register
PMU_PMU_INT_ST	0x00034	W	0x00000000	pmu interrupt status register
PMU_PMU_GPIO_POS_INT_ST	0x00038	W	0x00000000	pmu gpio posedge interrupt status register
PMU_PMU_GPIO_NEG_INT_ST	0x0003c	W	0x00000000	pmu gpio negedge interrupt status register

Name	Offset	Size	Reset Value	Description
PMU_PMU_SYS_REG0	0x00040	W	0x00000000	pmu system register 0
PMU_PMU_SYS_REG1	0x00044	W	0x00000000	pmu system register 1
PMU_PMU_SYS_REG2	0x00048	W	0x00000000	pmu system register 2
PMU_PMU_SYS_REG3	0x0004c	W	0x00000000	pmu system register 3
PMU_PMU_GPIO_POS_INT_CON	0x00060	W	0x00000000	pmu gpio posedge interrupt configure register
PMU_PMU_GPIO_NEG_INT_CON	0x00064	W	0x00000000	pmu gpio negedge interrupt configure register
PMU_SOFTRST_CON	0x00080	W	0x00000000	Internal software reset control register

5.4.2 Detail Register Description

PMU_PMU_WAKEUP_CFG0

Address: Operational Base + offset (0x00000)

pmu wakeup configure register 0

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	gpio_int_en gpio interrupt wakeup enable 0: disable 1: enable
0	RO	0x0	reserved

PMU_PMU_WAKEUP_CFG1

Address: Operational Base + offset (0x00004)

pmu wakeup configure register 1

Bit	Attr	Reset Value	Description
31:24	RW	0x00	gpio2d_posedge_en gpio2d posedge pulse wakeup enable 0: disable 1: enable
23:16	RW	0x00	gpio2c_posedge_en gpio2c posedge pulse wakeup enable 0: disable 1: enable
15:8	RW	0x00	gpio2b_posedge_en gpio2b posedge pulse wakeup enable 0: disable 1: enable
7:0	RW	0x00	gpio2a_posedge_en gpio2a posedge pulse wakeup enable 0: disable 1: enable

PMU_PMU_WAKEUP_CFG2

Address: Operational Base + offset (0x00008)
pmu wakeup configure register 2

Bit	Attr	Reset Value	Description
31:24	RW	0x00	gpio2d_negedge_en gpio2d negedge pulse wakeup enable 0: disable 1: enable
23:16	RW	0x00	gpio2c_negedge_en gpio2c negedge pulse wakeup enable 0: disable 1: enable
15:8	RW	0x00	gpio2b_negedge_en gpio2b negedge pulse wakeup enable 0: disable 1: enable
7:0	RW	0x00	gpio2a_negedge_en gpio2a negedge pulse wakeup enable 0: disable 1: enable

PMU_PMU_PWRDN_CON
Address: Operational Base + offset (0x0000c)
pmu power down configure register

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	pd_high_pwrdwn_en pd_high power down enable 0: disable 1: enable
0	RW	0x0	reserved

PMU_PMU_PWRDN_ST
Address: Operational Base + offset (0x00010)
pmu power down status register

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	pd_high_pwr_stat pd_high power state 0: powered up 1: powered down
0	RW	0x0	vd_logic_pwr_stat vd_logic power state 0: powered up 1: powered down

PMU_PMU_PWRMODE_CON

Address: Operational Base + offset (0x00014)
pmu power mode configure register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	pll_pd_en pll power down when in power mode 0: disable 1: enable
6	RW	0x0	osc_pd_en osc power down when in power mode 0: disable 1: enable
5	RW	0x0	pmu_int_en pmu interrupt enable in power mode 0: disable 1: enable
4	RW	0x0	reserved
3	RW	0x0	pmu_use_low_freq_en pmu use low frequency in power mode 0: disable 1: enable
2	RW	0x0	vd_lgc_pd_en power down vd_logic in power mode 0: disable 1: enable
1	RW	0x0	global_int_disable_en global interrupt disable during power_mode processing: 0: enable 1: disable
0	RW	0x0	power_mode_en power mode enable 0: disable 1: enable

PMU_PMU_OSC_CNT
Address: Operational Base + offset (0x0001c)
pmu osc counter register

Bit	Attr	Reset Value	Description
31:0	RW	0x00005dc0	pmu_osc_cnt pmu osc stable counter value

PMU_VD_LOGIC_PWRDWN_CNT

Address: Operational Base + offset (0x00020)
core power down count register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00005dc0	pmu_vd_logic_power_down_cnt VD_LOGIC power down stable counter value

PMU_VD_LOGIC_PWRUP_CNT
Address: Operational Base + offset (0x00024)
core power UP count register

Bit	Attr	Reset Value	Description
31:0	RW	0x00005dc0	pmu_vd_logic_power_up_cnt VD_LOGIC power up stable counter value

PMU_SOFT_CON
Address: Operational Base + offset (0x00028)
Internal software control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:3	RO	0x0	reserved
2	RW	0x0	osc_pd_en OSC power down enable 1'b0: disable 1'b1: enable
1	RW	0x0	low_clk_sel low freq clk select 1'b1: pvtm clk 1'b0: 24m div out clk
0	RW	0x0	pmu_lf_clk_sel pd_pmu use low freq clk enable 1'b0: disable 1'b1: enable

PMU_PMU_PLLLOCK_CNT
Address: Operational Base + offset (0x0002c)
pmu pll lock count register

Bit	Attr	Reset Value	Description
31:0	RW	0x00005dc0	pmu_plllock_cnt pmu pll stable lock counter value

PMU_PMU_INT_CON
Address: Operational Base + offset (0x00030)
pmu interrupt configure register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:4	RW	0x0	reserved
3	RW	0x0	vd_logic_int_en vd_logic 0: disable 1: enable
2	RW	0x0	pd_high_int_en pd_high interrupt enable 0: disable 1: enable
1	RW	0x0	pwrmode_wakeup_int_en power mode wakeup interrupt enable 0: disable 1: enable
0	RW	0x0	reserved

PMU_PMU_INT_ST
Address: Operational Base + offset (0x00034)
pmu interrupt status register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RO	0x0	cal_core_sleep_status cal_core sleep status 0: not sleep 1: sleep
5	RO	0x0	wakeup_int_logic_status vd_logic interrupt wakeup status 0: not wakeup by interrupt vd_logic 1: wakeup by interrupt vd_logic
4	RO	0x0	reserved
3	RO	0x0	wakeup_int_gpio_status gpio interrupt wakeup status 0: not wakeup by interrupt gpio 1: wakeup by interrupt gpio
2	RO	0x0	wakeup_pd_high_status pd_high wakeup status 0: not wakeup by pd_high status change 1: wakeup by pd_high status change
1	RO	0x0	pwrmode_wakeup_status power mode wakeup status 0: not wakeup from power mode change 1: wakeup from power mode change
0	RO	0x0	reserved

PMU_PMU_GPIO_POS_INT_ST
Address: Operational Base + offset (0x00038)

pmu gpio posedge interrupt status register

Bit	Attr	Reset Value	Description
31:24	RO	0x00	gpio2d_pos_int_status gpio2d posedge pulse interrupt status 0: not wakeup by gpio2d posedge pulse 1: wakeup by gpio2d posedge pulse
23:16	RO	0x00	gpio2c_pos_int_status gpio2c posedge pulse interrupt status 0: not wakeup by gpio2c posedge pulse 1: wakeup by gpio2c posedge pulse
15:8	RO	0x00	gpio2b_pos_int_status gpio2b posedge pulse interrupt status 0: not wakeup by gpio2b posedge pulse 1: wakeup by gpio2b posedge pulse
7:0	RO	0x00	gpio2a_pos_int_status gpio2a posedge pulse interrupt status 0: not wakeup by gpio2a posedge pulse 1: wakeup by gpio2a posedge pulse

PMU_PMU_GPIO_NEG_INT_ST

Address: Operational Base + offset (0x0003c)

pmu gpio negedge interrupt status register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	gpio2d_neg_int_status gpio2d negedge pulse interrupt status 0: not wakeup by gpio2d negedge pulse 1: wakeup by gpio2d negedge pulse
23:16	RW	0x00	gpio2c_neg_int_status gpio2c negedge pulse interrupt status 0: not wakeup by gpio2c negedge pulse 1: wakeup by gpio2c negedge pulse
15:8	RW	0x00	gpio2b_neg_int_status gpio2b negedge pulse interrupt status 0: not wakeup by gpio2b negedge pulse 1: wakeup by gpio2b negedge pulse
7:0	RW	0x00	gpio2a_neg_int_status gpio2a negedge pulse interrupt status 0: not wakeup by gpio2a negedge pulse 1: wakeup by gpio2a negedge pulse

PMU_PMU_SYS_REG0

Address: Operational Base + offset (0x00040)

pmu system register 0

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pmu_sys_reg0 system register 0

PMU_PMU_SYS_REG1
Address: Operational Base + offset (0x00044)
pmu system register 1

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pmu_sys_reg1 system register 1

PMU_PMU_SYS_REG2
Address: Operational Base + offset (0x00048)
pmu system register 2

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pmu_sys_reg2 system register 2

PMU_PMU_SYS_REG3
Address: Operational Base + offset (0x0004c)
pmu system register 3

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	pmu_sys_reg3 system register 3

PMU_PMU_GPIO_POS_INT_CON
Address: Operational Base + offset (0x00060)
pmu gpio posedge interrupt configure register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	gpio2d_pos_int_en gpio2d posedge pulse interrupt enable 0: disable 1: enable
23:16	RW	0x00	gpio2c_pos_int_en gpio2c posedge pulse interrupt enable 0: disable 1: enable
15:8	RW	0x00	gpio2b_pos_int_en gpio2b posedge pulse interrupt enable 0: disable 1: enable
7:0	RW	0x00	gpio2a_pos_int_en gpio2a posedge pulse interrupt enable 0: disable 1: enable

PMU_PMU_GPIO_NEG_INT_CON
Address: Operational Base + offset (0x00064)
pmu gpio negedge interrupt configure register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	gpio2d_neg_int_en gpio2d negedge pulse interrupt enable 0: disable 1: enable
23:16	RW	0x00	gpio2c_neg_int_en gpio2c negedge pulse interrupt enable 0: disable 1: enable
15:8	RW	0x00	gpio2b_neg_int_en gpio2b negedge pulse interrupt enable 0: disable 1: enable
7:0	RW	0x00	gpio2a_neg_int_en gpio2b negedge pulse interrupt enable 0: disable 1: enable

PMU_SOFTRST_CON
Address: Operational Base + offset (0x00080)
Internal software reset control register

Bit	Attr	Reset Value	Description
31:16	WO	0x0000	write_mask write mask. When every bit HIGH, enable the writing corresponding bit When every bit LOW, don't care the writing corresponding bit
15:4	RO	0x0	reserved
3	RW	0x0	pvtm_srst_sel pvtm software reset 1'b0: not reset 1'b1: reset
2	RW	0x0	grf_srst_sel grf software reset 1'b0: not reset 1'b1: reset
1	RW	0x0	gpio2_srst_sel gpio2 software reset 1'b0: not reset 1'b1: reset
0	RW	0x0	pmu_sram_srst_sel pd_pmu sram software reset 1'b0: not reset 1'b1: reset

5.5 Timing Diagram

5.5.1 Each domain power switch timing

The following figure is the each domain power down and power up timing.

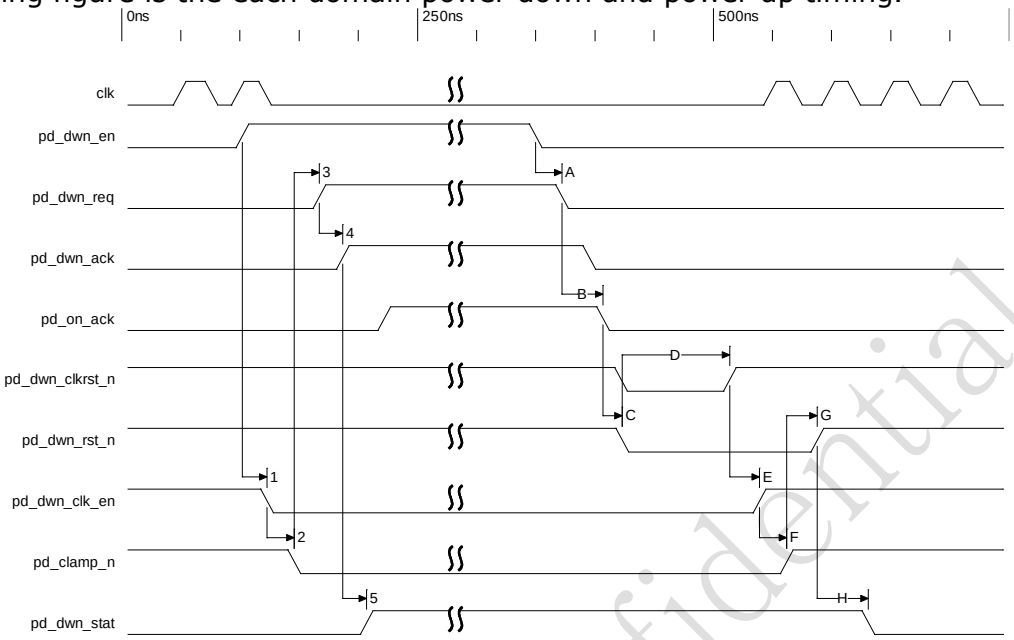


Fig 5-3 Each Domain Power Switch Timing

5.5.2 External wakeup PAD timing

The PMU supports GPIO2 as wakeup source. All these external wakeup sources must meet the timing requirement (at least 200us) when the wakeup event is asserted. The following figure gives the timing information.

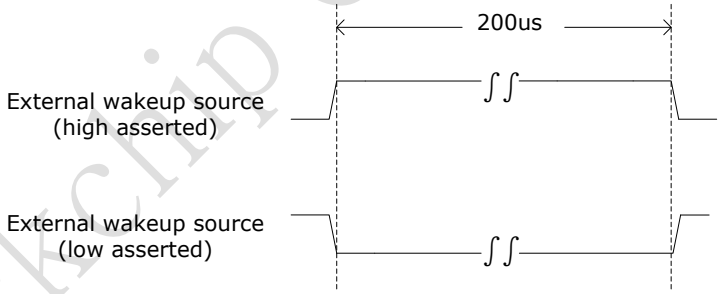


Fig 5-4 External Wakeup Source PAD Timing

5.6 Application Notes

5.6.1 Recommend configurations for power mode.

The PMU is a design with great flexibilities, but just for facilities and inheritances, a group of recommend configurations will be shown below for software. And for convenience, we will define several modes.

The chip can support following 3 recommended power modes:

- normal
- system mode
- sleep mode

Normal mode

In this mode, you can power off/on or enable/disable the following power domain to save power: PD_HIGH

System mode

In system mode, pd_high should be either power off for saving power.

Sleep mode

The sleep mode can power off VD_LOGIC except VD_PMU. The VD_LOGIC is turned off externally, and pd_high domain power off by software.
In sleep mode the clock of PD_PMU can be switched from 24MHz to low speed clock optionally by hardware. The low speed clock can be selected from clock pvtm or 24M_div clock.
In sleep mode the PLL power down mandatorily to save power by software.
In sleep mode OSC can be disabled optionally by software.

5.6.2 System Register

PMU support 4 words register: PMU_SYS_REG0, PMU_SYS_REG1, PMU_SYS_REG2, PMU_SYS_REG3. These registers are always on no matter what power mode. So software can use these registers to retain information which is useful after wakeup from any mode.

5.6.3 Configuration Constraint

In order to shut down the power domains correctly, the software must obey the rules bellow:

- Send power request to the power domain through PMU_PWRDN_CON register.
- Check PMU_PWRDN_ST register to make sure the power domain is power down.

The PD_HIGH power domain is controlled only by software. So you could power off the power domain before enter low power mode if you need.

Chapter 6 System Debug

6.1 Overview

The chip contains an Advanced High-performance Bus Access Port (AHB-AP) interface for debug accesses. An external DP component accesses this interface. The system support standard JTAG-Upland serial wire DP.

6.1.1 Features

- Invasive debug with core halted
- SW-DP
- JTAG-DP

6.2 Block Diagram

Fig 6-1 Debug system structure

6.3 Function description

6.3.1 DAP (Debug Access Port)

SWJ-DP:

The SWJ-DP is a combined JTAG-DP and SW-DP that enables you to connect either an SWD or JTAG probe to a target.

The SWJ-DP consists of a wrapper around the JTAG-DP and SW-DP. It selects JTAG or SWD as the connection mechanism and enables either JTAG-DP or SW-DP as the interface to the DAP.

AHB-AP:

The AHB-AP implements the MEM-AP architecture to directly connect to an AHB-based memory system. Connection to other memory systems is possible through suitable bridging.

6.4 Register description

For details of DEBUG registers, please reference ARM document "DDI0337G_cortex_m3_r2p0_trm" chapter 11.

6.5 Interface description

6.5.1 DAP SWJ-DP interface

The following figure is the DAP SWJ-DP interface, the SWJ-DP is a combined JTAG-DP and SW-DP that enable you connect either a Serial Wire Debug(SWJ) to JTAG probe to a target.

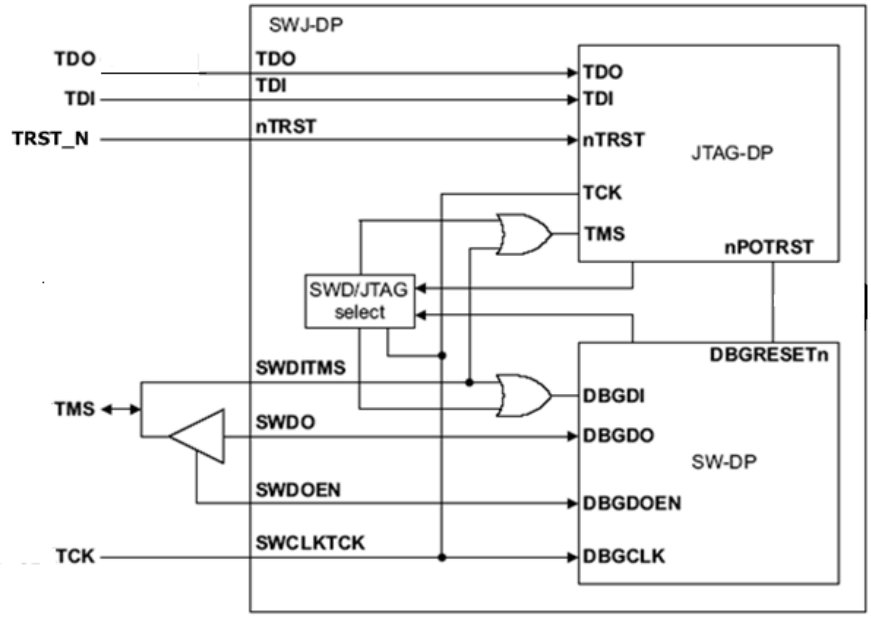


Fig 6-2 DAP SWJ interface

6.5.2 DAP SW-DP interface

This implementation is taken from ADIv5.1 and operates with a synchronous serial interface. It uses a single bidirectional data signal, and a clock signal.

Table 6-1SYS-JTAG Interface Description

Module pin	Direction	Pad name	IOMUX
SYS-JTAG Interface			
jtag0_tck	I	IO_UART1Actx_JTG0tck_SPI1Brx_GPIOP2b7	GRF_GPIO2B_IOMUX[15:14]=2'b10
jtag0_tms	I/O	IO_UART1Arts_JTG0tms_SPI1Bcs_GPIOP2b6	GRF_GPIO2B_IOMUX[13:12]=2'b10

Table 6-2CAL-JTAG Interface Description

Module pin	Direction	Pad name	IOMUX
CAL-JTAG Interface			
jtag1_tck	I	IO_UART0Atx_JTG1tck_I2C1Cscl_GPIOP2b5	GRF_GPIO2B_IOMUX[11:10]=2'b10
jtag1_tms	I/O	IO_UART0Arx_JTG1tms_I2C1Csda_GPIOP2b4	GRF_GPIO2B_IOMUX[9:8]=2'b10

Chapter 7 CPU

7.1 Overview

The processor is a low-power processor that features low gate count, low interrupt latency, and low-cost debug. It is intended for deeply embedded applications that require fast interrupt response features. The processor implements the ARMv7-M architecture. The SOC includes two processors: sys-core and cal-core.
The processor incorporates:

- Processor core
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing
- Bus interfaces
- Low-cost debug solution

7.1.1 Features

The processor features include:

- A Thumb instruction set subset
- Banked Stack Pointer (SP) only.
- Hardware divide instructions, SDIV and UDIV (Thumb 32-bit instructions).
- Handler and Thread modes.
- Support 4 flash devices at most.
- Thumb and Debug states.
- Interruptible-continued LDM/STM, PUSH/POP for low interrupt latency.
- Automatic processor state saving and restoration for low latency Interrupt Service Routine (ISR) entry and exit.
- Support for ARMv6 unaligned accesses.

The NVIC features include:

- 20 External interrupts.
- 32 levels(5 bits) of priority.
- Dynamic reprioritization of interrupts.
- Priority grouping. This enables selection of pre-empting interrupt levels and non pre-empting interrupt levels.
- Support for tail-chaining and late arrival of interrupts. This enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
- Processor state automatically saved on interrupt entry, and restored on interrupt exit, with no instruction overhead.

The Bus interface features include:

- Advanced High-performance Bus-Lite (AHB-Lite) ICode, DCode and System bus interfaces.
- Bit band support that includes atomic bit band write and read operations.
- Memory access alignment.

The Low-cost debug solution features include:

- Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESETn is asserted.
- Serial Wire Debug Port (SW-DP) or Serial Wire JTAG Debug Port (SWJ-DP)debug access, or both.

7.2 Block Diagram

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

7.2.1 Overview

The processor comprises with:

- CPU core.
- NVIC(Nested Vectored Interrupt Controller).
- Bus matrix.

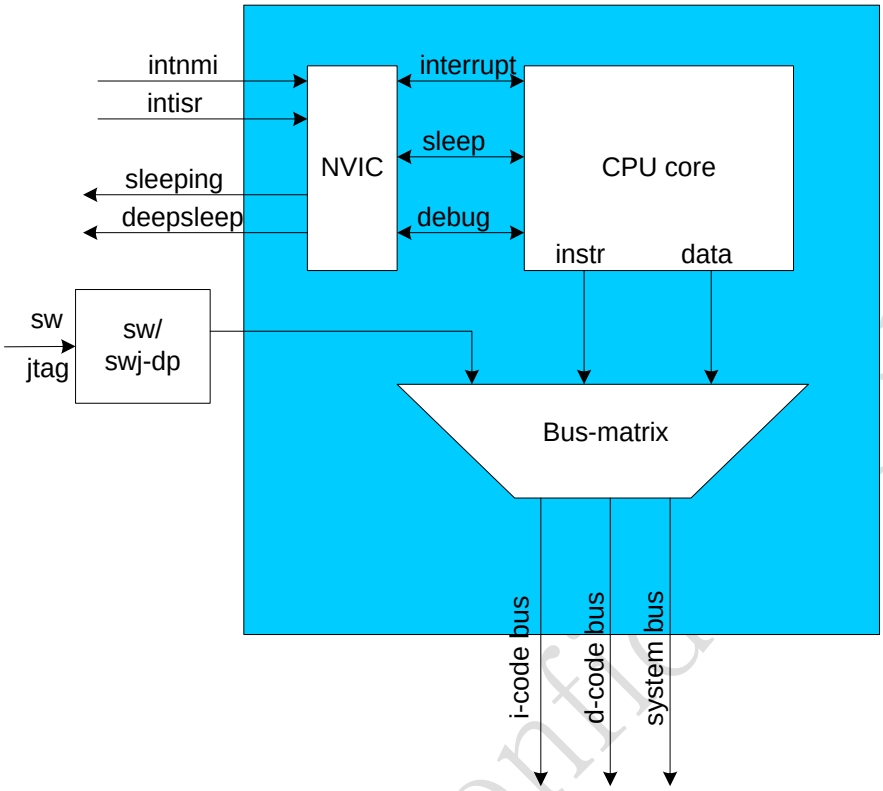


Fig 7-1processor architecture

7.2.2 Block Descriptions

CPU core

The processor core implements the ARMv7-M architecture. It has the following main features: Thumb instruction set subset, consisting of all base Thumb instructions, 16-bit and 32-bit;Harvard processor architecture enabling simultaneous instruction fetch with data load/store; Three-stage pipeline; Single cycle 32-bit multiply; Hardware divide; Thumb and Debug states; Handler and Thread modes; Low latency ISR entry and exit; Interruptible-continued LDM/STM, PUSH/POP;ARMv6 compatible unaligned access support.

NVIC

The NVIC supports up to 240 dynamically reprioritizable interrupts each with up to 256 levels of priority. The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

Bus matrix

The bus matrix connects the processor and debug interface to the external buses. The bus matrix interfaces to the following external buses: Code Bus; Code Bus; System Bus.

SW/SWJ-DP

The processor contains an Advanced High-performance Bus Access Port (AHB-AP) interface for debug accesses. An external DP component accesses this interface. The system support standard JTAG-Upland serial wire DP.

7.3 Registers

For details of CPU registers, please reference ARM document "DDI0337G_cortex_m3_r2p0_trm".

Chapter 8 DMAC

8.1 Overview

The DMAC support all kinds of burst type and data size, which define in AHB protocol. It can support on-the-fly DMA transfer on AHB bus devices and AHB bus memory. There are two DMAC in the chip. DMAC1 in pd_logic and DMAC2 in pd_high.
DMAC supports the following features:

- Provide AHB-to-AHB bus protocol translation
- Support AHB-to-AHB DMA or Single AHB DMA
- DMAC1 supports six channels, DMAC2 supports two channels
- Support byte, half word and word data transfer sizes
- Support incremental and fixed addressing mode
- Support block and software DMA transfer mode
- Support channel priority arbitration
- LLP transfer support(all channels)
- Auto-reload support(all channels)

8.2 Block Diagram

The figure below shows the block diagram of DMAC

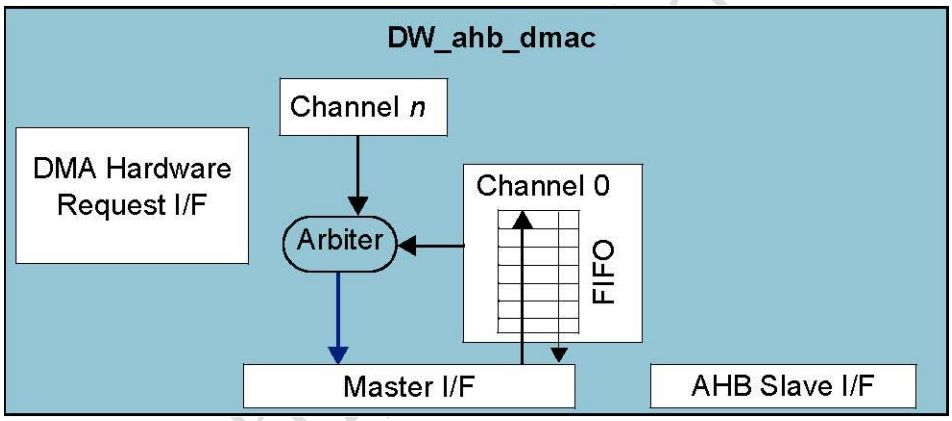


Fig 8-1Block diagram of dmac0

8.3 Register Description

8.3.1 Registers Summary for DMAC1

Name	Offset	Size	Reset Value	Description
DWDMA_SAR0	0x0000	W	0x00000000	Channel 0 Source Address Register
DWDMA_DAR0	0x0008	W	0x00000000	Channel 0 Destination Address Register
DWDMA_LLP0	0x0010	W	0x00000000	Channel 0 Linked List Pointer Register
DWDMA_CTL0	0x0018	W	0x00304801	Channel 0 Control Register Lower Bits
DWDMA_CTL0_H	0x001c	W	0x00000002	Channel 0 Control Register Higher Bits
DWDMA_SSTAT0	0x0020	W	0x00000000	Source status register for channel 0
DWDMA_DSTAT0	0x0028	W	0x00000000	Destination status register for channel 0
DWDMA_SSTATAR0	0x0030	W	0x00000000	Source status address register for channel 0
DWDMA_DSTATAR0	0x0038	W	0x00000000	Destination status address register for channel 0
DWDMA_CFG0	0x0040	W	0x00000e00	Channel 0 Configuration Register Lower Bits

Name	Offset	Size	Reset Value	Description
DWDMA_CFG0_H	0x0044	W	0x00000004	Channel 0 Configuration Register Higher Bits
DWDMA_SGR0	0x0048	W	0x00000000	Channel 0 Source Gather Register
DWDMA_DSR0	0x0050	W	0x00000000	Channel 0 Destination Scatter Register
DWDMA_SAR1	0x0058	W	0x00000000	Channel 1 Source Address Register
DWDMA_DAR1	0x0060	W	0x00000000	Channel 1 Destination Address Register
DWDMA_LLP1	0x0068	W	0x00000000	Channel 1 Linked List Pointer Register
DWDMA_CTL1	0x0070	W	0x00304801	Channel 1 Control Register Lower Bits
DWDMA_CTL1_H	0x0074	W	0x00000002	Channel 1 Control Register Higher Bits
DWDMA_SSTAT1	0x0078	W	0x00000000	Source status register for channel 1
DWDMA_DSTAT1	0x0080	W	0x00000000	Destination status register for channel 1
DWDMA_SSTATAR1	0x0088	W	0x00000000	Source status address register for channel 1
DWDMA_DSTATAR1	0x0090	W	0x00000000	Destination status address register for channel 1
DWDMA_CFG1	0x0098	W	0x00000e00	Channel 1 Configuration Register Lower Bits
DWDMA_CFG1_H	0x009c	W	0x00000004	Channel 1 Configuration Register Higher Bits
DWDMA_SGR1	0x00a0	W	0x00000000	Channel 1 Source Gather Register
DWDMA_DSR1	0x00a8	W	0x00000000	Channel 1 Destination Scatter Register
DWDMA_SAR2	0x00b0	W	0x00000000	Channel 2 Source Address Register
DWDMA_DAR2	0x00b8	W	0x00000000	Channel 2 Destination Address Register
DWDMA_LLP2	0x00c0	W	0x00000000	Channel 2 Linked List Pointer Register
DWDMA_CTL2	0x00c8	W	0x00304801	Channel 2 Control Register Lower Bits
DWDMA_CTL2_H	0x00cc	W	0x00000002	Channel 2 Control Register Higher Bits
DWDMA_SSTAT2	0x00d0	W	0x00000000	Source status register for channel 2
DWDMA_DSTAT2	0x00d8	W	0x00000000	Destination status register for channel 2
DWDMA_SSTATAR2	0x00e0	W	0x00000000	Source status address register for channel 2
DWDMA_DSTATAR2	0x00e8	W	0x00000000	Destination status address register for channel 2
DWDMA_CFG2	0x00f0	W	0x00000e00	Channel 2 Configuration Register Lower Bits
DWDMA_CFG2_H	0x00f4	W	0x00000004	Channel 2 Configuration Register Higher Bits
DWDMA_SGR2	0x00f8	W	0x00000000	Channel 2 Source Gather Register
DWDMA_DSR2	0x0100	W	0x00000000	Channel 2 Destination Scatter Register
DWDMA_SAR3	0x0108	W	0x00000000	Channel 3 Source Address Register
DWDMA_DAR3	0x0110	W	0x00000000	Channel 3 Destination Address Register
DWDMA_LLP3	0x0118	W	0x00000000	Channel 3 Linked List Pointer Register
DWDMA_CTL3	0x0120	W	0x00304801	Channel 3 Control Register Lower Bits
DWDMA_CTL3_H	0x0124	W	0x00000002	Channel 3 Control Register Higher Bits
DWDMA_SSTAT3	0x0128	W	0x00000000	Source status register for channel 3
DWDMA_DSTAT3	0x0130	W	0x00000000	Destination status register for channel 3
DWDMA_SSTATAR3	0x0138	W	0x00000000	Source status address register for channel 3
DWDMA_DSTATAR3	0x0140	W	0x00000000	Destination status address register for channel 3
DWDMA_CFG3	0x0148	W	0x00000e00	Channel 3 Configuration Register Lower Bits

Name	Offset	Size	Reset Value	Description
DWDMA_CFG3_H	0x014c	W	0x00000004	Channel 3 Configuration Register Higher Bits
DWDMA_SGR3	0x0150	W	0x00000000	Channel 3 Source Gather Register
DWDMA_DSR3	0x0158	W	0x00000000	Channel 3 Destination Scatter Register
DWDMA_SAR4	0x0160	W	0x00000000	Channel 4 Source Address Register
DWDMA_DAR4	0x0168	W	0x00000000	Channel 4 Destination Address Register
DWDMA_LLP4	0x0170	W	0x00000000	Channel 4 Linked List Pointer Register
DWDMA_CTL4	0x0178	W	0x00304801	Channel 4 Control Register Lower Bits
DWDMA_CTL4_H	0x018c	W	0x00000002	Channel 4 Control Register Higher Bits
DWDMA_SSTAT4	0x0180	W	0x00000000	Source status register for channel 4
DWDMA_DSTAT4	0x0188	W	0x00000000	Destination status register for channel 4
DWDMA_SSTATAR4	0x0190	W	0x00000000	Source status address register for channel 4
DWDMA_DSTATAR4	0x0198	W	0x00000000	Destination status address register for channel 4
DWDMA_CFG4	0x01a0	W	0x00000e00	Channel 4 Configuration Register Lower Bits
DWDMA_CFG4_H	0x01a4	W	0x00000004	Channel 4 Configuration Register Higher Bits
DWDMA_SGR4	0x01a8	W	0x00000000	Channel 4 Source Gather Register
DWDMA_DSR4	0x01b0	W	0x00000000	Channel 4 Destination Scatter Register
DWDMA_SAR5	0x01b8	W	0x00000000	Channel 5 Source Address Register
DWDMA_DAR5	0x01c0	W	0x00000000	Channel 5 Destination Address Register
DWDMA_LLP5	0x01c8	W	0x00000000	Channel 5 Linked List Pointer Register
DWDMA_CTL5	0x01d0	W	0x00304801	Channel 5 Control Register Lower Bits
DWDMA_CTL5_H	0x01d4	W	0x00000002	Channel 5 Control Register Higher Bits
DWDMA_SSTAT5	0x01d8	W	0x00000000	Source status register for channel 5
DWDMA_DSTAT5	0x01e0	W	0x00000000	Destination status register for channel 5
DWDMA_SSTATAR5	0x01e8	W	0x00000000	Source status address register for channel 5
DWDMA_DSTATAR5	0x01f0	W	0x00000000	Destination status address register for channel 5
DWDMA_CFG5	0x01f8	W	0x00000e00	Channel 5 Configuration Register Lower Bits
DWDMA_CFG5_H	0x01fc	W	0x00000004	Channel 5 Configuration Register Higher Bits
DWDMA_SGR5	0x0200	W	0x00000000	Channel 5 Source Gather Register
DWDMA_DSR5	0x0208	W	0x00000000	Channel 5 Destination Scatter Register
DWDMA_RAWTFR	0x02c0	W	0x00000000	Raw Status for IntTfr Interrupt
DWDMA_RAWBLOCK	0x02c8	W	0x00000000	Raw Status for IntBlock Interrupt
DWDMA_RAWSRCTRAN	0x02d0	W	0x00000000	Raw Status for IntSrcTran Interrupt
DWDMA_RAWDSTTRAN	0x02d8	W	0x00000000	Raw Status for IntDstTran Interrupt
DWDMA_RAWERR	0x02e0	W	0x00000000	Raw Status for IntErr Interrupt
DWDMA_STATUSTFR	0x02e8	W	0x00000000	Status for IntTfr Interrupt
DWDMA_STATUSBLOCK	0x02f0	W	0x00000000	Status for IntBlock Interrupt
DWDMA_STATUSSRCTRAN	0x02f8	W	0x00000000	Status for IntSrcTran Interrupt
DWDMA_STATUSDSTTRAN	0x0300	W	0x00000000	Status for IntDstTran Interrupt

Name	Offset	Size	Reset Value	Description
DWDMA_STATUSERR	0x0308	W	0x00000000	Status for IntErr Interrupt
DWDMA_MASKTFR	0x0310	W	0x00000000	Mask for IntTfr Interrupt
DWDMA_MASKBLOCK	0x0318	W	0x00000000	Mask for IntBlock Interrupt
DWDMA_MASKSRCTRAN	0x0320	W	0x00000000	Mask for IntSrcTran Interrupt
DWDMA_MASKDSTTRAN	0x0328	W	0x00000000	Mask for IntDstTran Interrupt
DWDMA_MASKERR	0x0330	W	0x00000000	Mask for IntErr Interrupt
DWDMA_CLEARTFR	0x0338	W	0x00000000	Clear for IntTfr Interrupt
DWDMA_CLEARBLOCK	0x0340	W	0x00000000	Clear for IntBlock Interrupt
DWDMA_CLEARSRCTRAN	0x0348	W	0x00000000	Clear for IntSrcTran Interrupt
DWDMA_CLEARDSTTRAN	0x0350	W	0x00000000	Clear for IntDstTran Interrupt
DWDMA_CLEARERR	0x0358	W	0x00000000	Clear for IntErr Interrupt
DWDMA_STATUSINT	0x0360	W	0x00000000	Status for each interrupt type
DWDMA_DMACFGREG	0x0398	W	0x00000000	DMA Configuration Register
DWDMA_CHENREG	0x03a0	W	0x00000000	DMA Channel Enable Register

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

8.3.2 Registers Summary for DMAC2

Name	Offset	Size	Reset Value	Description
DWDMA_SAR0	0x0000	W	0x00000000	Channel 0 Source Address Register
DWDMA_DAR0	0x0008	W	0x00000000	Channel 0 Destination Address Register
DWDMA_LLP0	0x0010	W	0x00000000	Channel 0 Linked List Pointer Register
DWDMA_CTL0	0x0018	W	0x00304801	Channel 0 Control Register Lower Bits
DWDMA_CTL0_H	0x001c	W	0x00000002	Channel 0 Control Register Higher Bits
DWDMA_SSTAT0	0x0020	W	0x00000000	Source status register for channel 0
DWDMA_DSTAT0	0x0028	W	0x00000000	Destination status register for channel 0
DWDMA_SSTATAR0	0x0030	W	0x00000000	Source status address register for channel 0
DWDMA_DSTATAR0	0x0038	W	0x00000000	Destination status address register for channel 0
DWDMA_CFG0	0x0040	W	0x00000e00	Channel 0 Configuration Register Lower Bits
DWDMA_CFG0_H	0x0044	W	0x00000004	Channel 0 Configuration Register Higher Bits
DWDMA_SGR0	0x0048	W	0x00000000	Channel 0 Source Gather Register
DWDMA_DSR0	0x0050	W	0x00000000	Channel 0 Destination Scatter Register
DWDMA_SAR1	0x0058	W	0x00000000	Channel 1 Source Address Register
DWDMA_DAR1	0x0060	W	0x00000000	Channel 1 Destination Address Register
DWDMA_LLP1	0x0068	W	0x00000000	Channel 1 Linked List Pointer Register
DWDMA_CTL1	0x0070	W	0x00304801	Channel 1 Control Register Lower Bits
DWDMA_CTL1_H	0x0074	W	0x00000002	Channel 1 Control Register Higher Bits
DWDMA_SSTAT1	0x0078	W	0x00000000	Source status register for channel 1

Name	Offset	Size	Reset Value	Description
DWDMA_DSTAT1	0x0080	W	0x00000000	Destination status register for channel 1
DWDMA_SSTATAR1	0x0088	W	0x00000000	Source status address register for channel 1
DWDMA_DSTATAR1	0x0090	W	0x00000000	Destination status address register for channel 1
DWDMA_CFG1	0x0098	W	0x00000e00	Channel 1 Configuration Register Lower Bits
DWDMA_CFG1_H	0x009c	W	0x00000004	Channel 1 Configuration Register Higher Bits
DWDMA_SGR1	0x00a0	W	0x00000000	Channel 1 Source Gather Register
DWDMA_DSR1	0x00a8	W	0x00000000	Channel 1 Destination Scatter Register
DWDMA_RAWTFR	0x02c0	W	0x00000000	Raw Status for IntTfr Interrupt
DWDMA_RAWBLOCK	0x02c8	W	0x00000000	Raw Status for IntBlock Interrupt
DWDMA_RAWSRCTRAN	0x02d0	W	0x00000000	Raw Status for IntSrcTran Interrupt
DWDMA_RAWDSTTRAN	0x02d8	W	0x00000000	Raw Status for IntDstTran Interrupt
DWDMA_RAWERR	0x02e0	W	0x00000000	Raw Status for IntErr Interrupt
DWDMA_STATUSTFR	0x02e8	W	0x00000000	Status for IntTfr Interrupt
DWDMA_STATUSBLOCK	0x02f0	W	0x00000000	Status for IntBlock Interrupt
DWDMA_STATUSSRCTRAN	0x02f8	W	0x00000000	Status for IntSrcTran Interrupt
DWDMA_STATUSDSTTRAN	0x0300	W	0x00000000	Status for IntDstTran Interrupt
DWDMA_STATUSERR	0x0308	W	0x00000000	Status for IntErr Interrupt
DWDMA_MASKTFR	0x0310	W	0x00000000	Mask for IntTfr Interrupt
DWDMA_MASKBLOCK	0x0318	W	0x00000000	Mask for IntBlock Interrupt
DWDMA_MASKSRCTRAN	0x0320	W	0x00000000	Mask for IntSrcTran Interrupt
DWDMA_MASKDSTTRAN	0x0328	W	0x00000000	Mask for IntDstTran Interrupt
DWDMA_MASKERR	0x0330	W	0x00000000	Mask for IntErr Interrupt
DWDMA_CLEARTFR	0x0338	W	0x00000000	Clear for IntTfr Interrupt
DWDMA_CLEARBLOCK	0x0340	W	0x00000000	Clear for IntBlock Interrupt
DWDMA_CLEARSRCTRAN	0x0348	W	0x00000000	Clear for IntSrcTran Interrupt
DWDMA_CLEARDSTTRAN	0x0350	W	0x00000000	Clear for IntDstTran Interrupt
DWDMA_CLEARERR	0x0358	W	0x00000000	Clear for IntErr Interrupt
DWDMA_STATUSINT	0x0360	W	0x00000000	Status for each interrupt type
DWDMA_DMACFGREG	0x0398	W	0x00000000	DMA Configuration Register

Name	Offset	Size	Reset Value	Description
DWDMA_CHENREG	0x03a0	W	0x00000000	DMA Channel Enable Register

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

8.3.3 Detail Register Description

DWDMA_SARx

Channel x Source Address Register
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_SAR0 – 0x0000
DWDMA_SAR1 – 0x0058
DWDMA_SAR2 – 0x00b0
DWDMA_SAR3 – 0x0108
DWDMA_SAR4 – 0x0160
DWDMA_SAR5 – 0x01b8

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	SAR 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.

DWDMA_DARx

Channel x Destination Address Register
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_DAR0 – 0x0008
DWDMA_DAR1 – 0x0060
DWDMA_DAR2 – 0x00b8
DWDMA_DAR3 – 0x0110
DWDMA_DAR4 – 0x0168
DWDMA_DAR5 – 0x01c0

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DAR 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.

DWDMA_LLPx

Channel x Linked List Pointer Register
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_LLP0 – 0x0010
DWDMA_LLP1 – 0x0068
DWDMA_LLP2 – 0x00c0
DWDMA_LLP3 – 0x0118

DWDMA_LLP4 – 0x0170
DWDMA_LLP5 – 0x01c8

Bit	Attr	Reset Value	Description
31:2	RW	0x00000000	LOC Starting AddressIn Memory of next LLI ifblock 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	RW	0x0	LMS 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

DWDMA_CTLx
Channel x Control Register Lower Bits
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_CTL0 – 0x0018
DWDMA_CTL1 – 0x0070
DWDMA_CTL2 – 0x00c8
DWDMA_CTL3 – 0x0120
DWDMA_CTL4 – 0x0178
DWDMA_CTL5 – 0x01d0

Bit	Attr	Reset Value	Description
31:29	RO	0x0	reserved
28	RW	0x0	LLP_SRC_EN Block chaining is enabled on the source side only if the LLP_SRC_EN field is high and LLPx.LOC is non-zero;
27	RW	0x0	LLP_DST_EN Block chaining is enabled on the destination side only if the LLP_DST_EN field is high and LLPx.LOC is non-zero.
26:25	RW	0x0	SMS 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]

Bit	Attr	Reset Value	Description
24:23	RW	0x0	DMS 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	RW	0x0	TT_FC Transfer Type and Flow Control. The following transfer types are supported. a. Memory to Memory b. Memory to Peripheral c. Peripheral to Memory d. Peripheral to Peripheral Flow Control can be assigned to the 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	RO	0x0	reserved
18	RW	0x0	DST_SCATTER_EN 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.

Bit	Attr	Reset Value	Description
17	RW	0x0	SRC_GATHER_EN 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.
16:14	RW	0x1	SRC_MSIZE 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.
13:11	RW	0x1	DEST_MSIZE 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.
10:9	RW	0x0	SINC 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 = Increment 01 = Decrement 1x = No change NOTE: Incrementing or decrementing is done for alignment to the next CTLx.SRC_TR_WIDTH boundary. 00= Increment 01 = Decrement 1x = No change NOTE: Incrementing or decrementing is done for alignment to the next CTLx.SRC_TR_WIDTH boundary.

Bit	Attr	Reset Value	Description
8:7	RW	0x0	DINC 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	RW	0x0	SRC_TR_WIDTH 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	RW	0x0	DST_TR_WIDTH 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	RW	0x1	INT_EN Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled

DWDMA_CTLx_H

Channel x Control Register Higher Bits
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_CTL0_H – 0x001c
DWDMA_CTL1_H – 0x0074
DWDMA_CTL2_H – 0x00cc
DWDMA_CTL3_H – 0x0124
DWDMA_CTL4_H – 0x017c
DWDMA_CTL5_H – 0x01d4

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12	RW	0x0	DONE 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.
11:0	RW	0x002	BLOCK_TS Block Transfer Size. When the 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 TLx.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 the flow controller is. 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.

Table 8-1CTLx.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 orCTLx.DST_TR_WIDTH)
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

Table 8-2 CTLx.SRC_TR_WIDTH and CTLx.DST_TR_WIDTH Decoding

CTLx.SRC_TR_WIDTH /CTLx.DST_TR_WIDTH	Size (bits)
000	8
001	16
010	32
011	64
100	128
101	256
11x	256

Table 8-3CTLx.TT_FC Field Decoding

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

DWDMA_SSTATx

Source status register for channel x
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_SSTAT0 – 0x0020
DWDMA_SSTAT1 – 0x0078
DWDMA_SSTAT2 – 0x00d0
DWDMA_SSTAT3 – 0x0128
DWDMA_SSTAT4 – 0x0180
DWDMA_SSTAT5 – 0x01d8

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	SSTAT Source status information retrieved by hardware from the address pointed to by the contents of the SSTATARx register.

DWDMA_DSTATx

Destination status register for channel x
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_DSTAT0 – 0x0028
DWDMA_DSTAT1 – 0x0080
DWDMA_DSTAT2 – 0x00d8
DWDMA_DSTAT3 – 0x0130
DWDMA_DSTAT4 – 0x0188
DWDMA_DSTAT5 – 0x01e0

Bit	Attr	Reset Value	Description
63:32	RO	0x0	reserved

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DSTAT Destination status information retrieved by hardware from the address pointed to by the contents of the DSTATARx register.

DWDMA_SSTATARx
Source status address register for channel x
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_SSTATAR0 – 0x0030
DWDMA_SSTATAR0 – 0x0088
DWDMA_SSTATAR0 – 0x00e0
DWDMA_SSTATAR0 – 0x0138
DWDMA_SSTATAR0 – 0x0190
DWDMA_SSTATAR0 – 0x01e8

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	SSTATAR Pointer from where hardware can fetch the source status information, which is registered in the SSTATx register and written out to the SSTATx register location of the LLI before the start of the next block.

DWDMA_DSTATARx
Destination status address register for channel x
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_DSTATAR0 – 0x0038
DWDMA_DSTATAR1 – 0x0090
DWDMA_DSTATAR2 – 0x00e8
DWDMA_DSTATAR3 – 0x0140
DWDMA_DSTATAR4 – 0x0198
DWDMA_DSTATAR5 – 0x01f0

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DSTATAR Pointer from where hardware can fetch the destination status information, which is registered in the DSTATx register and written out to the DSTATx register location of the LLIbefore the start of the next block.

DWDMA_CFGx
Channel x Configuration Register Lower Bits
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_CFG0 – 0x0040
DWDMA_CFG1 – 0x0098
DWDMA_CFG2 – 0x00f0
DWDMA_CFG3 – 0x0148
DWDMA_CFG4 – 0x01a0
DWDMA_CFG5 – 0x01f8

Bit	Attr	Reset Value	Description
31	RW	0x0	RELOAD_DST 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 theninitiated. For conditions under which this occurs, refer to Table 5.
30	RW	0x0	RELOAD_SRC 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	RW	0x000	MAX_ABRST 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	RW	0x0	SRC_HS_POL Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
18	RW	0x0	DST_HS_POL Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
17	RW	0x0	LOCK_B Bus Lock Bit. When active, the AHB bus master signal hlock is asserted for the duration specified in CFGx.LOCK_B_L.
16	RW	0x0	LOCK_CH 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.

Bit	Attr	Reset Value	Description
15:14	RW	0x0	LOCK_B_L Bus Lock Level. Indicates the duration over whichCFGx.LOCK_B bit applies. 00 = Over complete DMA transfer 01 = Over complete DMA block transfer 1x = Over complete DMA transaction
13:12	RW	0x0	LOCK_CH_L 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	RW	0x1	HS_SEL_SRC 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	RW	0x1	HS_SEL_DST 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.

Bit	Attr	Reset Value	Description
9	RO	0x0	FIFO_EMPTY 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	RW	0x0	CH_SUSP 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	RW	0x0	CH_PRIOR 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. Reset Value: Channel Number. for example: Chan0=0 Chan1=1
4:0	RO	0x0	reserved

DWDMA_CFGx_H

Channel x Configuration Register Higher Bits
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_CFG0 – 0x0044
DWDMA_CFG1 – 0x009c
DWDMA_CFG2 – 0x00f4
DWDMA_CFG3 – 0x014c
DWDMA_CFG4 – 0x01a4
DWDMA_CFG5 – 0x01fc

Bit	Attr	Reset Value	Description
31:15	RO	0x0	reserved

Bit	Attr	Reset Value	Description
14:11	RW	0x0	DEST_PER 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.
10:7	RW	0x0	SRC_PER 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 DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
6	RW	0x0	SS_UPD_EN 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.
5	RW	0x0	DS_UPD_EN 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.

Bit	Attr	Reset Value	Description
4:2	RW	0x1	PROTCTL Protection Control bits used to drive the AHBHPROT[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.
1	RW	0x0	FIFO_MODE 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.
0	RW	0x0	FCMODE 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.

Table 8-4 PROTCTL field to HPROT Mapping

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

DWDMA_SGRx
Channel x Source Gather Register
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_SGR0 – 0x0048
DWDMA_SGR1 – 0x00a0
DWDMA_SGR2 – 0x00f8
DWDMA_SGR3 – 0x0150
DWDMA_SGR4 – 0x01a8
DWDMA_SGR5 – 0x0200

Bit	Attr	Reset Value	Description
31:20	RW	0x000	SGC 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	RW	0x00000	SGI Source gather interval.

DWDMA_DSRx
Channel x Destination Scatter Register
Address: Operational Base + offset
Address Offset: x = 0 to 5for DMAC1, x = 0 to 1 for DMAC2:
DWDMA_DSR0 – 0x0050
DWDMA_DSR1 – 0x00a8
DWDMA_DSR2 – 0x0100
DWDMA_DSR3 – 0x0158
DWDMA_DSR4 – 0x01b0
DWDMA_DSR5 – 0x0208

Bit	Attr	Reset Value	Description
31:20	RW	0x000	DSC Destination scatter count. Destination contiguous transfer count between successive scatter boundaries. b = log2 (DMAH_CHx_MAX_BLK_SIZE + 1) + 19 Bits31:b+1 do not exist and read 0.
19:0	RW	0x00000	DSI Destination scatter interval.

DWDMA_RAWTFR
Address: Operational Base + offset (0x02c0)
Raw Status for IntTfr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved

Bit	Attr	Reset Value	Description
n-1:0	RO	0x0	RAW Raw interrupt status

DWDMA_RAWBLOCK
Address: Operational Base + offset (0x02c8)
Raw Status for IntBlock Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	RAW Raw interrupt status

DWDMA_RAWSRCTRAN
Address: Operational Base + offset (0x02d0)
Raw Status for IntSrcTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	RAW Raw interrupt status

DWDMA_RAWDSTTRAN
Address: Operational Base + offset (0x02d8)
Raw Status for IntDstTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	RAW Raw interrupt status

DWDMA_RAWERR
Address: Operational Base + offset (0x02e0)
Raw Status for IntErr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	RAW Raw interrupt status

DWDMA_STATUSTFR
Address: Operational Base + offset (0x02e8)
Status for IntTfr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	STATUS Interrupt status

DWDMA_STATUSBLOCK
Address: Operational Base + offset (0x02f0)

Status for IntBlock Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	STATUS Interrupt status

DWDMA_STATUSSRCTRAN
Address: Operational Base + offset (0x02f8)
Status for IntSrcTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	STATUS Interrupt status

DWDMA_STATUSDSTTRAN
Address: Operational Base + offset (0x0300)
Status for IntDstTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	STATUS Interrupt status

DWDMA_STATUSERR
Address: Operational Base + offset (0x0308)
Status for IntErr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	RO	0x0	STATUS Interrupt status

DWDMA_MASKTFR
Address: Operational Base + offset (0x0310)
Mask for IntTfr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:8+n	RO	0x0	reserved
7+n:8	WO	0x0	INT_MASK_WE Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:n	RO	0x0	reserved
n-1:0	RW	0x0	INT_MASK Interrupt Mask 0 = masked 1 = unmasked

DWDMA_MASKBLOCK
Address: Operational Base + offset (0x0318)
Mask for IntBlock Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:8+n	RO	0x0	reserved
7+n:8	WO	0x0	INT_MASK_WE Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:n	RO	0x0	reserved
n-1:0	RW	0x0	INT_MASK Interrupt Mask 0 = masked 1 = unmasked

DWDMA_MASKSRCTRAN
Address: Operational Base + offset (0x0320)
Mask for IntSrcTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:8+n	RO	0x0	reserved
7+n:8	WO	0x0	INT_MASK_WE Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:n	RO	0x0	reserved
n-1:0	RW	0x0	INT_MASK Interrupt Mask 0 = masked 1 = unmasked

DWDMA_MASKDSTTRAN
Address: Operational Base + offset (0x0328)
Mask for IntDstTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:8+n	RO	0x0	reserved
7+n:8	WO	0x0	INT_MASK_WE Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:n	RO	0x0	reserved
n-1:0	RW	0x0	INT_MASK Interrupt Mask 0 = masked 1 = unmasked

DWDMA_MASKERR
Address: Operational Base + offset (0x0330)
Mask for IntErr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:8+n	RO	0x0	reserved
7+n:8	WO	0x0	INT_MASK_WE Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
7:n	RO	0x0	reserved
n-1:0	RW	0x0	INT_MASK Interrupt Mask 0 = masked 1 = unmasked

DWDMA_CLEARTFR
Address: Operational Base + offset (0x0338)
Clear for IntTfr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	WO	0x0	CLEAR Interrupt clear. 0 = no effect 1 = clear interrupt

DWDMA_CLEARBLOCK
Address: Operational Base + offset (0x0340)
Clear for IntBlock Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	WO	0x0	CLEAR Interrupt clear. 0 = no effect 1 = clear interrupt

DWDMA_CLEARSRCTRAN
Address: Operational Base + offset (0x0348)
Clear for IntSrcTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	WO	0x0	CLEAR Interrupt clear. 0 = no effect 1 = clear interrupt

DWDMA_CLEARDSTTRAN

Address: Operational Base + offset (0x0350)
Clear for IntDstTran Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	WO	0x0	CLEAR Interrupt clear. 0 = no effect 1 = clear interrupt

DWDMA_CLEARERR
Address: Operational Base + offset (0x0358)
Clear for IntErr Interrupt
n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:n	RO	0x0	reserved
n-1:0	WO	0x0	CLEAR Interrupt clear. 0 = no effect 1 = clear interrupt

DWDMA_STATUSINT
Address: Operational Base + offset (0x0360)
Status for each interrupt type

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RO	0x0	ERR OR of the contents of Status Err register
3	RO	0x0	DSTT OR of the contents of StatusDst register
2	RO	0x0	SRCT OR of the contents of StatusSrcTran register.
1	RO	0x0	BLOCK OR of the contents of StatusBlock register
0	RO	0x0	TFR OR of the contents of StatusTfr register.

DWDMA_DMACFGREG
Address: Operational Base + offset (0x0398)
DMA Configuration Register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	DMA_EN DMA Enable bit. 0 = DMA Disabled 1 = DMA Enabled.

DWDMA_CHENREG
Address: Operational Base + offset (0x03a0)
DMA Channel Enable Register

n = 6for DMAC1, n = 2 for DMAC2

Bit	Attr	Reset Value	Description
31:8+n	RO	0x0	reserved
7+n:8	WO	0x0	CH_EN_WE Channel enable write enable.
7:n	RO	0x0	reserved
n-1:0	RW	0x0	CH_EN 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.

8.4 Application Notes

8.4.1 Illegal Register Access

An illegal access can be any of the following:

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

The response to an illegal access is configured using the configuration parameter DMAH_RETURN_ERR_RESP. When DMAH_RETURN_ERR_RESP is set to True, an illegal access (read/write) returns an error response.
If DMAH_RETURN_ERR_RESP is set to False, an OKAY response is returned, a read reads back 0x0, and a write is ignored.

8.4.2 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 dmac is reprogrammed using either of the following methods:

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

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

- Block chaining using linked lists
- Auto-reloading

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

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

Multi-Block Transfers

Multi-block transfers are enabled by setting theDMAH_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 dmac reprograms the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

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

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

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

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

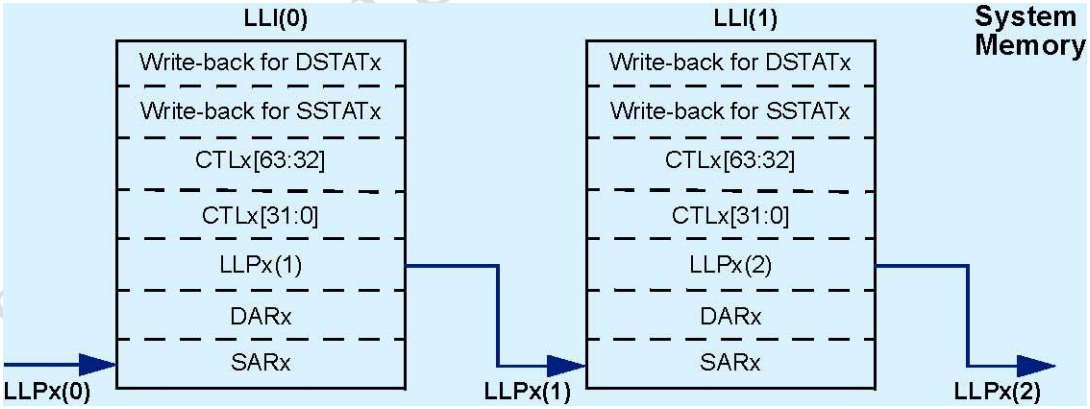


Fig 8-2 Multi-Block Transfer Using Linked Lists When DMAH_CHx_STAT_SRC Set toTrue

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

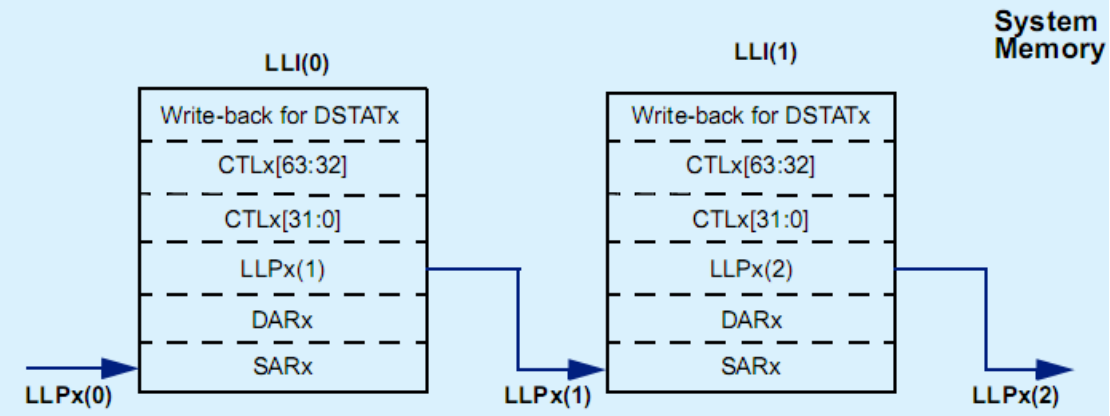


Fig 8-3 Multi-Block Transfer Using Linked Lists WhenDMAH_CHx_STAT_SRC Set to false

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

For rows 6 through 10 of following Table, 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 dmacLLPx 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 dmacSARx/DARx registers.

Table 8-5Programming 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,LLPxUpdateMethod**	**SARxUpdateMethod**	**DARxUpdateMethod**	**WriteBack[a]**
1. Single-block orlast transfer ofmulti-block	Yes	0	0	0	0	None, user reprograms	None (single)	None (single)	No
2.Auto-reloadmulti-blocktransfer with contiguous SAR	Yes	0	0	0	1	CTLx, LLPxare reloaded from initial values	Contiguous	Auto-reload	No
3.Auto-reloadmulti-blocktransfer with contiguous DAR.	Yes	0	1	0	0	CTLx, LLPxare reloadedfrom initialvalues	Auto-reload	Contiguous	No
4.Auto-reloadmulti-blocktransfer	Yes	0	1	0	1	CTLx, LLPxare reloaded from initial values	Auto-reload	Auto-reload	No
5. Single-block orlast transfer ofmulti-block.	No	0	0	0	0	None, user reprograms	None (single)	None (single)	Yes
6. Linked listmulti-blocktransfer with contiguous SAR	No	0	0	1	0	CTLx, LLPxloaded from next LinkedList item.	Contiguous	Linked List	Yes
7. Linked listmulti-block transfer with auto-reload SAR	No	0	1	1	0	CTLx, LLPxloaded from next LinkedList item	Auto-reload	Linked List	Yes
8. Linked list multi-block transfer with contiguous DAR	No	1	0	0	0	CTLx, LLPxloaded from next LinkedList item	Linked List	Contiguous	Yes
9. Linked list multi-block transfer with auto-reload DAR	No	1	0	0	1	CTLx, LLPxloaded from next Linked List item	Linked List	Auto-Reload	Yes
10. Linked list multi-block transfer	No	1	0	1	0	CTLx, LLPxloaded 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”.

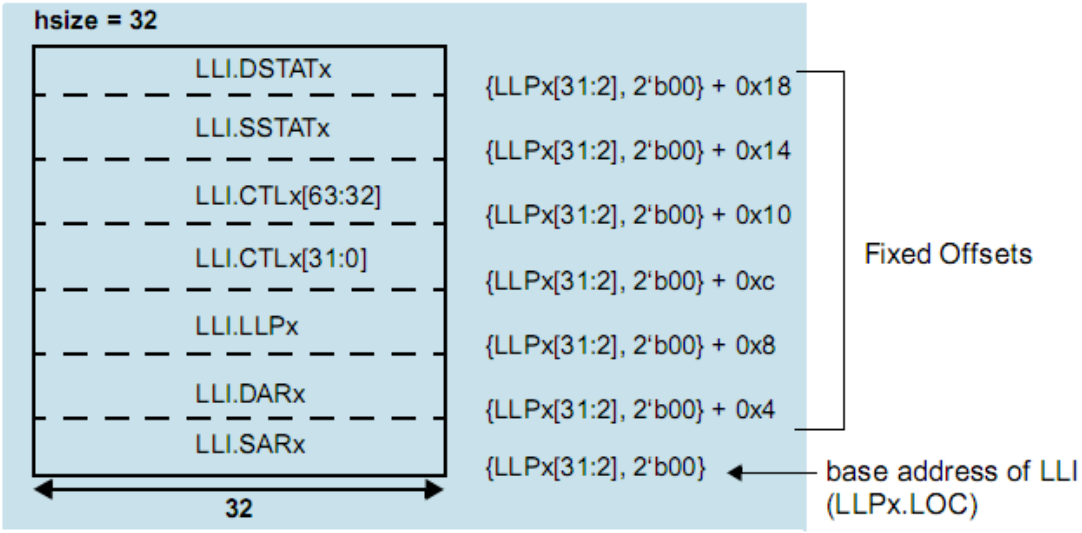


Fig 8-4Mapping of Block Descriptor (LLI) in Memory to Channel Registers WhenDMAH_CHx_STAT_SRC Set to True

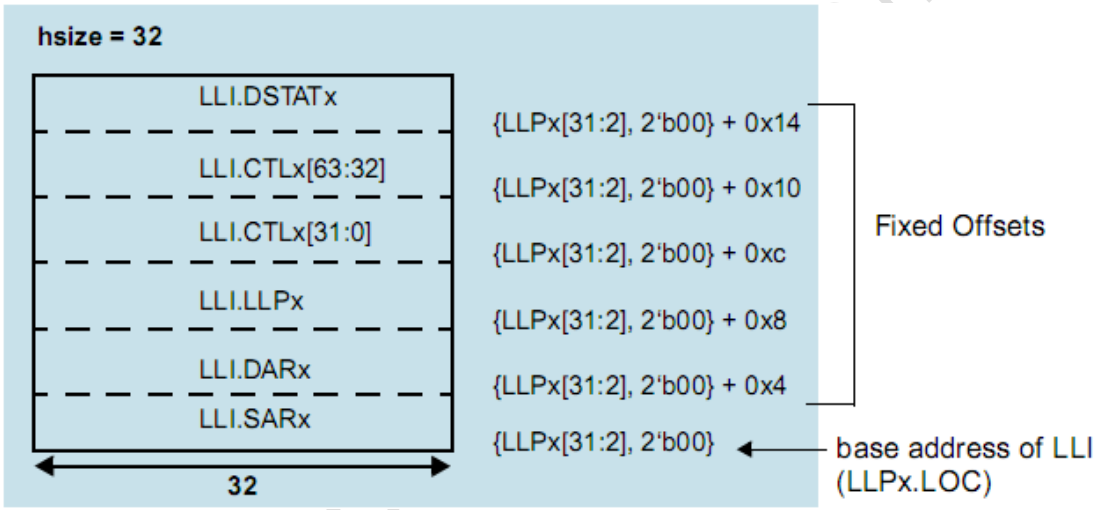


Fig 8-5Mapping of Block Descriptor (LLI) in Memory to Channel Registers WhenDMAH_CHx_STAT_SRC Set to False

Chapter 9 ACODEC

9.1 Overview

ACODEC is a high resolution stereo audio CODEC that employs Sigma-Delta technique, with 24 bits resolution for DAC and DAC.
APB slave peripheral can be used to prevent system lockup that may be caused by conflicting parts or programs in a SoC. The WDT would generate interrupt or reset signal when its counter reaches zero, then a reset controller would reset the system.
ACODEC supports the following features:

- 24 bit DAC which support headphone and line-out.
- 24 bit ADC which support microphone and line-in.
- One internal PLL
- Support microphone input and line input
- Support I2S as the digital signal interface for both DAC and DAC.
- Support APB as the configure interface.
- Support sample rate:
  8khz,16khz,32kHz,64kHz,128khz,11.025khz,22.05khz,44.1khz,88.2khz,176.4khz,12khz,24khz,48khz,96khz,192khz.
- Support Automatic Level Control (ALC), limiter and noise gating.
- Support programmable digital and analog gains.

9.2 Block Diagram

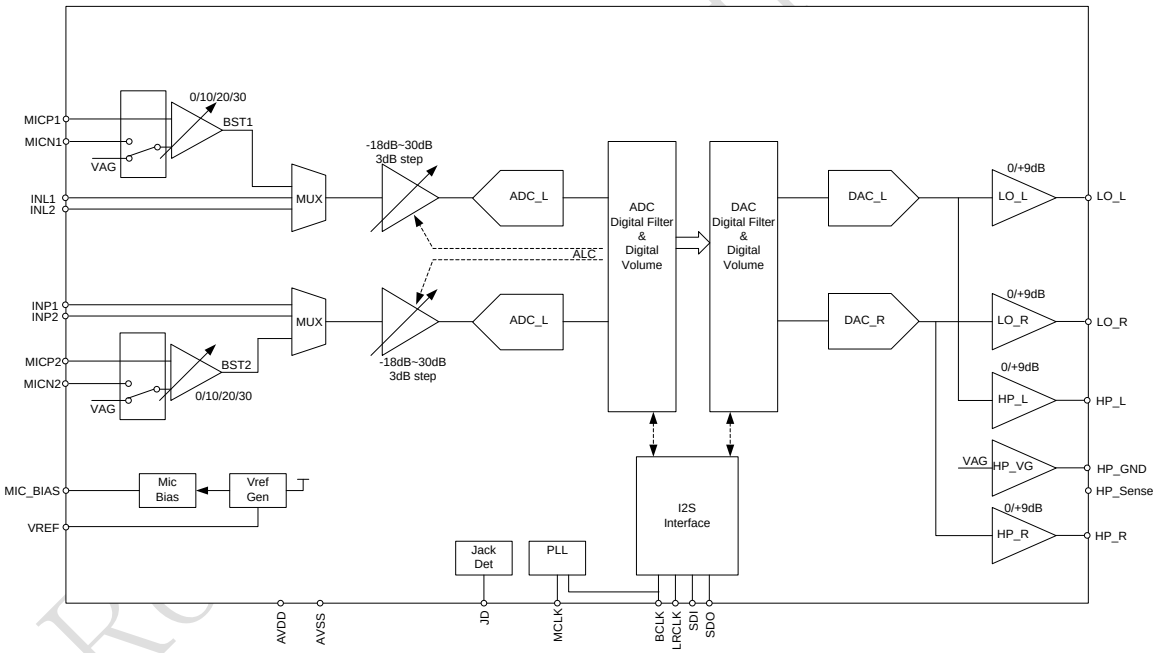


Fig 9-1Acodec block diagram

9.3 Function description

9.3.1 Operation

ACODEC initialization

CODEC Initialization is the first configuration step.

PLL configuration

This IP has one PLL to generate internal high frequency clock and the frequency should be different for each audio sample rate group. The relationship is listed below:

Sample rate group	PLL target frequency
8Khz/16Khz/32Khz/64Khz/128Khz	40.96Mhz
11.025Khz/22.05Khz/44.1Khz/88.2Khz/176.4Khz	56.448Mhz
12Khz/24Khz/48Khz/96Khz/192Khz	61.44Mhz

The source clock of PLL can be selected between "i_clk_sys" and "i_sclk_i2s" while PLL function is: F_target = F_source*PLL_POSDIV/(PLL_PREDIV_BIT*PLL_OUTDIV).

If audio CODEC works in master mode, one constant frequency clock outside should be supported through the port "i_clk_sys". And the PLL configure flow should be called before ADC or DAC starts, when PLL has not been configured or audio sample rate group is changed. There's an example listed below (assume that the frequency of "i_clk_sys" is 24Mhz and target frequency is 61.44Mhz):

- PLL power down &reset:
  - set #PLLCFG5.PLL_OUTDIV_EN = 0x0
  - set #PLLCFG5.PLL_RESET = 0x1
  - set #PLLCFG5.PLL_PWD = 0x1

- Select i_clk_sys as the source clock of PLL
  - set #PLLCFG0. PLL_CLKIN_SEL = 0x2
- Configure PLL function:
  - set #PLLCFG1. PLL_POSDIV_L3 = 0x0
  - set #PLLCFG2. PLL_POSDIV_H8 = 0x30
  - set #PLLCFG3.PLL_PREDIV_BIT = 0x19
  - set #PLLCFG4. PLL_OUTDIV = 0x6

  note: the 5 lines above are the PLL default configurations.
- PLL power on & release reset:
  - set #PLLCFG5.PLL_PWD = 0x0
  - set #PLLCFG5.PLL_RESET = 0x0
- wait 200us for pll lock
- enable PLL_OUTDIV clk
  - set #PLLCFG5.PLL_OUTDIV_EN = 0x1

If audio CODEC works in slave mode, the source clock of PLL can be selected between i_sclk_i2s (as I2S BCLK) or i_clk_sys (as I2S MCLK). But the frequency of selected clock must be faster than 2Mhz.

PLL configure flow should be called before ADC or DAC starts, when PLL has not been configured or PLL function needs to be changed. Note that: because the source clock frequency is changed accordingly with sample rate, PLL function could be the same mostly. There's an example listed below (assume that "i_sys_clk" is selected and the frequency is 12Khz*256 = 3.072Mhz):

- PLL power down & reset:
  - set #PLLCFG5.PLL_OUTDIV_EN = 0x0
  - set #PLLCFG5.PLL_RESET = 0x1
  - set #PLLCFG5.PLL_PWD = 0x1

- Select i_sys_clk as the source clock of PLL
  - set #PLLCFG0. PLL_CLKIN_SEL = 0x2
- Configure PLL function:
  - set #PLLCFG1. PLL_POSDIV_L3 = 0x0
  - set #PLLCFG2. PLL_POSDIV_H8 = 0x2D
  - set #PLLCFG3.PLL_PREDIV_BIT = 0x3
  - set #PLLCFG4. PLL_OUTDIV = 0x6
- PLL power on & release reset:
  - set #PLLCFG5.PLL_PWD = 0x0
  - set #PLLCFG5.PLL_RESET = 0x0

- wait 200us for pll lock
- enable PLL_OUTDIV clk
  - set #PLLCFG5.PLL_OUTDIV_EN = 0x1

Here is the architecture of CRU module:

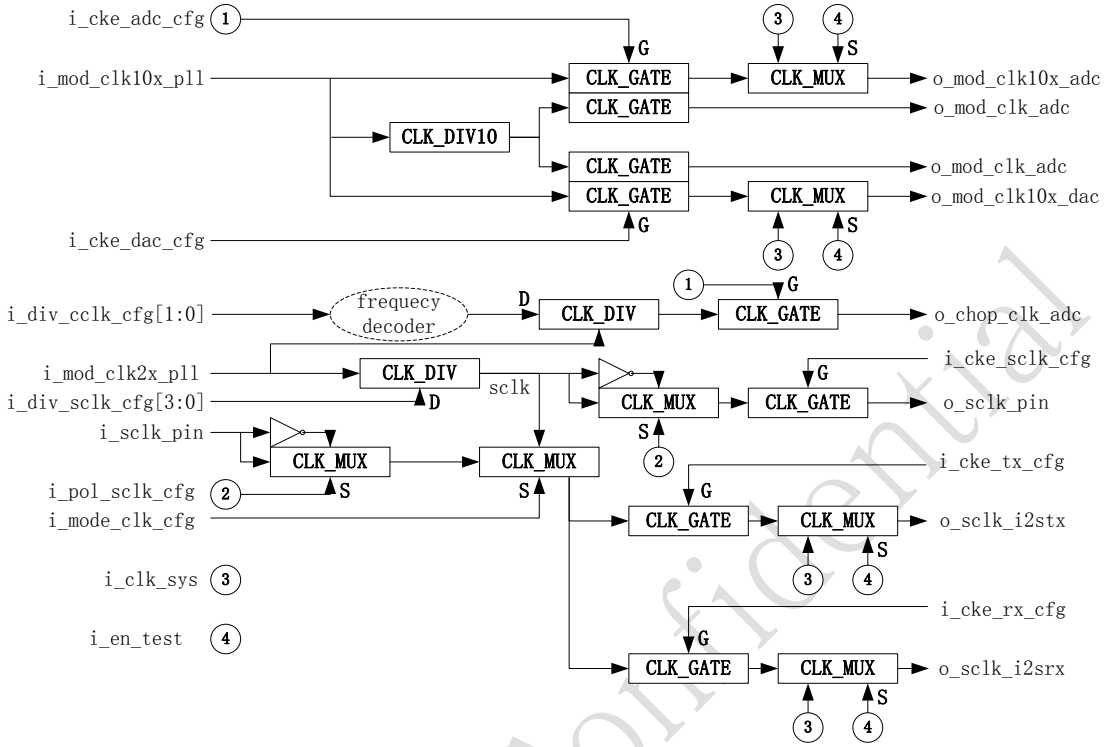


Fig 9-2 Architecture of CRU module

I2S configuration

This module is an interface for transmitting audio data, which can send ADC data and receive DAC data.

In slave mode, the master I2S can not generate the lrck with perfect frequency, such as 192Khz, 44.1Khz, 8Khz and so on. So the master mode is recommended. The following text will focus on how to config with the master mode.

The configuration of I2S can be divided by two steps:

- I2S clk configuration

  I2S clk configuration is to set the sample rate, the frequency of sck and work mode(master/slave). There is an example listed below (assume that sample rate is 192Khz, 32bit for left/right channel, work in 61.44Mhz):
  - set #I2STXCR0.SCKD_TX = 0x0, 64 sck per period of lrck
  - set #I2SRXCR0.SCKD_RX = 0x0, 64 sck per period of lrck
  - set #I2SCKM.SCK_DIV = 0x0, sck_div = F(clk)/5/F(sclk) - 1
  - set #I2SCKM.I2S_MST = 0x1, master mode

  note: I2STXCR0. TXRL_P/I2SRXCR0.RXRL_P/I2SCKM.SCK_P also can be set for timing.
- I2S RX/TX configuration

  I2S RX/TX configuration is to set the specific protocol. There are two examples listed below(assume that I2S normal mode, msb, 24bits valid data width)

  TX:
  - set #I2STXCR1.TFS_TX = 0x0, I2S mode
  - set #I2STXCR1.IBM_TX = 0x0, normal mode

- set #I2STXCR1.LSB_TX = 0X0, msb
- set #I2STXCR2 = 0x17, 24 bits valid data width

RX:

- set #I2SRXCR1.TFS_RX = 0x0, I2S mode
- set #I2SRXCR1.IBM_RX = 0x0, normal mode
- set #I2SRXCR1.LSB_RX = 0X0, msb
- set #I2SRXCR2 = 0x17, 24 bits valid data width

ADC configuration

Before config or power on ADC, the ADC should be muted.

There is the ADC configuration main flow:

- set #ADCCFG1. ADC_MUTE_L = 0x1 and #ADCCFG1.ADC_MUTE_R = 0x1, mute the left/right channel of ADC
- set #LICFG1.MUX_L_IN_SEL and #LICFG1.MUX_R_IN_SEL, switch the source of ADC
- set follow volume control registers:
  #VCTL.ADC_BYPS = 0
  #VCTL.ADCFade = 1
  #VCTL.ADCZDT = 1
  #VCTIME(shared with DAC volume control)
- set #ADCSR, to set sample rate:
- set #ADCCFG0.ADC_L_PWD = 0x0 and #ADCCFG0.ADC_R_PWD = 0x0, power on analog part of ADC
- set #CLKE.ADC_CKE = 1 and #CLKE.I2STX_CKE = 1, enable ADC and i2s tx clk
- set #DIGEN.I2STX_EN = 1, enable i2s tx
- set #I2STXCMD.TXS = 0x01, tx transfer start
- wait 1ms
- set #DIGEN.ADC_EN = 1, enable adc
- wait 1ms
- set #ADCCFG1. ADC_MUTE_L = 0x0 and #ADCCFG1.ADC_MUTE_R = 0x0, unmute the left/right channel of ADC

The above configuration should be set before ADC enabled, but follow functions can be set after it: ALC/limiter/noise_gate/high pass filter. There is an example of alc listed below:

- set #ALC0 = 0x0, disable ALC
- set #ALC1 and #ALC2, set the registers of ALC
- set #ALC0 = 0xc0, enable ALC

DAC configuration

There is the DAC configuration main flow:

- ref top power on with antipop flow:
  set #RTCFG2.REF_PWD = 0;
  wait 200ms;
  #RTCFG2.VAG_BUF_PWD = 0;
  #RTCFG2.IBIAS_PWD = 0;
  wait 6s;
- set follow volume control registers:
  #VCTL.DAC_BYPS = 0
  #VCTL.DACFade = 1
  #VCTL.DACZDT = 1

#VCTIME (shared with ADC volume control)

- set #DACSR, to set sample rate:
- set #CLKE.DAC_CKE = 1 and #CLKE.I2SRX_CKE = 1, enable ADC and i2s rx clk
- set #DIGEN.I2SRX_EN = 1, enable i2s rx
- set #I2SRXCMD.RXS = 0x01, rx transfer start
- wait 1ms
- set #DIGEN.DAC_EN = 1, enable DAC
- set #DACPOPD for auto power on
- set #DACPOPD.ATPO = 1, power on analog part of DAC

DACsong switching flow

- set #DACPOPD.ATPO = 0, and wait 1ms, power down analog part of DAC
- set #DIGEN.DAC_EN = 0, disable digital part of DAC
- set #DIGEN.I2SRX_EN = 0, disable I2S RX
- set #CLKE.DAC_CKE = 0 and #CLKE.I2SRX_CKE = 0, disable ADC and i2s rx clk
- run PLL configuration flow for new sample rate of next song
- run I2S RX configuration flow
- run DAC configuration flow(without ref top power on flow)

9.3.2 Programming sequence

9.4 Register Description

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

9.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
ACODEC_VCTL	0x0040	W	0x00000003	volume control register
ACODEC_VCTIME	0x0044	W	0x00000000	volume control time
ACODEC_LPST	0x0048	W	0x00000000	low power status
ACODEC_LPT	0x004c	W	0x00000000	low power threshold
ACODEC_SRST	0x0054	W	0x00000000	soft reset
ACODEC_DIGEN	0x0058	W	0x00000000	digital module enable
ACODEC_CLKE	0x0060	W	0x00000000	clock enable
ACODEC_RTCFG0	0x0080	W	0x00000000	analog REF TOP configure0
ACODEC_RTCFG1	0x0084	W	0x00000000	analog REF TOP configure1
ACODEC_RTCFG2	0x0088	W	0x00000007	analog REF TOP configure2
ACODEC_ADCCFG0	0x00c0	W	0x000000d8	analog ADC config0
ACODEC_ADCCFG1	0x00c4	W	0x000000c0	analog ADC config1
ACODEC_ADCVCTLL	0x00c8	W	0x00000000	ADC left channel digital volume control register
ACODEC_ADCVCTLR	0x00cc	W	0x00000000	ADC right channel digital volume control register
ACODEC_ADCSR	0x00d0	W	0x00000000	ADC sample rate
ACODEC_ALC0	0x00d4	W	0x00000000	ADC ALC configure register0
ACODEC_ALC1	0x00d8	W	0x00000000	ADC ALC configure register1
ACODEC_ALC2	0x00dc	W	0x00000000	ADC ALC configure register2
ACODEC_ADCNG	0x00e0	W	0x00000000	ADC noise gate
ACODEC_ADCNGST	0x00e4	W	0x00000000	ADC noise gate status

Name	Offset	Size	Reset Value	Description
ACODEC_ADCHPF	0x00e8	W	0x00000000	ADC high pass filter
ACODEC_ADCVSTL	0x00ec	W	0x000000ff	ADC volume status for left channel
ACODEC_ADCVSTR	0x00f0	W	0x000000ff	ADC volume status for right channel
ACODEC_DACACFG0	0x0100	W	0x00000000	the No.0 of DAC analog configure registers
ACODEC_DACACFG1	0x0104	W	0x00000000	the No.1 of DAC analog configure registers
ACODEC_DACACFG2	0x0108	W	0x00000007	the No.2 of DAC analog configure registers
ACODEC_DACPOPD	0x0140	W	0x000000b0	DAC power on and power down
ACODEC_DACST	0x0144	W	0x00000010	DAC status
ACODEC_DACVCTLL	0x0148	W	0x00000000	ADC left channel digital volume control register
ACODEC_DACVCTLR	0x014c	W	0x00000000	ADC right channel digital volume control register
ACODEC_DACSR	0x0150	W	0x00000000	DAC sample rate
ACODEC_LMT0	0x0154	W	0x00000000	No.0 of DAC limiter registers
ACODEC_LMT1	0x0158	W	0x00000000	No.1 of DAC limiter registers
ACODEC_LMT2	0x015c	W	0x00000000	No.2 of DAC limiter registers
ACODEC_DACMUTE	0x0160	W	0x00000000	DAC mute
ACODEC_MIXCTRL	0x0164	W	0x00000000	mixer control register
ACODEC_DACVSTL	0x0168	W	0x00000000	DAC left channel volume status
ACODEC_DACVSTR	0x016c	W	0x00000000	DAC right channel volume status
ACODEC_LICFG0	0x0180	W	0x00000000	ADC line-in configure register0
ACODEC_LICFG1	0x0184	W	0x00000000	ADC line-in configure register1
ACODEC_LICFG2	0x0188	W	0x00000066	ADC line-in configure register2
ACODEC_LICFG3	0x018c	W	0x00000000	ADC line-in configure register3
ACODEC_LICFG4	0x0190	W	0x00000000	ADC line-in configure register4
ACODEC_LILMT0	0x0198	W	0x00000000	line-in limiter configure register0
ACODEC_LILMT1	0x019c	W	0x00000000	line-in limiter configure register1
ACODEC_LILMT2	0x01a0	W	0x00000000	line-in limiter configure register2
ACODEC_ADCNGLMTCFG	0x01a4	W	0x00000000	ADC limiter noise gate configure register
ACODEC_ADCNGLMTST	0x01a8	W	0x00000000	ADC limiter noise gate status
ACODEC_HPLOCFG0	0x01c0	W	0x00000000	headphone and line-out configure register0
ACODEC_HPLOCFG1	0x01c4	W	0x00000000	headphone and line-out configure register1
ACODEC_HPLOCFG2	0x01c8	W	0x00000000	headphone and line-out configure register2
ACODEC_HPLOCFG3	0x01cc	W	0x0000000f	headphone and line-out configure register3

Name	Offset	Size	Reset Value	Description
ACODEC_HPLOCFG4	0x01d0	W	0x00000000	headphone and line-out configure register4
ACODEC_HPLOCFG5	0x01d4	W	0x00000000	headphone and line-out configure register5
ACODEC_PLLCFG0	0x0200	W	0x00000014	PLL configure register0
ACODEC_PLLCFG1	0x0204	W	0x00000000	PLL configure register1
ACODEC_PLLCFG2	0x0208	W	0x00000030	PLL configure register2
ACODEC_PLLCFG3	0x020c	W	0x00000019	PLL configure register3
ACODEC_PLLCFG4	0x0210	W	0x00000065	PLL configure register4
ACODEC_PLLCFG5	0x0214	W	0x00000000	PLL configure register5
ACODEC_I2SCKM	0x0240	W	0x00000000	I2S clock mode
ACODEC_I2SRXCR0	0x0244	W	0x00000000	I2S Rx control register0
ACODEC_I2SRXCR1	0x0248	W	0x00000000	I2S Rx control register1
ACODEC_I2SRXCR2	0x024c	W	0x00000017	I2S Rx control register2
ACODEC_I2SRXCMD	0x0250	W	0x00000000	I2S Rx command
ACODEC_I2STXCR0	0x0260	W	0x00000000	I2S Tx configure register0
ACODEC_I2STXCR1	0x0264	W	0x00000000	I2S Tx configure register1
ACODEC_I2STXCR2	0x0268	W	0x00000017	I2S Tx configure register2
ACODEC_I2STXCR3	0x026c	W	0x00000000	I2S Tx configure register3
ACODEC_I2STXCMD	0x0270	W	0x00000000	I2S Tx command
ACODEC_TMCFG0	0x0300	W	0x00000000	test mode configure register

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

9.4.2 Detail Register Description

Operational Base Address of ACcodec is 0x40090000.

ACODEC_VCTL

Address: Operational Base + offset (0x0040)
volume control register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	ADCBYPS ADC volume control bypass 1'b0: ADC volume control enable 1'b1: ADC volume control bypass
6	RW	0x0	DACBYPS DAC volume control bypass 1'b0: DAC volume control enable 1'b1: DAC volume control bypass
5	RW	0x0	ADCFADE ADC fade ADC volume adjust mode 1'b0: update to new volume immediately. 1'b1: update volume as ADCCZDT field describes.

4	RW	0x0	DACFADE DAC fade DAC volume adjust mode 1'b0: update to new volume immediately. 1'b1: update volume as DACCZDT field describes.
3:2	RO	0x0	reserved
1	RW	0x1	ADCCZDT ADC cross zero detect ADC cross-zero detect enable. It works when ADC_BYPS is 0 and ADC_FADE is 1. 1'b0: volume adjusts every sample. 1'b1: volume adjusts only when audio waveform crosses zero or volume-control time-limit condition meets.
0	RW	0x1	DACCZDT DAC cross-zero detect DAC cross-zero detect enable. It works when DAC_BYPS is 0 and DAC_FADE is 1. 1'b0: volume adjusts every sample. 1'b1: volume adjusts only when audio waveform crosses zero or volume-control time-limit condition meets.

ACODEC_VCTIME
Address: Operational Base + offset (0x0044)
volume control time

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	VCTLMT volume control time limit volume control time limit, valid only in fade cross zero mode time limit = VCTLMT * (1/sample rate) unit: lrck

ACODEC_LPST
Address: Operational Base + offset (0x0048)
low power status

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved

Bit	Attr	Reset Value	Description
0	RO	0x0	LPDET low power detect valid signal detected, valid when DAC automatically power-on and power-down enabled. 1'b0: no valid signal detected. 1'b1: valid signal detected.

ACODEC_LPT
Address: Operational Base + offset (0x004c)
low power threshold

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	LPT low power threshold valid signal detect threshold:-6db * LPT

ACODEC_SRST
Address: Operational Base + offset (0x0054)
soft reset

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	SRST soft reset write 1 to reset all internal register. hardware will reset this field to 0 after soft reset finishes automatically.

ACODEC_DIGEN
Address: Operational Base + offset (0x0058)
digital module enable

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	ADC_EN digital ADC portion enable 1'b0: digital ADC portion disabled. 1'b1: digital ADC portion enabled.
4	RW	0x0	I2STX_EN I2S Tx enable 1'b0: I2S Tx disabled. 1'b1: I2S Tx enabled.
3:2	RO	0x0	reserved

Bit	Attr	Reset Value	Description
1	RW	0x0	DAC_EN digital DAC portion enable 1'b0: digital DAC portion disabled. 1'b1: digital DAC portion enabled.
0	RW	0x0	I2SRX_EN I2S Rx enable 1'b0: I2S Rx module disabled. 1'b1: I2S Rx module enabled.

ACODEC_CLKE
Address: Operational Base + offset (0x0060)
clock enable

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	ADC_CKE ADC digital portion enable ADC digital portion enable 1'b1: enable 1'b0: disable
4	RW	0x0	I2STX_CKE I2S Tx clock enable 1'b1: I2S Tx clock enable. 1'b0: I2S Tx clock disabled.
3:2	RO	0x0	reserved
1	RW	0x0	DAC_CKE DAC digital portion clock enable DAC digital portion clock enable 1'b1: enable 1'b0: disable
0	RW	0x0	I2SRX_CKE I2S Rx clock enable 1'b1: I2S Rx clock enable. 1'b0: I2S Rx clock disabled.

ACODEC_RTCFG0
Address: Operational Base + offset (0x0080)
analog REF TOP configure0(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	HALF_ADC_BUF half ADC reference buffer drive ability decrease the buffer bias current to decrease the drive ability. 1'b1: enable 1'b0: disable

Bit	Attr	Reset Value	Description
6	RW	0x0	HALF_VAG_BUF half VAG buffer drive ability decrease the VAG buffer drive ability in REF TOP. 1'b1: enable 1'b0: disable
5:4	RW	0x0	IBIAS_ANA_SEL total bias current select the total bias current. 2'b00: 100% 2'b01: 80% 2'b10: 120% 2'b11: 140%
3:0	RW	0x0	IBIAS_BIT bias current configure field select the bias current of opamp. from 33% to 200%. OP current select: 0000 -> 100% 0001 -> 70% 0010 -> 62.5% 0011 -> 50% 0100 -> 50% 0101 -> 41% 0110 -> 38% 0111 -> 33% 1000 -> 200% 1001 -> 140% 1010 -> 120% 1011 -> 100% 1100 -> 100% 1101 -> 80% 1110 -> 75% 1111 -> 66%

ACODEC_RTCFG1

Address: Operational Base + offset (0x0084)

analog REF TOP configure1(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

7	RW	0x0	BOOST_ADC_BUF increase the ADC reference buffer drive ability double the ADC buffer bias current to increase its drive ability. 1'b1: enable 1'b0: disable
6	RW	0x0	BOOST_VAG_BUF increase VAG buffer drive ability increase the VAG buffer drive ability in REF TOP. 1'b1: enable 1'b0: disable
5	RW	0x0	REF_ADC_SEL ADC reference voltage select 0: 1.2V 1: 1.5V
4:3	RW	0x0	VAG_SEL VAG voltage select select the VAG voltage 00 -> 1.6V 01 -> 1.47V 10 -> 1.745V 11-> 1.92V
2:0	RW	0x0	VBG_TRIM VBG TRIM change the VBG voltage to make it stable in 1.2V select the bandgap voltage: 000 -> 1.189V 001 -> 1.204V 010 -> 1.219V 011 -> 1.233V 100 -> 1.248V 101 -> 1.263V 110 -> 1.277V 111 -> 1.292V

ACODEC_RTCFG2

Address: Operational Base + offset (0x0088)

analog REF TOP configure2(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x1	IBIAS_PWD IBIAS power down 1'b0: IBIAS power on. 1'b1: IBIAS power down.

Bit	Attr	Reset Value	Description
1	RW	0x1	VAG_BUF_PWD VAG buffer power down 1'b0: VAG buffer power on. 1'b1: VAG buffer power down.
0	RW	0x1	REF_PWD REF TOP block power down 1'b0: REF TOP power on. 1'b1: REF TOP power down.

ACODEC_ADCCFG0

Address: Operational Base + offset (0x00c0)

analog ADC config0(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x1	ADC_L_PWD ADC left channel power down 1'b0: ADC right channel power on 1'b1: ADC right channel power down
6	RW	0x1	ADC_R_PWD ADC right channel power down 1'b0: ADC right channel power on 1'b1: ADC right channel power down
5	RW	0x0	ADC_CLK_EDGE_SEL select ADC output data and clock edge relationship 0-> use the ADC falling edge to send the ADC data 1-> using the ADC rising edge to send the ADC data
4	RW	0x1	ADC_DEM_EN ADC dem enable 1'b0: disable ADC dem strategy. 1'b1: enable ADC dem strategy.
3	RW	0x1	ADC_DITH_OFF ADC dither disable 1'b0: enable ADC dither. 1'b1: disable ADC dither.

Bit	Attr	Reset Value	Description
2:0	RW	0x0	ADC_DITH_SEL ADC dither frequency selector ADC_dither_clk select: 000 -> 1/50 of ADC clock 001 -> 1/33 010 -> 1/20 011 -> 1/15 100 -> 1/10 101 -> 1/8 110 -> 1/6 111 -> 1/4

ACODEC_ADCCFG1
Address: Operational Base + offset (0x00c4)
analog ADC config1(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x1	ADC_MUTE_L mute the left channel of ADC 1'b1: enable 1'b0: disable
6	RW	0x1	ADC_MUTE_R mute the right channel of ADC 1'b1: enable 1'b0: disable
5	RW	0x0	ADC_ATTN_ALLIBIAS decrease the total ADC ibias current 1'b1: enable 1'b0: disable
4	RW	0x0	ADC_ATTN_OPBIAS decrease the opamp bias current 1'b1: enable 1'b0: disable
3	RW	0x0	ADC_DLY_INC increase the delay time of ADC of ADC clock signal 1'b1: enable 1'b0: disable
2	RW	0x0	ADC_OVERLAP_INC increase the non-overlap time of ADC clock signal 1'b1: enable 1'b0: disable

Bit	Attr	Reset Value	Description
1	RW	0x0	ADC_BOOST_OPAMP increase the opamp bias current 1'b1: enable 1'b0: disable
0	RW	0x0	ADC_BOOST_VAGOP increase the VAG buffer bias current 1'b1: enable 1'b0: disable

ACODEC_ADCVCTLL
Address: Operational Base + offset (0x00c8)
ADC left channel digital volume control register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	ADCVOLL ADC left channel digital volume control register ADC left channel digital volume control register, 0.375db/step. 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db ... 8'hff: -95 db

ACODEC_ADCVCTLR
Address: Operational Base + offset (0x00cc)
ADC right channel digital volume control register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	ADCVOLR ADC right channel digital volume control register ADC right channel digital volume control register, 0.375db/step. 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db ... 8'hff: -95 db

ACODEC_ADCSR
Address: Operational Base + offset (0x00d0)

ADC sample rate

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2:0	RW	0x0	ADCSRT ADC sample rate time ADC sample rate = 8khz/11.025khz/12khz * power(2,ADCSRT). note that sample rate base(8khz/11.025khz/12khz) is decided by PLL configuration.

ACODEC_ALC0

Address: Operational Base + offset (0x00d4)

ADC ALC configure register0

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	ALCL ADC left channel ALC enable automatic level control enable for ADC left channel 1'b1: enable 1'b0: disable
6	RW	0x0	ALCR ADC right channel ALC enable automatic level control enable for ADC right channel 1'b1: enable 1'b0: disable
5:0	RO	0x0	reserved

ACODEC_ALC1

Address: Operational Base + offset (0x00d8)

ADC ALC configure register1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	ALCARATE ALC attack rate ALC attack rate = sample rate/ (8*power(2,ALCARATE))
3:0	RW	0x0	ALCRRATE ALC release rate ALC release rate = sample rate/ (8*power(2,ALCRRATE))

ACODEC_ALC2

Address: Operational Base + offset (0x00dc)

ADC ALC configure register2

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6:4	RW	0x0	ALCMAX ALC maximum threshold The maximum threshold of ALC: 3'b000: 0db 3'b001: -3db 3'b010: -6db 3'b011: -9db 3'b100: -12db 3'b101: -18db 3'b110: -24db 3'b111: -30db
3	RO	0x0	reserved
2:0	RW	0x0	ALCMIN ALC minimum threshold The minimum threshold of ALC: 3'b000: 0db 3'b001: -3db 3'b010: -6db 3'b011: -9db 3'b100: -12db 3'b101: -18db 3'b110: -24db 3'b111: -30db

ACODEC_ADCNG

Address: Operational Base + offset (0x00e0)

ADC noise gate

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	NGCHL noise gate channel 1'b0: either channel 1'b1: both channel
6	RW	0x0	NGEN noise gate enable 1'b0: noise gate disable 1'b1: noise gate enable
5	RW	0x0	NGBOOST noise gate boost 1'b0: normal noise gate 1'b1: boost noise gate

Bit	Attr	Reset Value	Description
4:2	RW	0x0	NGGATE noise gate threshold noise gate threshold 3'b000~3'b111: -63db~-84db, 3db/step when NGBOOST is 0; otherwise -33db~-54db, 3db/step.
1:0	RW	0x0	NGDLY noise gate delay the delay time before the noise gate works 2'b00: 2048 samples 2'b01: 4096 samples 2'b10: 8192 samples 2'b11: 16384 samples

ACODEC_ADCNGST
Address: Operational Base + offset (0x00e4)
ADC noise gate status

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	NGVALID noise gate valid status 1'b0: ADC is not in noise gate status 1'b1: ADC is in noise gate status

ACODEC_ADCHPF
Address: Operational Base + offset (0x00e8)
ADC high pass filter

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	HPFLE high pass filter enable for left channel 1'b0: high pass filter for right channel is disabled. 1'b1: high pass filter for right channel is enabled.
6	RW	0x0	HPFRE high pass filter enable for right channel 1'b0: high pass filter for right channel is disabled. 1'b1: high pass filter for right channel is enabled.

Bit	Attr	Reset Value	Description
5:4	RW	0x0	HPF_CF high pass filter configure register high pass filter configure register 2'b00: 3.79Hz 2'b01: 60Hz 2'b10: 243Hz 2'b11: 493Hz
3:0	RO	0x0	reserved

ACODEC_ADCVSTL
Address: Operational Base + offset (0x00ec)
ADC volume status for left channel

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0xff	ADCVSTL ADC volume status for left channel ADC volume status for left channel, read only

ACODEC_ADCVSTR
Address: Operational Base + offset (0x00f0)
ADC volume status for right channel

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0xff	ADCVSTR ADC volume status for right channel ADC volume status for right channel

ACODEC_DACCFG0
Address: Operational Base + offset (0x0100)

(debug only, maintain default value)Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	DAC_AMP_INC2DB increase the DAC differential output about 2dB increase the DAC differential output about 2dB 1'b1: enable 1'b0: disable
6	RW	0x0	DAC_CLK_EDGE_SEL DAC clock edge selector 0 -> using the falling edge of DAC clock to catch the DAC data. 1 -> using the rising edge of DAC clock to catch the DAC data.

5	RW	0x0	DAC_MAX_OUT Field0000 Abstract make DAC to the maximum output 1'b1: enable 1'b0: disable
4	RW	0x0	INC_DAC_RSTB increase the DAC internal RSTB increase the DAC internal RSTB 1'b1: enable 1'b0: disable
3:2	RW	0x0	DAC_NBIAS_SEL DAC NMOS bias selector change the circuit internal NBIAS current. 00 -> original design 01 -> bias current change to 80% 10 -> bias current to 120% 11 -> bias current to 140%
1:0	RW	0x0	DAC_PBIAS_SEL DAC PMOS bias selector change the internal PBIAS current: 00 -> 100% current 01 -> 80% current 10 -> 120% current 11 -> 140% current

ACODEC_DACCFG1
Address: Operational Base + offset (0x0104)
(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	INC_DAC_SWITCH increase the DAC internal switch signal control time increase the DAC internal switch signal control time 1'b1: enable 1'b0: disable
1	RW	0x0	STOP_DAC_RSTB stop the rstb clock stop the DAC internal RSTB clock. 1'b1: enable 1'b0: disable

0	RW	0x0	STOP_DAC_SW stop the switch clock in DAC stop the DAC internal switch clock. 1'b1: enable 1'b0: disable

ACODEC_DACCFG2
Address: Operational Base + offset (0x0108)
the No.2 of DAC analog configure registers

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x1	PWD_DACBIAS power down DAC bias 1'b1: power down DAC bias 1'b0: power down DAC bias
1	RW	0x1	PWD_DACL power down DAC left channel 1'b1: power down DAC left channel 1'b0: power on DAC left channel
0	RW	0x1	PWD_DACR power down DAC right channel 1'b1: power down DAC right channel 1'b0: power on DAC right channel

ACODEC_DACPOPD
Address: Operational Base + offset (0x0140)
DAC power on and power down

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x1	ATPCE automatically power control enable 1'b0: automatical power control is disabled. 1'b1: automatical power control is enabled.
6	RW	0x0	HPSEL head-phone selector 1'b0: select line-out as DAC output. 1'b1: select headphone as DAC output.
5	RW	0x1	SMTPO smart power on 1'b0: smart power on is disabled. 1'b1: smart power on is enabled.
4	RW	0x1	ANTIPOP anti-pop 1'b0: anti-pop flow is disabled 1'b1: anti-pop flow is enabled
3:1	RO	0x0	reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	DACATPO DAC automatical power-on 1'b0: DAC automatical power-down 1'b1: DAC automatical power-on

ACODEC_DACST
Address: Operational Base + offset (0x0144)
DAC status

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RO	0x1	DAC_MTST DAC mute status 1'b0: DAC is not in mute status 1'b1: DAC is in mote status
3:1	RO	0x0	reserved
0	RO	0x0	DAC_PWRST DAC power status 1'b0: DAC is powered on. 1'b0: DAC is powered down.

ACODEC_DACVCTLL
Address: Operational Base + offset (0x0148)
ADC left channel digital volume control register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	DACLV DAC left channel volume digital volume of DAC left channel, 0.375db/step: 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db ... 8'hff: -95 db

ACODEC_DACVCTLR
Address: Operational Base + offset (0x014c)
ADC right channel digital volume control register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:0	RW	0x00	DACRV DAC right channel volume digital volume of DAC right channel, 0.375db/step: 8'h0: 0db 8'h1: -0.375db 8'h2: -0.75db 8'h3: -1.125db ... 8'hff: -95 db

ACODEC_DACSR
Address: Operational Base + offset (0x0150)
DAC sample rate

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2:0	RW	0x0	DACSRT DAC sample rate times DAC sample rate = 8khz/11.025khz/12khz * power(2,DACSRT). note that sample rate base(8khz/11.025khz/12khz) is decided by PLL configuration.

ACODEC_LMT0
Address: Operational Base + offset (0x0154)
No.0 of DAC limiter registers

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	LMTEN limiter enable 1'b0: limiter is disabled. 1'b1: limiter is enabled.
6	RW	0x0	LMTCHL limiter channel 1'b0: combinational signal (left+right/2) detected. 1'b1: detect independently.
5:0	RO	0x0	reserved

ACODEC_LMT1
Address: Operational Base + offset (0x0158)
No.1 of DAC limiter registers

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:4	RW	0x0	LMTRRATE limiter release rate limiter release rate: 8* power(2,LMTARATE) samples
3:0	RW	0x0	LMTARATE limiter attack rate limiter attack rate: 8* power(2,LMTARATE) samples

ACODEC_LMT2
Address: Operational Base + offset (0x015c)
No.2 of DAC limiter registers

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6:4	RW	0x0	LMTMAX limiter maximum threshold limiter maximum threshold: 3'b000~3'b100: 0db~-12db, 3db/step; 3'b101~3'b111: -18db~-30db, 6db/step;
3	RO	0x0	reserved
2:0	RW	0x0	LMTMIN limiter minimum threshold limiter minimum threshold: 3'b000~3'b100: 0db~-12db, 3db/step; 3'b101~3'b111: -18db~-30db, 6db/step;

ACODEC_DACMUTE
Address: Operational Base + offset (0x0160)
DAC mute

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	DACMTE DAC mute enable 1'b0: DAC mute is disabled. 1'b1: DAC mute is enabled.

ACODEC_MIXCTRL
Address: Operational Base + offset (0x0164)
mixer control register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	MIXE mixer enable 1'b0: mixer is disabled. 1'b1: mixer is enabled.

ACODEC_DACVSTL
Address: Operational Base + offset (0x0168)
DAC left channel volume status

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	DACVSTL DAC left channel volume status DAC left channel volume status, read only

ACODEC_DACVSTR
Address: Operational Base + offset (0x016c)
DAC right channel volume status

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	DACVSTR DAC right channel volume status DAC right channel volume status, read only.

ACODEC_LICFG0
Address: Operational Base + offset (0x0180)
ADC line-in configure register0

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6:4	RW	0x0	MICBIAS_SEL select the output voltage of MIC bias select the mic bias voltage 000 -> 1.5V 001 -> 1.8V 010 -> 2.0V 011 -> 2.2V 100 -> 2.5V 101 -> 2.8V 110 -> 3.0V 111 -> 3.3V
3:2	RW	0x0	MIC_L_BOOST left channel microphone analog gain left channel microphone analog gain 2'b00: 0dB 2'b01: 10dB 2'b10: 20dB 2'b11: 30dB

Bit	Attr	Reset Value	Description
1:0	RW	0x0	MIC_R_BOOST right channel microphone analog gain right channel microphone analog gain 2'b00: 0dB 2'b01: 10dB 2'b10: 20dB 2'b11: 30dB

ACODEC_LICFG1
Address: Operational Base + offset (0x0184)
ADC line-in configure register1

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	MIC_L_DIFF_EN left channel difference mic mode control register 0: disable 1: enable
4	RW	0x0	MIC_R_DIFF_EN right channel difference mic mode control register 0: disable 1: enable
3:2	RW	0x0	MUX_L_IN_SEL select the left channel MUX input source 00: Line_1 01: Line_2 10: Mic
1:0	RW	0x0	MUX_R_IN_SEL select the right channel MUX input source 00: Line_1 01: Line_2 10: Mic

ACODEC_LICFG2
Address: Operational Base + offset (0x0188)
ADC line-in configure register2

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x6	MUX_L_GAIN change the gain of MUX block, the value changed from -18dB to 27dB, 3db/step. 0000: -18db 1111: 27db

3:0	RW	0x6	MUX_R_GAIN change the gain of MUX block, the value changed from -18dB to 27dB, 3db/step. 0000: -18db 1111: 27db

ACODEC_LICFG3
Address: Operational Base + offset (0x018c)
ADC line-in configure register3

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:2	RW	0x0	CHOP_SEL chopping clk freq sel the chopping clock is generated from acodec cru, and used in line-in block 00: 200k 01: 400k 10: 800k 11: reserve
1	RW	0x0	MIC_CHOP_EN mic input block open chopping function switch. 1'b1: enable 1'b0: disable
0	RW	0x0	MUX_CHOP_EN enable the chopping function of MUX block in line-in 1'b1: enable 1'b0: disable

ACODEC_LICFG4
Address: Operational Base + offset (0x0190)
ADC line-in configure register4

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:2	RW	0x0	CHOP_SEL chopping clk freq sel the chopping clock is generated from acodec cru, and used in line-in block 00: 200k 01: 400k 10: 800k 11: reserve

1	RW	0x0	MIC_CHOP_EN mic input block open chopping function switch. 1'b1: enable 1'b0: disable
0	RW	0x0	MUX_CHOP_EN enable the chopping function of MUX block in line-in 1'b1: enable 1'b0: disable

ACODEC_LILMT0
Address: Operational Base + offset (0x0198)
line-in limiter configure register0

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	

ACODEC_LILMT1
Address: Operational Base + offset (0x019c)
line-in limiter configure register1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	MAX_LILMT The highest threshold of LIMITER 000~100: 0db~-12db, 3db/step 101~111: -18db~-30db, 6db/step
3:0	RO	0x0	reserved

ACODEC_LILMT2
Address: Operational Base + offset (0x01a0)
line-in limiter configure register2

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x0	ATK_RATE_LILMT LIMITER Attack rate = (2^ATK_RATE_LILMT) * (8*clk1x) clk1x is such as 4.096Mhz, 5.6448Mhz, 6.144Mhz
3:0	RW	0x0	RLS_RATE_LILMT LIMITER Release rate = (2^RLS_RATE_LILMT) * (8*clk1x) clk1x is such as 4.096Mhz, 5.6448Mhz, 6.144Mhz

ACODEC_ADCNGLMTCFG
Address: Operational Base + offset (0x01a4)
ADC limiter noise gate configure register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	NGCHL_LI 0: individual channel (or); 1: both channel (and);
6	RW	0x0	NGEN_LI 0: Noise gate Disable; 1: Noise gate enable;
5	RW	0x0	NGBOOST_LI 0: Normal noise gate; 1: Boost noise gate;
4:2	RW	0x0	NGGATE_LI NGBOOST = 0: 000~111(-63~-84,3db/step) NGBOOST = 1: 000~111(-33~-54,3db/step)
1:0	RW	0x0	NGDLY_LI the delay time before the noise gate attacks, unit: (clk1x * 8) 00: 2048 01: 4096 10: 8192 11: 16384

ACODEC_ADCNGLMTST
Address: Operational Base + offset (0x01a8)
ADC limiter noise gate status

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	NGACT_LI noise gate active status 1: noise gate is active 0: noise gate is not active

ACODEC_HPLOCFG0
Address: Operational Base + offset (0x01c0)
headphone and line-out configure register0

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3:2	RW	0x0	INC_LO_AMP increase the LO amplitude from 3dB to 9dB 00: 0db 01: 3db 10: 6db 11: 9db

1	RW	0x0	LO_VAG_RISE_SLOW for Line-OUT antipop process, slow down the VAG rising speed. 1'b1: enable 1'b0: disable
0	RW	0x0	LO_OUT_VAG make the Lineout voltage stable at VAG voltage. 1'b1: enable 1'b0: disable

ACODEC_HPLOCFG1
Address: Operational Base + offset (0x01c4)
headphone and line-out configure register1

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:3	RW	0x0	INC_HP_AMP increase the HP amplitude from 3dB to 9dB 00: 0db 01: 3db 10: 6db 11: 9db
2	RW	0x0	HP_TWO_STAGE make the HP as two stage opamp 1'b1: enable 1'b0: disable
1	RW	0x0	INC_OC_RANGE increase the HP over-current threshold 1'b1: enable 1'b0: disable
0	RW	0x0	HP_OPAMP_HALF_BIAS make the total HP ibias to 50% 1'b1: enable 1'b0: disable

ACODEC_HPLOCFG2
Address: Operational Base + offset (0x01c8)
(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

4:3	RW	0x0	INC_HP_AMP increase the HP amplitude from 3dB to 9dB 00: 0db 01: 3db 10: 6db 11: 9db
2	RW	0x0	HP_TWO_STAGE make the HP as two stage opamp 1'b1: enable 1'b0: disable
1	RW	0x0	INC_OC_RANGE increase the HP over-current threshold 1'b1: enable 1'b0: disable
0	RW	0x0	HP_OPAMP_HALF_BIAS make the total HP ibias to 50% 1'b1: enable 1'b0: disable

ACODEC_HPLOCFG3
Address: Operational Base + offset (0x01cc)
(debug only, maintain default value)

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x1	PWD_HP_OSTG power down the HP L and R channel output stage 1'b1: power down 1'b0: power on
2	RW	0x1	PWD_HP_VGND power down the HP VGND opamp 1'b1: power down 1'b0: power on
1	RW	0x1	PWD_HP_BUF power down the HP L and R channel pre-amp 1'b1: power down 1'b0: power on
0	RW	0x1	HP_SHORT_OUT connect the HP L and R output to VGND 1'b1: enable 1'b0: disable

ACODEC_HPLOCFG4
Address: Operational Base + offset (0x01d0)
headphone and line-out configure register4

Bit	Attr	Reset Value	Description

31:5	RO	0x0	reserved
4	RW	0x0	HP_ANTIPOP_EN enable the HP antipop function 1'b1: enable 1'b0: disable
3:0	RW	0x0	HP_ANTIPOP_BIT control the HP antipop gain from -16dB to 0dB the HP internal gain 0000-> 0dB 0001-> -1dB 0010-> -2dB 0011-> -3dB 0100-> -4dB 0101-> -5dB 0110-> -6dB 0111-> -7dB 1000-> -8dB 1001-> -9dB 1010-> -10dB 1011-> -11dB 1100-> -12dB 1101-> -13dB 1110-> -14dB 1111-> -15dB.

ACODEC_HPLOCFG5

Address: Operational Base + offset (0x01d4)

headphone and line-out configure register5

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	PWD_LO_OSTG Line out output stage power down 1'b1: power down 1'b0: power on
5	RW	0x0	PWD_LO_BUF power down the Line-out pre amp stage 1'b1: power down 1'b0: power on
4	RW	0x0	LO_ANTIPOP_EN enable the Line-out antipop function 1'b1: enable 1'b0: disable
3:0	RW	0x0	LO_ANTIPOP_BIT control the Line-out antipop function, gain change from -16dB to 0dB

ACODEC_PLLCFG0
Address: Operational Base + offset (0x0200)
PLL configure register0

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:3	RW	0x2	PLL_CLKIN_SEL PLL clock input selector PLL clock input selector 2'b00: main clock 2'b01: main clock/2 2'b10: system clock(24Mhz for NANOD) 2'b11: I2S SCLK
2	RW	0x1	PLL_OUTDIV_EN PLL VCO output clock divider enable PLL VCO output clock divider enable 1'b1: enable 1'b0: disable
1:0	RW	0x0	PLL_VCO_BANDSEL PLL VCO working band select (debug only, maintain default value)

ACODEC_PLLCFG1
Address: Operational Base + offset (0x0204)
PLL configure register1

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RW	0x0	PLL_RES_SEL PLL filter resistor value selector PLL filter resistor value selector
5:3	RW	0x0	PLL_CUR_SEL PLL current selector PLL current selector
2:0	RW	0x0	PLL_POSDIV_L3 lowest 3 bits of PLL_POSDIV lowest 3 bits of PLL_POSDIV

ACODEC_PLLCFG2
Address: Operational Base + offset (0x0208)
PLL configure register2

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x30	PLL_POSDIV_H8 the highest 8 bits of PLL_POSDIV the highest 8 bits of PLL_POSDIV

ACODEC_PLLCFG3

Address: Operational Base + offset (0x020c)
PLL configure register3

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x19	PLL_PREDIV PLL pre-divider PLL pre-divider

ACODEC_PLLCFG4
Address: Operational Base + offset (0x0210)
PLL configure register4

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:4	RW	0x6	PLL_OUTDIV PLL VCO output clock divider
3:0	RW	0x5	PLL_CLK_DIV PLL_HIGH_CLK divider PLL_HIGH_CLK divider

ACODEC_PLLCFG5
Address: Operational Base + offset (0x0214)
PLL configure register5

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	PLL_RESET PLL reset reset the PLL, but do not clear the registers 1'b1: set reset 1'b0: release reset
1	RW	0x0	PLL_TEST PLL test check the PLL internal VCO control voltage
0	RW	0x0	PLL_PWD PLL power down 1'b0: PLL power on 1'b1: PLL power down

ACODEC_I2SCKM
Address: Operational Base + offset (0x0240)
I2S clock mode

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:4	RW	0x0	MCK_DIV I2S mclk divider I2S mclk divider, active only in I2S master mode for sclk generation. SCK_DIV = F(pll_clk_div)/F(sclk) - 1 note: pll_clk_div is 12.288Mhz if sample rate is 12Khz/24Khz/48Khz/96Khz/192Khz; 8.192Mhz if sample rate is 8Khz/16Khz/32Khz/64Khz/128Khz; otherwise pll_clk_div is 11.2896Mhz
3	RO	0x0	reserved
2	RW	0x0	SCK_EN I2S sclk enable I2S sclk enable, active only in I2S master mode 1'b1: enable 1'b0: disable
1	RW	0x0	SCK_P sclk polarity sclk polarity 1'b1: enable 1'b0: disable
0	RW	0x0	I2S_MST I2S master mode 1'b0: I2S slave mode 1'b1: I2S master mode

ACODEC_I2SRXCR0
Address: Operational Base + offset (0x0244)
I2S Rx control register0

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2:1	RW	0x0	SCKD_RX I2S Rx sclk divider sclk divider for rx lrck generation 2'b00: 64 2'b01: 128 2'b10: 256 2'b11: reserved
0	RW	0x0	RXRL_P I2S Rx lrck polarity I2S Rx lrck polarity 1'b0: normal 1'b1: inverted

ACODEC_I2SRXCR1
Address: Operational Base + offset (0x0248)
I2S Rx control register1

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	TFS_RX Rx transfer mode selector Rx transfer mode selector: 1'b0: I2S 1'b1: PCM
5:4	RW	0x0	PBM_RX PCM Rx bus mode PCM Rx bus mode 2'b00: delay0 2'b01: delay1 2'b10: delay2 2'b11: delay3
3:2	RW	0x0	IBM_RX I2S Rx bus mode I2S Rx bus mode 2'b00: normal 2'b01: left 2'b10: right
1	RW	0x0	EXRL_RX exchange right/left channel for Rx exchange right/left channel for Rx 1'b0: normal 1'b1: exchange right and left channel
0	RW	0x0	LSB_RX I2S sdi endian 1'b0: LSB 1'b1: MSB

ACODEC_I2SRXCR2
Address: Operational Base + offset (0x024c)
I2S Rx control register2

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x17	VDW_RX valid data width for Rx valid data width for Rx 5'h17: 24 bits 5'h0f: 16 bits

ACODEC_I2SRXCMD
Address: Operational Base + offset (0x0250)
I2S Rx command

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	RXS Rx transfer start Rx transfer start 1'b1: Rx start 1'b0: Rx stop
0	RW	0x0	RXC rx transfer clear rx transfer clear, high active.

ACODEC_I2STXCR0
Address: Operational Base + offset (0x0260)
I2S Tx configure register0

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	SCKD_TX I2S Tx sclk divider sclk divider for tx lrck generation 2'b00: 64 2'b01: 128 2'b10: 256
0	RW	0x0	TXRL_P I2S Tx lrck polarity I2S Tx lrck polarity 1'b0: normal 1'b1: inverted

ACODEC_I2STXCR1
Address: Operational Base + offset (0x0264)
I2S Tx configure register1

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	TFS_TX Tx transfer mode selector Tx transfer mode selector: 1'b0: I2S 1'b1: PCM
5:4	RW	0x0	PBM_TX PCM Tx bus mode PCM Tx bus mode 2'b00: delay0 2'b01: delay1 2'b10: delay2

Bit	Attr	Reset Value	Description
3:2	RW	0x0	IBM_TX I2S Tx bus mode I2S Tx bus mode 2'b00: normal 2'b01: left 2'b10: right
1	RW	0x0	EXRL_TX exchange right/left channel for Tx exchange right/left channel for Tx 1'b0: normal 1'b1: exchange right and left channel
0	RW	0x0	LSB_TX I2S sdo endian 1'b0: LSB 1'b1: MSB

ACODEC_I2STXCR2
Address: Operational Base + offset (0x0268)
I2S Tx configure register2

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x17	VDW_TX valid data width for Tx valid data width for Tx 5'h17: 24 bits 5'h0f: 16 bits

ACODEC_I2STXCR3
Address: Operational Base + offset (0x026c)
I2S Tx configure register3

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RW	0x00	RCNT_TX Tx right justified counter right justified counter for I2S right justified slave mode

ACODEC_I2STXCMD
Address: Operational Base + offset (0x0270)
I2S Tx command

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved

Bit	Attr	Reset Value	Description
1	RW	0x0	TXS Tx transfer start Tx transfer start 1'b1: Rx start 1'b0: Rx stop
0	RW	0x0	TXC tx transfer clear tx transfer clear, high active.

ACODEC_TMCFG0
Address: Operational Base + offset (0x0300)
test mode configure register

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	ATE_TEST_MODE ate test mode 1'b1: enable 1'b0: disable
0	RW	0x0	BURN_IN_MODE burn in mode 1'b1: enable 1'b0: disable

Chapter 10 SDMMC& SDIO

10.1 Overview

The SDMMC Host Controller is designed to support Secure Digital memory (SD mem-version 2.00), Secure Digital I/O(SDIO-version 2.00), Multimedia Cards(MMC-version 4.41).The SDMMC support SD Card(1/4bit), SDIO, eMMC(1/4bit).

10.2 Features

- Supports AMBA AHB interface
- Supports DMA controller for data transfers
- Supports interrupt output
- Supports SD version2.0 except SPI mode
- Supports MMC version4.41 except SPI mode
- Supports SDIO version2.0
- 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

10.3 Architecture

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

10.3.1 Block Diagram

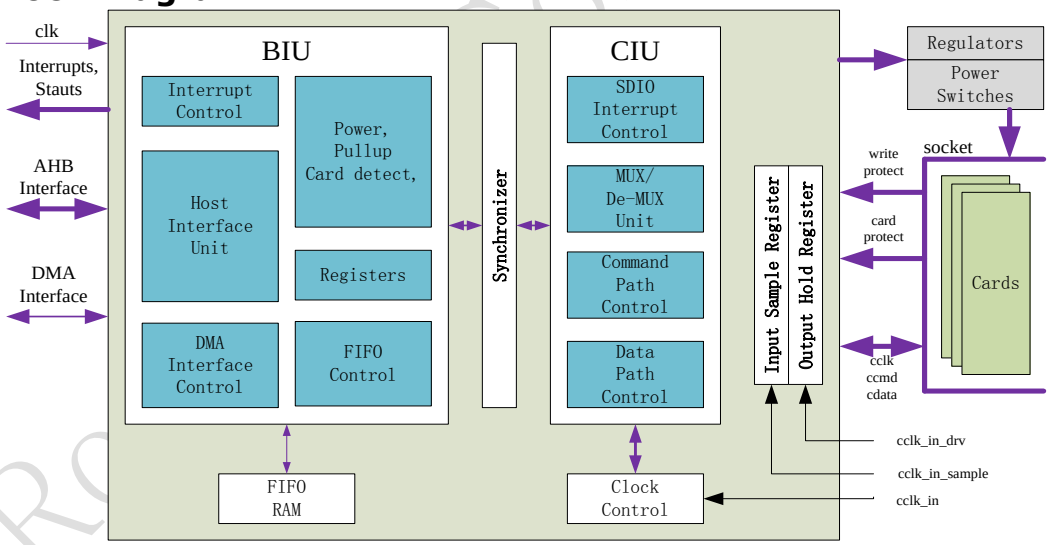


Fig 10-1 SD/MMC Controller Interface Signals

10.3.2 Block Descriptions

- 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.

10.4 Registers

10.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description

Name	Offset	Size	Reset Value	Description
SDMMC_CTRL	0x0000	W	0x00000000	Control register
SDMMC_PWREN	0x0004	W	0x00000000	Power-enable register
SDMMC_CLKDIV	0x0008	W	0x00000000	Clock-divider register
SDMMC_CLKENA	0x0010	W	0x00000000	Clock-enable register
SDMMC_TMOUT	0x0014	W	0xffffff40	Time-out register
SDMMC_CTYPE	0x0018	W	0x00000000	Card-type register
SDMMC_BLKSIZ	0x001c	W	0x00000200	Block-size register
SDMMC_BYTCNT	0x0020	W	0x00000200	Byte-count register
SDMMC_INTMASK	0x0024	W	0x00000000	Interrupt-mask register
SDMMC_CMDARG	0x0028	W	0x00000000	Command-argument register
SDMMC_CMD	0x002c	W	0x00000000	Command register
SDMMC_RESP0	0x0030	W	0x00000000	Response-0 register
SDMMC_RESP1	0x0034	W	0x00000000	Response-1 register
SDMMC_RESP2	0x0038	W	0x00000000	Response-2 register
SDMMC_RESP3	0x003c	W	0x00000000	Response-3 register
SDMMC_MINTSTS	0x0040	W	0x00000000	Masked interrupt-status register
SDMMC_RINTSTS	0x0044	W	0x00000000	Raw interrupt-status register
SDMMC_STATUS	0x0048	W	0x00000406	Status register
SDMMC_FIFOTH	0x004c	W	0x00000000	FIFO threshold register
SDMMC_CDETECT	0x0050	W	0x00000000	Card-detect register
SDMMC_WRTPRT	0x0054	W	0x00000000	Write-protect register
SDMMC_TCBCNT	0x005c	W	0x00000000	Transferred CIU card byte count
SDMMC_TBBCNT	0x0060	W	0x00000000	Transferred host/DMA to/from BIU-FIFO byte count
SDMMC_DEBNCE	0x0064	W	0x00ffffff	Card detect debounce register
SDMMC_USRID	0x0068	W	0x00000000	User ID register
SDMMC_RST_n	0x0078	W	0x00000001	Hardware reset register
SDMMC_BACK_END_POWER	0x0104	W	0x00000000	Back-end Power
SDMMC_FIFO_BASE	0x0200	W	0x00000000	

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

10.4.2 Detail Register Description

SDMMC_CTRL

Address: Operational Base + offset (0x0000)

Control register

Bit	Attr	Reset Value	Description
31:9	RO	0x0	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.

Bit	Attr	Reset Value	Description
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(s). In meantime, if host wants SDMMC Controller to exit waiting for interrupt state, it can set this bit, at which time SDMMC 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 For sending read-wait to SDIO cards
5	RW	0x0	dma_enable 0 –Disable DMA transfer mode 1 –Enable DMA transfer mode Even when DMA mode is enabled, host can still push/pop data into or from FIFO; this should not happen during the normal operation. If there is simultaneous FIFO access from host/DMA, the data coherency is lost. Also, there is no arbitration inside SDMMC Controller to prioritize simultaneous host/DMA access.
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	RO	0x0	reserved
2	W1C	0x0	dma_reset 0 –No change 1 –Reset internal DMA interface control logic To reset DMA interface, firmware should set bit to 1. This bit is auto-cleared after two AHB clocks.
1	W1C	0x0	fifo_reset 0 –No change 1 –Reset to 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

Bit	Attr	Reset Value	Description
0	W1C	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 Does not affect any registers or DMA interface, or FIFO or host interrupts

SDMMC_PWREN
Address: Operational Base + offset (0x0004)
Power-enable register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	power_enable Power on/off switch for the card. 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 Bit values output to card_power_en port.

SDMMC_CLKDIV
Address: Operational Base + offset (0x0008)
Clock-divider register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	clk_divider0 Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2*0 = 0 (no division, bypass), value of 1 means divide by 2*1 = 2

SDMMC_CLKENA
Address: Operational Base + offset (0x0010)
Clock-enable register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved

Bit	Attr	Reset Value	Description
16	RW	0x0	cclk_low_power Low-power control for SD card clock and MMC card clock supported. 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	RO	0x0	reserved
0	RW	0x0	cclk_enable Clock-enable control for SD card clock and MMC card clock supported. 0 –Clock disabled 1 –Clock enabled

SDMMC_TMOUT
Address: Operational Base + offset (0x0014)
Time-out register

Bit	Attr	Reset Value	Description
31:8	RW	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	RW	0x40	response_timeout Response timeout value. Value is in number of card output clocks –cclk_out.

SDMMC_CTYPE
Address: Operational Base + offset (0x0018)
Card-type register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RW	0x0	card_width_8 Indicates if card is 8-bit: 0 –Non 8-bit mode 1 –8-bit mode
15:1	RO	0x0	reserved
0	RW	0x0	card_width Indicates if card is 1-bit or 4-bit: 0 –1-bit mode 1 –4-bit mode

SDMMC_BLKSIZ
Address: Operational Base + offset (0x001c)
Block-size register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0200	block_size Block size

SDMMC_BYTCNT
Address: Operational Base + offset (0x0020)
Byte-count register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000200	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 (0x0024)
Interrupt-mask register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RO	0x0	sdio_int_mask Mask SDIO interrupts When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enables an interrupt.
15:0	RW	0x0000	int_mask Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt. bit 15 –End-bit error (read)/Write no CRC (EBE) bit 14 –Auto command done (ACD) bit 13 –Start-bit error (SBE) bit 12 –Hardware locked write error (HLE) bit 11 –FIFO underrun/overrun error (FRUN) bit 10 –Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9 –Data read timeout (DRTO) bit 8 –Response timeout (RTO) bit 7 –Data CRC error (DCRC) bit 6 –Response CRC error (RCRC) bit 5 –Receive FIFO data request (RXDR) bit 4 –Transmit FIFO data request (TXDR) bit 3 –Data transfer over (DTO) bit 2 –Command done (CD) bit 1 –Response error (RE) bit 0 –Card detect (CD)

SDMMC_CMDARG
Address: Operational Base + offset (0x0028)
Command-argument register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	cmd_arg Value indicates command argument to be passed to card.

SDMMC_CMD
Address: Operational Base + offset (0x002c)
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. Once command is sent and response is received from SD_MMC cards, Command Done bit is set in raw interrupt register.
30	RO	0x0	reserved
29	RW	0x0	use_hold_reg Use Hold Register 0 - CMD and DATA sent to card bypassing HOLD Register 1 - CMD and DATA sent to card through the HOLD Register Note: a. Set to 1'b1 for SDR12 and SDR25 (with non-zero phase-shifted cclk_in_drv); zero phase shift is not allowed in these modes. b. Set to 1'b0 for SDR50(with zero phase- shifted cclk_in_drv) c. Set to 1'b1 for SDR50 (with non-zero phase-shifted cclk_in_drv)
28	RW	0x0	volt_switch Voltage switch bit 0 - No voltage switching 1 - Voltage switching enabled; must be set for CMD11 only
27	RW	0x0	boot_mode Boot Mode 0 - Mandatory Boot operation 1 - Alternate Boot operation

Bit	Attr	Reset Value	Description
26	RW	0x0	disable_boot Disable Boot. When software sets this bit along with start_cmd, CIU terminates the boot operation. Do NOT set disable_boot and enable_boot together.
25	RW	0x0	expect_boot_ack Expect Boot Acknowledge. When Software sets this bit along with enable_boot, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.
24	RW	0x0	enable_boot Enable Boot—this bit should be set only for mandatory boot mode. When Software sets this bit along with start_cmd, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set disable_boot and enable_boot together.
23:22	RO	0x0	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, CLRSRC, and CLKENA. Changes card clocks (change frequency, truncate 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 is no Command Done interrupts because no command is sent to SD_MMC cards.
20:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
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 so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).
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. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = disable_boot.
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.

Bit	Attr	Reset Value	Description
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 SD_MMC cards at end of data transfer. * when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands * open-ended transfers that software should explicitly send to stop command Additionally, when "resume" is sent to resume –suspended memory access of SD-Combo card –bit should be set correctly if suspended data transfer needs send_auto_stop. Don't care if no data expected from card.
11	RW	0x0	transfer_mode 0 –Block data transfer command 1 –Stream data transfer command Don't care if no data expected.
10	RW	0x0	wr 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	0x00	cmd_index Command index

SDMMC_RESP0

Address: Operational Base + offset (0x0030)

Response-0 register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response0 Bit[31:0] of response

SDMMC_RESP1
Address: Operational Base + offset (0x0034)
Response-1 register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response 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.

SDMMC_RESP2
Address: Operational Base + offset (0x0038)
Response-2 register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response2 Bit[95:64] of long response

SDMMC_RESP3
Address: Operational Base + offset (0x003c)
Response-3 register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response3 Bit[127:96] of long response

SDMMC_MINTSTS
Address: Operational Base + offset (0x0040)
Masked interrupt-status register

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RO	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 enables interrupt; 0 masks interrupt). 0 –No SDIO interrupt from card 1 –SDIO interrupt from card
23:17	RO	0x0	reserved
16	RW	0x0	new_int_status New Interrupt Status 0: data_no busy int

Bit	Attr	Reset Value	Description
15:0	RO	0x0000	int_status Interrupt enabled only if corresponding bit in interrupt mask register is set. bit 15 –End-bit error (read)/write no CRC (EBE) bit 14 –Auto command done (ACD) bit 13 –Start-bit error (SBE) bit 12 –Hardware locked write error (HLE) bit 11 –FIFO underrun/overrun error (FRUN) bit 10 –Data starvation by host timeout (HTO)/Volt_switch_int bit 9 –Data read timeout (DRTO) bit 8 –Response timeout (RTO) bit 7 –Data CRC error (DCRC) bit 6 –Response CRC error (RCRC) bit 5 –Receive FIFO data request (RXDR) bit 4 –Transmit FIFO data request (TXDR) bit 3 –Data transfer over (DTO) bit 2 –Command done (CD) bit 1 –Response error (RE) bit 0 –Card detect (CD)

SDMMC_RINTSTS

Address: Operational Base + offset (0x0044)

Raw interrupt-status register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RO	0x0	sdio_interrupt Interrupt from SDIO card; Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 0 –No SDIO interrupt from card 1 –SDIO interrupt from card

Bit	Attr	Reset Value	Description
15:0	RO	0x0000	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. bit 15 –End-bit error (read)/write no CRC (EBE) bit 14 –Auto command done (ACD) bit 13 –Start-bit error (SBE) bit 12 –Hardware locked write error (HLE) bit 11 –FIFO underrun/overrun error (FRUN) bit 10 –Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9 –Data read timeout (DRTO)/Boot Data Start (BDS) bit 8 –Response timeout (RTO)/Boot Ack Received (BAR) bit 7 –Data CRC error (DCRC) bit 6 –Response CRC error (RCRC) bit 5 –Receive FIFO data request (RXDR) bit 4 –Transmit FIFO data request (TXDR) bit 3 –Data transfer over (DTO) bit 2 –Command done (CD) bit 1 –Response error (RE) bit 0 –Card detect (CD)

SDMMC_STATUS
Address: Operational Base + offset (0x0048)
Status register

Bit	Attr	Reset Value	Description
31	RO	0x0	dma_req DMA request signal state
30	RO	0x0	dma_ack DMA acknowledge signal state
29:17	RO	0x0000	fifo_count FIFO count –Number of filled locations in FIFO
16:11	RO	0x00	response_index Index of previous response, including any auto-stop sent by core
10	RO	0x1	data_state_mc_busy Data transmit or receive state-machine is busy
9	RO	0x0	data_busy Inverted version of raw selected card_data[0] 0 –card data not busy 1 –card data busy default value is 1 or 0 depending on cdata_in

Bit	Attr	Reset Value	Description
8	RO	0x0	data_3_status Raw selected card_data[3]; checks whether card is present 0 –card not present 1 –card present default value is 1 or 0 depending on cdata_in
7:4	RO	0x0	command_fsm_states 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 NOTE: The command FSM state is represented using 19 bits. The STATUS Register(7:4) has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the STATUS(7:4) register. The three states that are not represented in the STATUS Register(7:4) are: * Bit 16 –Wait for CCS * Bit 17 –Send CCSD * Bit 18 –Boot Mode Due to this, while command FSM is in "Wait for CCS state" or "Send CCSD" or "Boot Mode", the Status register indicates status as 0 for the bit field 7:4.
3	RO	0x0	fifo_full FIFO is full status
2	RO	0x1	fifo_empty FIFO is empty status
1	RO	0x1	fifo_tx_watermark FIFO reached Transmit watermark level; not qualified with data transfer

Bit	Attr	Reset Value	Description
0	RO	0x0	fifo_rx_watermark FIFO reached Receive watermark level; not qualified with data transfer

SDMMC_FIFOTH
Address: Operational Base + offset (0x004c)
FIFO threshold register

Bit	Attr	Reset Value	Description
31	RO	0x0	reserved
30:28	RW	0x0	DMA_Mutiple_Transaction_Size Burst size of multiple transaction; should be programmed same as 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 The units for transfers is the H_DATA_WIDTH parameter. A single transfer would be signaled based on this value. Value should be sub-multiple of (RX_WMark + 1)* (F_DATA_WIDTH/H_DATA_WIDTH) and (FIFO_DEPTH - TX_WMark)* (F_DATA_WIDTH/ H_DATA_WIDTH) For example, if FIFO_DEPTH = 16, FDATA_WIDTH == H_DATA_WIDTH Allowed combinations for MSize and TX_WMark are: MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMark = 8 MSize = 4, TX_WMark = 4 MSize = 4, TX_WMark = 12 MSize = 8, TX_WMark = 8 MSize = 8, TX_WMark = 4 Allowed combinations for MSize and RX_WMark are: MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMark = 3 MSize = 4, RX_WMark = 7 MSize = 4, RX_WMark = 11 MSize = 8, RX_WMark = 7 Recommended: MSize = 8, TX_WMark = 8, RX_WMark = 7

Bit	Attr	Reset Value	Description
27:16	RW	0x000	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 enabled, 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. 12 bits –1 bit less than FIFO-count of status register, which is 13 bits. Limitation: RX_WMark <= FIFO_DEPTH-2 Recommended: (FIFO_DEPTH/2) - 1; (means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS timeout.
15:12	RO	0x0	reserved

Bit	Attr	Reset Value	Description
11:0	RW	0x000	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 enabled, 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. 12 bits –1 bit less than FIFO-count of status register, which is 13 bits. Limitation: TX_WMark >= 1; Recommended: FIFO_DEPTH/2; (means less than or equal to FIFO_DEPTH/2)

SDMMC_CDETECT
Address: Operational Base + offset (0x0050)
Card-detect register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	card_detect_n Value on card_detect_n input ports; read-only bits. 0 represents presence of card.

SDMMC_WRTPRT
Address: Operational Base + offset (0x0054)
Write-protect register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	write_protect Value on card_write_prt input port. 1 represents write protection.

SDMMC_TCBCNT
Address: Operational Base + offset (0x005c)
Transferred CIU card byte count

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	trans_card_byte_count Number of bytes transferred by CIU unit to card. In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied. Both TCBCNT and TBBCNT share same coherency register. When AREA_OPTIMIZED parameter is 1, register should be read only after data transfer completes; during data transfer, register returns 0.

SDMMC_TBBCNT
Address: Operational Base + offset (0x0060)
Transferred host/DMA to/from BIU-FIFO byte count

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	trans_fifo_byte_count Number of bytes transferred between Host/DMA memory and BIU FIFO. In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied. Both TCBCNT and TBBCNT share same coherency register.

SDMMC_DEBNCE
Address: Operational Base + offset (0x0064)
Card detect debounce register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:0	RW	0xffffff	debounce_count Number of host clocks (clk) used by debounce filter logic; typical debounce time is 5-25 ms.

SDMMC_USRID
Address: Operational Base + offset (0x0068)
User ID register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	USRID User identification register; value set by user. Default reset value can be picked by user while configuring core before synthesis. Can also be used as scratch pad register by user. the default value is determined by Configuration Value.

SDMMC_RST_n
Address: Operational Base + offset (0x0078)
Hardware reset register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x1	CARD_RESET Hardware reset. 1 –Active mode 0 –Reset These bits cause the cards to enter pre-idle state, which requires them to be re-initialized. CARD_RESET[0] should be set to 1'b1 to reset card

SDMMC_BACK_END_POWER
Address: Operational Base + offset (0x0104)
Back-end Power

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	Back_End_Power Back end power 1'b0 –Off; Reset 1'b1 –Back-end Power supplied to card application

SDMMC_FIFO_BASE
Address: Operational Base + offset (0x0200)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	fifo_base_addr fifo base addr

10.5 Application Notes

10.5.1 Software/Hardware Restriction

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.

If the card is enumerated in SDR12 or SDR25 mode, the application must program the use_hold_reg bit[29] in the CMD register to 1'b1.

This programming should be done for all data transfer commands and non-data commands that are sent to the card. When the use_hold_reg bit is programmed to 1'b0, the SD/MMC Controller bypasses the Hold Registers in the transmit path. The value of this bit should not be changed when a Command or Data Transfer is in progress. For more details on

using use_hold_reg and the implementation requirements for meeting the Card input hold time, refer to "Recommended Usage" and Table 10-1.

Table 10-1Recommended Usage of use_hold_reg

No.	Speed Mode	use_hold_reg	cclk_in	clk_in_drv	clk_divider
1	SDR25	1'b1	50	50	0
2	SDR12	1'b1	50	50	1

To avoid glitches in the card clock outputs (sdmmc_clkout), 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.

10.5.2 Programming Sequence

Initialization

Fig. 10-2 illustrates the initialization flow.

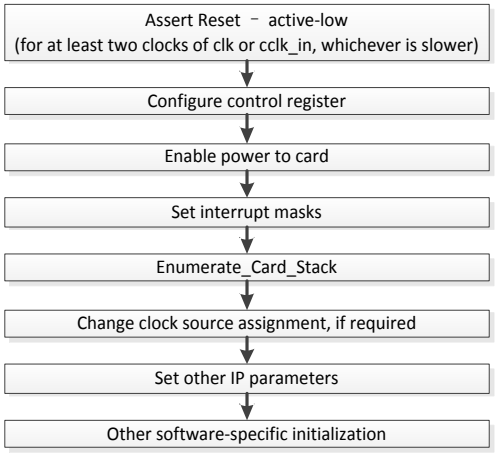


Fig 10-2 Initialization Sequence

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 pull-up 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 sdmmc_clkout according to SDMMC specifications.
   - Response_timeout = 0x64
   - Data_timeout = 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

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 disabled while switching off the power.

Clock Programming

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_cmdis 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 along response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3register 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 10-2.

Table 10-2Command Settings for No-Data Command

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used;ref to "use_hold_reg" on CMD register
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 Host 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 11-3. 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 10-3Command Register Setting for Single-Block or Multiple-Block Read

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
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 11-4. 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 10-4Command Register Settings for Single-Block or Multiple-Block Write

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
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	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, 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 Host 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 theAuto_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 ASx 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. 10-3:

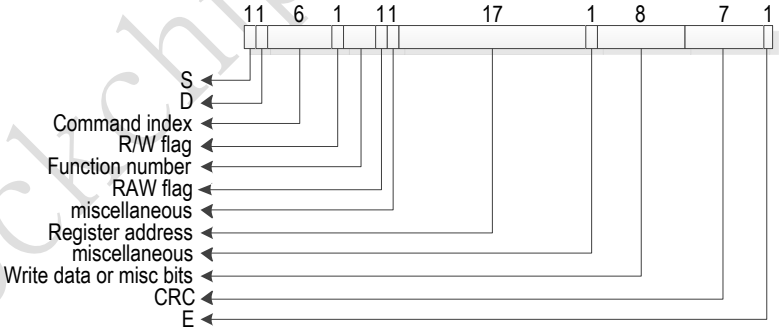


Fig 10-3 Command format for CMD52

a. Program the CMDARG register @0x28 with the appropriate command argument parameters listed in Table 10-5.

Table 10-5Parameters 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 CMD52command.

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 FSx 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 FSx 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 10-6.

Table 10-6CMDARG 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 Host 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 Host 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 DMA controller that are interfaced to the SD/MMC Host 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.

Notes: 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.

10.5.3 Programming SD/MMC Controller for Boot Operation

Boot Operation

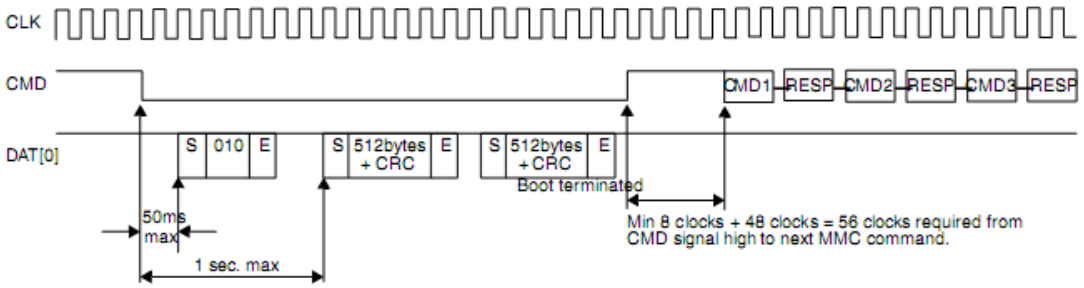


Fig 10-4 illustrates timing for Boot operation

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 following:

- Registers
- Ports
- FIFO-pointers
- DMA interface controls
- State-machines in the design

After power-on reset, the software should perform the appropriate steps described in the following sections for the respective types of cards.

Following are the steps that the software driver must follow when working with eMMC cards for Boot operation.

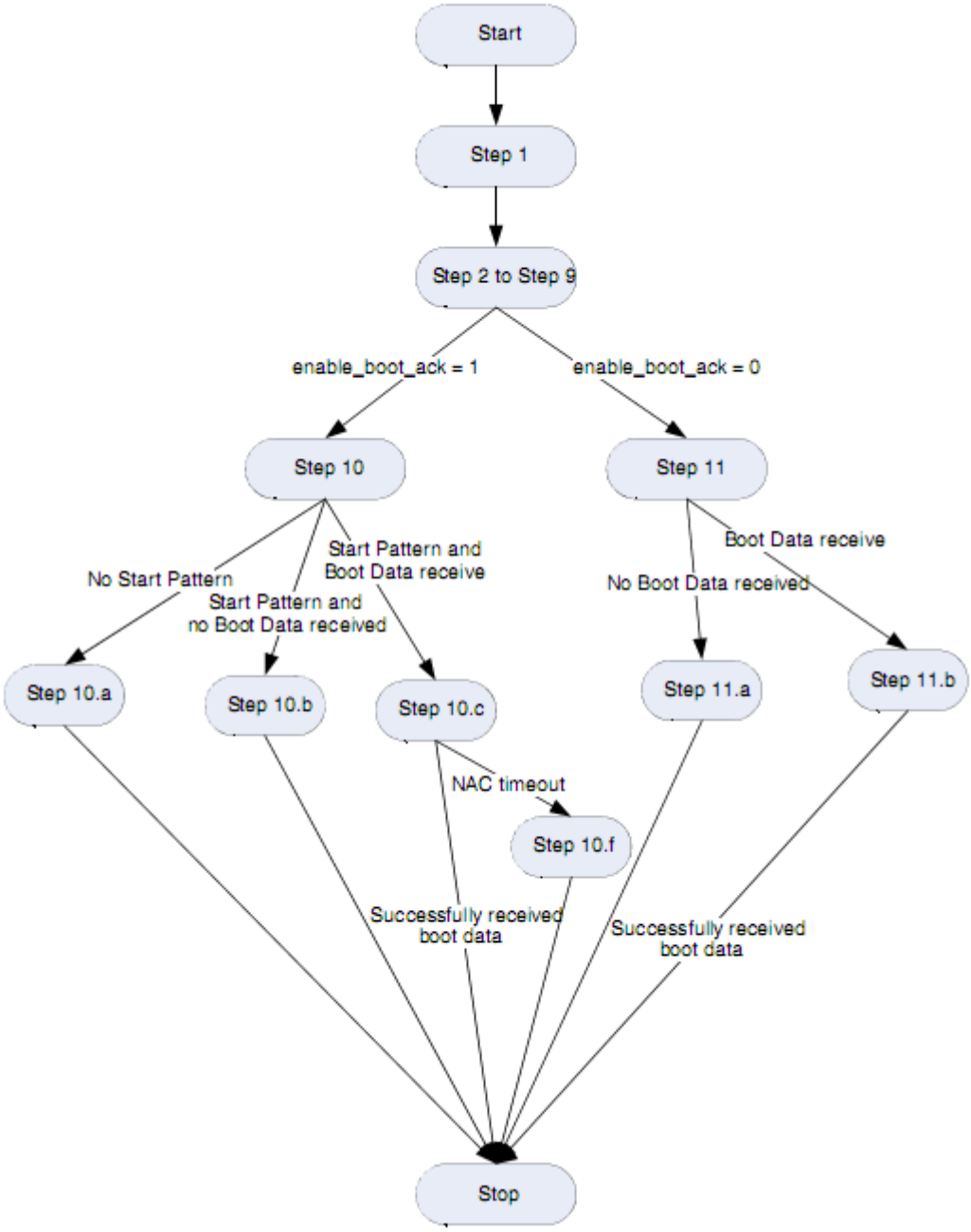


Fig 10-5 SD/MMC Controller Flow for Boot Operation

1. The software driver is aware:
   - That the card supports boot operation—BOOT_PARTITION_ENABLE bit set in the card.
   - Of the BOOT_SIZE_MULT value in the card and the data bus width to use during boot operation—Extend CSD register byte[177] bit[0:1].
2. Set the following:
   - Masks for interrupts by clearing appropriate bits in the Interrupt Mask register @0x024.
   - Global int_enable bit of the Control register @0x00.

   It is recommended that you write 0xffff_ffff to the Raw Interrupt register @0x044 and IDSTS @0x8C in order to clear any pending interrupts before setting the int_enable bit.

   For Internal DMAC mode, the software driver needs to unmask all the relevant fields in the IDINTEN register.

3. Configure control register (CTRL):
   - int_enable = 1'b1
   - Other fields should be 1'b0.
4. Change clock source assignment – Set the card frequency to 400 KHz using the clock-divider and clock-source registers; for details, refer to "Clock Programming".
5. Set Data_timeout = (10 * ((TAAC * Fop) + (100 * NSAC)); this is NAC.
6. Program the BLKSIZ register with 0x200 (512 bytes).
7. Program the BYTCNT register with multiples of 128K bytes, as indicated by the BOOT_SIZE_MULT value in the card.
8. Program the Rx FIFO threshold value in bytes in the FIFOTH register @0x04C.Typically, the threshold value can be set to half the FIFO depth; that is, RX_WMark = (FIFO_DEPTH/2) - 1.
9. Program the CMD register with the following fields:
   - start_cmd = 1'b1
   - enable_boot = 1'b1
   - enable_boot_ack – depends on whether a start-acknowledge pattern is expected from the card
   - Card_number = appropriate_card_number; obtained by referring to CDETECT register
   - Data_expected = 1'b1
   - Remainder of CMD register fields = 1'b0
10. If enable_boot_ack = 1'b1, the software driver should start a timer after step #9; the terminal value is 50ms.
    - Before this timer elapses, the BAR interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver must program the CMD register with the following fields:
      - start_cmd = 1'b1
      - disable_boot = 1'b1
      - All other fields = 0

      The SD/MMC Controller generates a Command Done (CD) interrupt after de-asserting the CMD line of the card.
    - If the BAR interrupt is received, the software driver should clear this interrupt by writing a 1 to it. The software driver should then start another timer with a terminal value of 1 - 0.05 = 0.95 seconds. Before this timer elapses, the BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver must program the CMD register with the following fields:
      - start_cmd = 1'b1
      - disable_boot = 1'b1
      - All other fields = 0

      The SD/MMC Controller generates a Command Done (CD) interrupt after de-asserting the CMD line of the card.
    - If the BDS interrupt is received, it indicates that the boot data is being received from the card. The software driver can then initiate a data read from the SD/MMC Controller based on the RXDR interrupt bit in the RINTSTS register. At the end of a successful boot data transfer from the card, the following interrupts are generated:
      - Command Done (CD) in RINTSTS register

- Data Transfer Over (DTO) in RINTSTS register
- If an Error occurs in Boot Ack pattern (010) or an end bit Error occurs:
  - RTL automatically aborts boot by pulling CMD line high
  - RTL generates Command done interrupt
  - RTL does not generate BAR interrupt
  - Application aborts boot transfer
- If between data block transfers NAC is violated, DRTO (Data Read Timeout) is asserted. Apart from this, if there are errors associated with Start/End bits, SBE/EBE interrupts are also generated.

11. If enable_boot_ack = 1'b0, the software driver should start a timer after the step #9 where the terminal value is 1 second.
    - Before this timer elapses, a BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver must program the CMD register with the following fields:
      - start_cmd = 1'b1
      - disable_boot = 1'b1
      - All other fields = 0

      The SD/MMC Controller generates a Command Done (CD) interrupt after de-asserting the CMD line of the card.
    - If a BDS interrupt is received, it indicates that the boot data is being received from the card. The software driver can then initiate a data read from the SD/MMC Controller based on the RXDR interrupt bit in the RINTSTS register.

      At the end of a successful boot data transfer from card, the following interrupts are generated.
      - Command Done (CD) in RINTSTS.
      - Data Transfer Over (DTO) in RINTSTS.

Alternative Boot Operation

The Alternative Boot Operation differs from the Boot Operation in that CMD0 is used to boot the card rather than holding down the CMD-line of the card. The Alternative Boot Operation can be done only if bit 0 in the extended CSD byte[228] (BOOT_INFO) is set to 1.

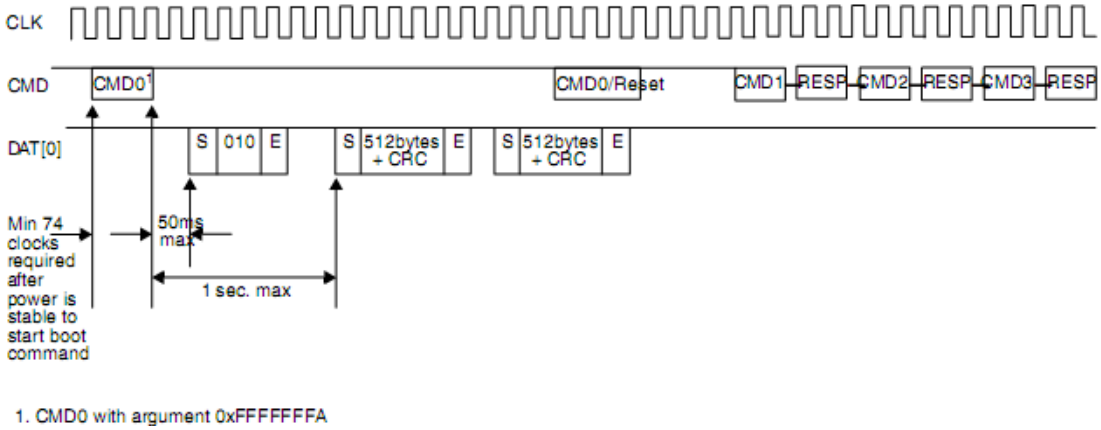


Fig 10-6 Alternative Boot Operation

Following are the steps that the software driver must follow when working with eMMC for the Alternative Boot operation.

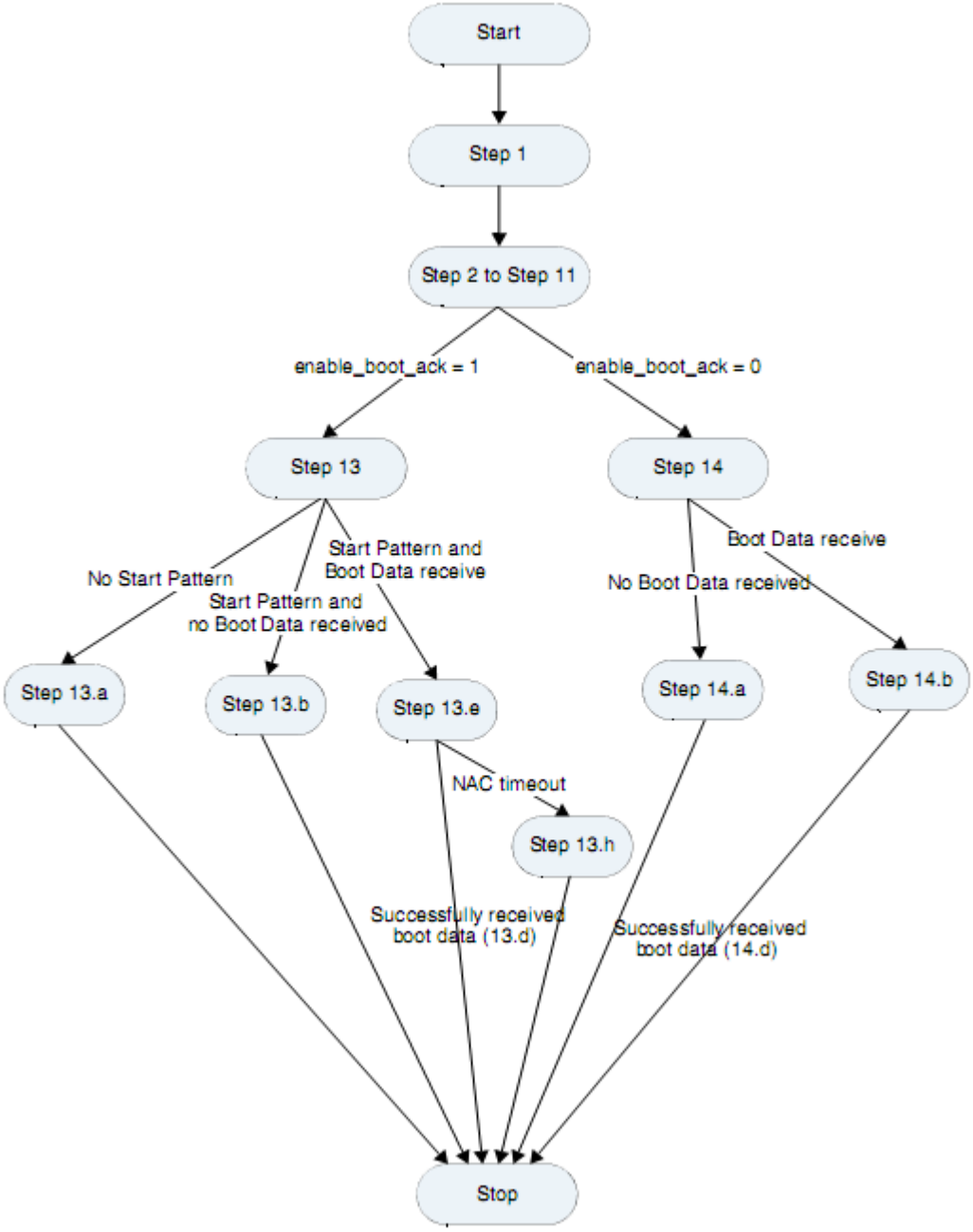


Fig 10-7 Host Controller Flow for Alternative Boot Mode

1. The software driver is aware:
   - That the card supports the Alternative Boot operation—BOOT_INFO bit is set in the card.
   - Of the BOOT_SIZE_MULT value in the card and the data bus width to use during the boot operation—Extend CSD register byte[177] bit[0:1]..
2. Set the following:
   - Masks for interrupts by clearing appropriate bits in the Interrupt Mask register @0x024
   - Global int_enable bit of the Control register @0x00

   It is recommended that you write 0xffff_ffff to the Raw Interrupt register @0x044 and IDSTS @0x8C in order to clear any pending interrupts before setting the int_enable bit.

   For Internal DMAC mode, software driver needs to unmask all the relevant fields in IDINTEN register.
3. Configure control register (CTRL):
   - enable_OD_pullup = 1'b0

- int_enable = 1'b1
- Other fields should be 1'b0

4. Changing clock source assignment – Set the card frequency to 400 KHz using the clock-divider and clock-source registers; for details, refer to "Clock Programming".Ensure that the card clock—cclk_out—is running.
5. Wait for a time that ensures that at least 74 card clock cycles have occurred on the card interface.
6. Set Data_timeout = (10 * ((TAAC * Fop) + (100 * NSAC)); this is NAC.
7. Program the BLKSIZ register with 0x200—512 bytes.
8. Program the BYTCNT register with multiples of 128K bytes, as indicated by the BOOT_SIZE_MULT value in the card.
9. Program the Rx FIFO threshold value in bytes in the FIFOTH register @0x04C.Typically, the threshold value can be set to half the FIFO depth; that is, RX_WMark = (FIFO_DEPTH/2) - 1.
10. Program CMDARG = 0xFFFFFFFA.
11. Program the CMD register with the following fields.
    - start_cmd = 1'b1
    - boot_mode = 1'b1
    - enable_boot_ack – depends on whether a start-acknowledge pattern is expected from the card.
    - Card_number – appropriate_card_number, obtained by referring to CDETECT register
    - Data_expected = 1'b1
    - Cmd_index = 0
    - Remainder of CMD register fields = 1'b0
12. The software driver should wait for the Command Done (CD) interrupt.
13. If enable_boot_ack = 1'b1 in step 11, the software driver should start a timer after the above step with a terminal value of 50ms.
    - Before this timer elapses, the BAR interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver needs to infer that the start-pattern has not been received and should discontinue the boot process and start with normal enumeration.
    - If the BAR interrupt is received, the software driver should clear this interrupt by writing a 1 to it. The software driver should then start another timer with a terminal value of 1 - 0.05 = 0.95 seconds. Before this timer elapses, the BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver should discontinue the boot process and start with normal enumeration.
    - If the BDS interrupt is received, it indicates that the boot data is being received from the card. In non-IDMAC mode, the software driver can then initiate a data read from the SD/MMC Controller based on the RXDR interrupt bit in the RINTSTS register.
    - It is the responsibility of the software driver to terminate the boot operation by programming the SD/MMC Controller to send a CMD0 by programming the registers CMDARG = 0 and CMD = {start_cmd = 1, card number = appropriate_card_number, cmd_index = 0, all_other_fields = 0}.
    - At the end of a successful boot data transfer from the card, the following interrupts are:
      - Command Done (CD) in RINTSTS
      - Data Transfer Over (DTO) in RINTSTS
      - Receive Interrupt (RI) in IDSTS in IDMAC mode only
    - If an Error occurs in Boot Ack pattern (010) or an end bit Error occurs:
      - RTL does not generate BAR interrupt
      - RTL detects Boot Data Start and generates BDS interrupt
      - RTL continues to receive Boot Data
      - Application must abort boot after receiving BDS interrupt
    - If between data block transfers NAC is violated, DRTO (Data Read Timeout) is

asserted. Apart from this, if there are errors associated with Start/End bits, SBE/EBE interrupts are also generated.

14. If enable_boot_ack = 1'b0 in step 11, the software driver should start a timer after step #11with a terminal value of 1 second.
    - Before this timer elapses, the BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver should discontinue the boot process and start with normal enumeration.
    - If the BDS interrupt is received, it indicates that the boot data is being received from the card. In non-IDMAC mode, the software driver can then initiate a data read from the SD/MMC Controller based on the RXDR (in RINTSTS) interrupt.
    - It is the responsibility of the software driver to terminate the boot operation by programming the SD/MMC Controller to send a CMD0 by programming the registers CMDARG = 0 and CMD = {start_cmd=1,card number = appropriate card number, cmd_index =0, rest of the fields = 0}.
    - At the end of a successful boot data transfer from card, the following interrupts are generated.
        - Command Done (CD) in RINTSTS.
        - Data Transfer Over (DTO) in RINTSTS.
        - Receive Interrupt (RI) in IDSTS in IDMAC mode only.

10.5.4 H/W Reset Operation

When the RST_n signal goes low, the card enters a pre-idle state from any state other than the inactive state.

H/W Reset Programming Sequence

The following outlines the steps for the H/W reset programming sequence:

1. Program CMD12 to end any transfer in process.
2. Wait for DTO, even if no response is sent back by the card.
3. Set the following resets:
    - DMA reset– CTRL[2]
    - FIFO reset – CTRL[1] bits

Note: The above steps are required only if a transfer is in process.

4. Program the CARD_RESET register with a value of 0; this can be done at any time when the card is connected to the controller. This programming asserts the RST_n signal and resets the card.
5. Wait for minimum of 1 µs or cclk_in period, whichever is greater
6. After a minimum of 1 µs, the application should program a value of 0 into the CARD_RESET register. This de-asserts the RST_n signal and takes the card out of reset.
7. The application can program a new CMD only after a minimum of 200 µs after the de-assertion of the RST_n signal, as per the MMC 4.41 standard.

Note: For backward compatibility, the RST_n signal is temporarily disabled in the card by default. The host may need to set the signal as either permanently enabled or permanently disabled before it uses the card.

10.6 Interface description

Table 10-7 SDMMC Interface Description

Module pin	Dir.	Pad name	IOMUX
SDMMC Interface			
ccmd	I/O	IO_SDMMCcmd_SPI1Acs_UART3tx_GPIO1a5	GRF_GPIO1A_IOMUX[11:10]=2'b01
cclk	O	IO_SDMMCclk_SPI1Aclk_UART3rx_GPIO1a6	GRF_GPIO1A_IOMUX[13:12]=2'b01
cdata0	I/O	IO_SDMMCd0_SPI1Arx_UART4tx_GPIO1	GRF_GPIO1A_IOMUX[15:14]=2'

		a7	b01
cdata1	I/O	IO_SDMMCd1_SPI1Atx_UART4rx_GPIO1b0	GRF_GPIO1B_IOMUX[1:0]=2'b01
cdata2	I/O	IO_SDMMCd2_I2C1Bscl_UART5tx_GPIO1b1	GRF_GPIO1B_IOMUX[3:2]=2'b01
cdata3	I/O	IO_SDMMCd3_I2C1Bsda_UART5rx_GPIO1b2	GRF_GPIO1B_IOMUX[5:4]=2'b01
SDIO Interface			
ccmd	I/O	IO_SDMMCcmd_SPI1Acs_UART3tx_GPIO1a5	GRF_GPIO1A_IOMUX[11:10]=2'b01
cclk	O	IO_SDMMCclk_SPI1Aclk_UART3rx_GPIO1a6	GRF_GPIO1A_IOMUX[13:12]=2'b01
cdata0	I/O	IO_SDMMCd0_SPI1Arx_UART4tx_GPIO1a7	GRF_GPIO1A_IOMUX[15:14]=2'b01
cdata1	I/O	IO_SDMMCd1_SPI1Atx_UART4rx_GPIO1b0	GRF_GPIO1B_IOMUX[1:0]=2'b01
cdata2	I/O	IO_SDMMCd2_I2C1Bscl_UART5tx_GPIO1b1	GRF_GPIO1B_IOMUX[3:2]=2'b01
cdata3	I/O	IO_SDMMCd3_I2C1Bsda_UART5rx_GPIO1b2	GRF_GPIO1B_IOMUX[5:4]=2'b01

Chapter 11 EMMC

11.1 Overview

The SDMMC Host Controller is designed to support Secure Digital memory(SD mem - version 2.00), Secure Digital I/O(SDIO-version 2.00), Multimedia Cards(MMC-version 4.41).The SDMMC support SD Card(1/4bit), SDIO, eMMC(1/4/8bit).

11.2 Features

- Supports AMBA AHB interface
- Supports DMA controller for data transfers
- Supports interrupt output
- Supports SD version2.0 except SPI mode
- Supports MMC version4.41 except SPI mode
- Supports SDIO version2.0
- 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

11.3 Architecture

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

11.3.1 Block Diagram

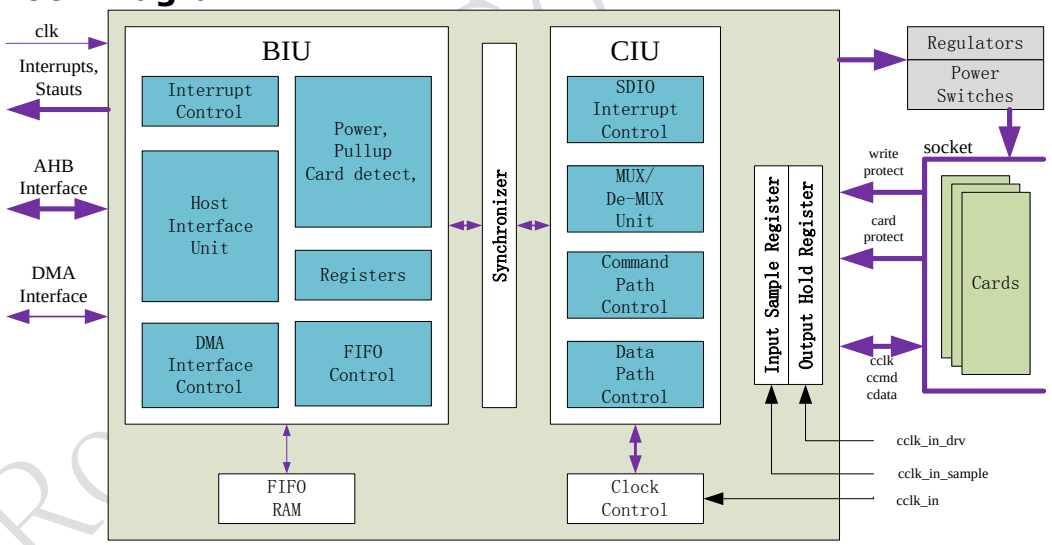


Fig 11-1 SD/MMC Controller Interface Signals

11.3.2 Block Descriptions

- 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.

11.4 Registers

11.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description

Name	Offset	Size	Reset Value	Description
SDMMC_CTRL	0x0000	W	0x00000000	Control register
SDMMC_PWREN	0x0004	W	0x00000000	Power-enable register
SDMMC_CLKDIV	0x0008	W	0x00000000	Clock-divider register
SDMMC_CLKENA	0x0010	W	0x00000000	Clock-enable register
SDMMC_TMOUT	0x0014	W	0xffffff40	Time-out register
SDMMC_CTYPE	0x0018	W	0x00000000	Card-type register
SDMMC_BLKSIZ	0x001c	W	0x00000200	Block-size register
SDMMC_BYTCNT	0x0020	W	0x00000200	Byte-count register
SDMMC_INTMASK	0x0024	W	0x00000000	Interrupt-mask register
SDMMC_CMDARG	0x0028	W	0x00000000	Command-argument register
SDMMC_CMD	0x002c	W	0x00000000	Command register
SDMMC_RESP0	0x0030	W	0x00000000	Response-0 register
SDMMC_RESP1	0x0034	W	0x00000000	Response-1 register
SDMMC_RESP2	0x0038	W	0x00000000	Response-2 register
SDMMC_RESP3	0x003c	W	0x00000000	Response-3 register
SDMMC_MINTSTS	0x0040	W	0x00000000	Masked interrupt-status register
SDMMC_RINTSTS	0x0044	W	0x00000000	Raw interrupt-status register
SDMMC_STATUS	0x0048	W	0x00000406	Status register
SDMMC_FIFOTH	0x004c	W	0x00000000	FIFO threshold register
SDMMC_CDETECT	0x0050	W	0x00000000	Card-detect register
SDMMC_WRTPRT	0x0054	W	0x00000000	Write-protect register
SDMMC_TCBCNT	0x005c	W	0x00000000	Transferred CIU card byte count
SDMMC_TBBCNT	0x0060	W	0x00000000	Transferred host/DMA to/from BIU-FIFO byte count
SDMMC_DEBNCE	0x0064	W	0x00ffffff	Card detect debounce register
SDMMC_USRID	0x0068	W	0x00000000	User ID register
SDMMC_RST_n	0x0078	W	0x00000001	Hardware reset register
SDMMC_BACK_END_POWER	0x0104	W	0x00000000	Back-end Power
SDMMC_FIFO_BASE	0x0200	W	0x00000000	

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

11.4.2 Detail Register Description

SDMMC_CTRL

Address: Operational Base + offset (0x0000)

Control register

Bit	Attr	Reset Value	Description
31:9	RO	0x0	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.

Bit	Attr	Reset Value	Description
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(s). In meantime, if host wants SDMMC Controller to exit waiting for interrupt state, it can set this bit, at which time SDMMC 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 For sending read-wait to SDIO cards
5	RW	0x0	dma_enable 0 –Disable DMA transfer mode 1 –Enable DMA transfer mode Even when DMA mode is enabled, host can still push/pop data into or from FIFO; this should not happen during the normal operation. If there is simultaneous FIFO access from host/DMA, the data coherency is lost. Also, there is no arbitration inside SDMMC Controller to prioritize simultaneous host/DMA access.
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	RO	0x0	reserved
2	W1C	0x0	dma_reset 0 –No change 1 –Reset internal DMA interface control logic To reset DMA interface, firmware should set bit to 1. This bit is auto-cleared after two AHB clocks.
1	W1C	0x0	fifo_reset 0 –No change 1 –Reset to 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

Bit	Attr	Reset Value	Description
0	W1C	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 Does not affect any registers or DMA interface, or FIFO or host interrupts

SDMMC_PWREN
Address: Operational Base + offset (0x0004)
Power-enable register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	power_enable Power on/off switch for the card. 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 Bit values output to card_power_en port.

SDMMC_CLKDIV
Address: Operational Base + offset (0x0008)
Clock-divider register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	clk_divider0 Clock divider-0 value. Clock division is 2*n. For example, value of 0 means divide by 2*0 = 0 (no division, bypass), value of 1 means divide by 2*1 = 2

SDMMC_CLKENA
Address: Operational Base + offset (0x0010)
Clock-enable register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved

Bit	Attr	Reset Value	Description
16	RW	0x0	cclk_low_power Low-power control for SD card clock and MMC card clock supported. 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	RO	0x0	reserved
0	RW	0x0	cclk_enable Clock-enable control for SD card clock and MMC card clock supported. 0 –Clock disabled 1 –Clock enabled

SDMMC_TMOUT
Address: Operational Base + offset (0x0014)
Time-out register

Bit	Attr	Reset Value	Description
31:8	RW	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	RW	0x40	response_timeout Response timeout value. Value is in number of card output clocks –cclk_out.

SDMMC_CTYPE
Address: Operational Base + offset (0x0018)
Card-type register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RW	0x0	card_width_8 Indicates if card is 8-bit: 0 –Non 8-bit mode 1 –8-bit mode
15:1	RO	0x0	reserved
0	RW	0x0	card_width Indicates if card is 1-bit or 4-bit: 0 –1-bit mode 1 –4-bit mode

SDMMC_BLKSIZ
Address: Operational Base + offset (0x001c)
Block-size register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0200	block_size Block size

SDMMC_BYTCNT
Address: Operational Base + offset (0x0020)
Byte-count register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000200	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 (0x0024)
Interrupt-mask register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RO	0x0	sdio_int_mask Mask SDIO interrupts When masked, SDIO interrupt detection for that card is disabled. A 0 masks an interrupt, and 1 enables an interrupt.
15:0	RW	0x0000	int_mask Bits used to mask unwanted interrupts. Value of 0 masks interrupt; value of 1 enables interrupt. bit 15 –End-bit error (read)/Write no CRC (EBE) bit 14 –Auto command done (ACD) bit 13 –Start-bit error (SBE) bit 12 –Hardware locked write error (HLE) bit 11 –FIFO underrun/overrun error (FRUN) bit 10 –Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9 –Data read timeout (DRTO) bit 8 –Response timeout (RTO) bit 7 –Data CRC error (DCRC) bit 6 –Response CRC error (RCRC) bit 5 –Receive FIFO data request (RXDR) bit 4 –Transmit FIFO data request (TXDR) bit 3 –Data transfer over (DTO) bit 2 –Command done (CD) bit 1 –Response error (RE) bit 0 –Card detect (CD)

SDMMC_CMDARG
Address: Operational Base + offset (0x0028)
Command-argument register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	cmd_arg Value indicates command argument to be passed to card.

SDMMC_CMD
Address: Operational Base + offset (0x002c)
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. Once command is sent and response is received from SD_MMC cards, Command Done bit is set in raw interrupt register.
30	RO	0x0	reserved
29	RW	0x0	use_hold_reg Use Hold Register 0 - CMD and DATA sent to card bypassing HOLD Register 1 - CMD and DATA sent to card through the HOLD Register Note: a. Set to 1'b1 for SDR12 and SDR25 (with non-zero phase-shifted cclk_in_drv); zero phase shift is not allowed in these modes. b. Set to 1'b0 for SDR50(with zero phase- shifted cclk_in_drv) c. Set to 1'b1 for SDR50 (with non-zero phase-shifted cclk_in_drv)
28	RW	0x0	volt_switch Voltage switch bit 0 - No voltage switching 1 - Voltage switching enabled; must be set for CMD11 only
27	RW	0x0	boot_mode Boot Mode 0 - Mandatory Boot operation 1 - Alternate Boot operation

Bit	Attr	Reset Value	Description
26	RW	0x0	disable_boot Disable Boot. When software sets this bit along with start_cmd, CIU terminates the boot operation. Do NOT set disable_boot and enable_boot together.
25	RW	0x0	expect_boot_ack Expect Boot Acknowledge. When Software sets this bit along with enable_boot, CIU expects a boot acknowledge start pattern of 0-1-0 from the selected card.
24	RW	0x0	enable_boot Enable Boot—this bit should be set only for mandatory boot mode. When Software sets this bit along with start_cmd, CIU starts the boot sequence for the corresponding card by asserting the CMD line low. Do NOT set disable_boot and enable_boot together.
23:22	RO	0x0	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, CLRSRC, and CLKENA. Changes card clocks (change frequency, truncate 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 is no Command Done interrupts because no command is sent to SD_MMC cards.
20:16	RO	0x0	reserved

Bit	Attr	Reset Value	Description
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 so that controller will initialize clocks before sending command to card. This bit should not be set for either of the boot modes (alternate or mandatory).
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. This is also applicable for Boot mode transfers. To Abort boot mode, this bit should be set along with CMD[26] = disable_boot.
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.

Bit	Attr	Reset Value	Description
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 SD_MMC cards at end of data transfer. * when send_auto_stop bit should be set, since some data transfers do not need explicit stop commands * open-ended transfers that software should explicitly send to stop command Additionally, when "resume" is sent to resume –suspended memory access of SD-Combo card –bit should be set correctly if suspended data transfer needs send_auto_stop. Don't care if no data expected from card.
11	RW	0x0	transfer_mode 0 –Block data transfer command 1 –Stream data transfer command Don't care if no data expected.
10	RW	0x0	wr 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	0x00	cmd_index Command index

SDMMC_RESP0

Address: Operational Base + offset (0x0030)

Response-0 register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response0 Bit[31:0] of response

SDMMC_RESP1
Address: Operational Base + offset (0x0034)
Response-1 register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response 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.

SDMMC_RESP2
Address: Operational Base + offset (0x0038)
Response-2 register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response2 Bit[95:64] of long response

SDMMC_RESP3
Address: Operational Base + offset (0x003c)
Response-3 register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	response3 Bit[127:96] of long response

SDMMC_MINTSTS
Address: Operational Base + offset (0x0040)
Masked interrupt-status register

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RO	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 enables interrupt; 0 masks interrupt). 0 –No SDIO interrupt from card 1 –SDIO interrupt from card
23:17	RO	0x0	reserved
16	RW	0x0	new_int_status New Interrupt Status 0: data_no busy int

Bit	Attr	Reset Value	Description
15:0	RO	0x0000	int_status Interrupt enabled only if corresponding bit in interrupt mask register is set. bit 15 –End-bit error (read)/write no CRC (EBE) bit 14 –Auto command done (ACD) bit 13 –Start-bit error (SBE) bit 12 –Hardware locked write error (HLE) bit 11 –FIFO underrun/overrun error (FRUN) bit 10 –Data starvation by host timeout (HTO)/Volt_switch_int bit 9 –Data read timeout (DRTO) bit 8 –Response timeout (RTO) bit 7 –Data CRC error (DCRC) bit 6 –Response CRC error (RCRC) bit 5 –Receive FIFO data request (RXDR) bit 4 –Transmit FIFO data request (TXDR) bit 3 –Data transfer over (DTO) bit 2 –Command done (CD) bit 1 –Response error (RE) bit 0 –Card detect (CD)

SDMMC_RINTSTS

Address: Operational Base + offset (0x0044)

Raw interrupt-status register

Bit	Attr	Reset Value	Description
31:17	RO	0x0	reserved
16	RO	0x0	sdio_interrupt Interrupt from SDIO card; Writes to these bits clear them. Value of 1 clears bit and 0 leaves bit intact. 0 –No SDIO interrupt from card 1 –SDIO interrupt from card

Bit	Attr	Reset Value	Description
15:0	RO	0x0000	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. bit 15 –End-bit error (read)/write no CRC (EBE) bit 14 –Auto command done (ACD) bit 13 –Start-bit error (SBE) bit 12 –Hardware locked write error (HLE) bit 11 –FIFO underrun/overrun error (FRUN) bit 10 –Data starvation-by-host timeout (HTO) /Volt_switch_int bit 9 –Data read timeout (DRTO)/Boot Data Start (BDS) bit 8 –Response timeout (RTO)/Boot Ack Received (BAR) bit 7 –Data CRC error (DCRC) bit 6 –Response CRC error (RCRC) bit 5 –Receive FIFO data request (RXDR) bit 4 –Transmit FIFO data request (TXDR) bit 3 –Data transfer over (DTO) bit 2 –Command done (CD) bit 1 –Response error (RE) bit 0 –Card detect (CD)

SDMMC_STATUS
Address: Operational Base + offset (0x0048)
Status register

Bit	Attr	Reset Value	Description
31	RO	0x0	dma_req DMA request signal state
30	RO	0x0	dma_ack DMA acknowledge signal state
29:17	RO	0x0000	fifo_count FIFO count –Number of filled locations in FIFO
16:11	RO	0x00	response_index Index of previous response, including any auto-stop sent by core
10	RO	0x1	data_state_mc_busy Data transmit or receive state-machine is busy
9	RO	0x0	data_busy Inverted version of raw selected card_data[0] 0 –card data not busy 1 –card data busy default value is 1 or 0 depending on cdata_in

Bit	Attr	Reset Value	Description
8	RO	0x0	data_3_status Raw selected card_data[3]; checks whether card is present 0 –card not present 1 –card present default value is 1 or 0 depending on cdata_in
7:4	RO	0x0	command_fsm_states 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 NOTE: The command FSM state is represented using 19 bits. The STATUS Register(7:4) has 4 bits to represent the command FSM states. Using these 4 bits, only 16 states can be represented. Thus three states cannot be represented in the STATUS(7:4) register. The three states that are not represented in the STATUS Register(7:4) are: * Bit 16 –Wait for CCS * Bit 17 –Send CCSD * Bit 18 –Boot Mode Due to this, while command FSM is in "Wait for CCS state" or "Send CCSD" or "Boot Mode", the Status register indicates status as 0 for the bit field 7:4.
3	RO	0x0	fifo_full FIFO is full status
2	RO	0x1	fifo_empty FIFO is empty status
1	RO	0x1	fifo_tx_watermark FIFO reached Transmit watermark level; not qualified with data transfer

Bit	Attr	Reset Value	Description
0	RO	0x0	fifo_rx_watermark FIFO reached Receive watermark level; not qualified with data transfer

SDMMC_FIFOTH
Address: Operational Base + offset (0x004c)
FIFO threshold register

Bit	Attr	Reset Value	Description
31	RO	0x0	reserved
30:28	RW	0x0	DMA_Mutiple_Transaction_Size Burst size of multiple transaction; should be programmed same as 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 The units for transfers is the H_DATA_WIDTH parameter. A single transfer would be signaled based on this value. Value should be sub-multiple of (RX_WMark + 1)* (F_DATA_WIDTH/H_DATA_WIDTH) and (FIFO_DEPTH - TX_WMark)* (F_DATA_WIDTH/ H_DATA_WIDTH) For example, if FIFO_DEPTH = 16, FDATA_WIDTH == H_DATA_WIDTH Allowed combinations for MSize and TX_WMark are: MSize = 1, TX_WMARK = 1-15 MSize = 4, TX_WMark = 8 MSize = 4, TX_WMark = 4 MSize = 4, TX_WMark = 12 MSize = 8, TX_WMark = 8 MSize = 8, TX_WMark = 4 Allowed combinations for MSize and RX_WMark are: MSize = 1, RX_WMARK = 0-14 MSize = 4, RX_WMark = 3 MSize = 4, RX_WMark = 7 MSize = 4, RX_WMark = 11 MSize = 8, RX_WMark = 7 Recommended: MSize = 8, TX_WMark = 8, RX_WMark = 7

Bit	Attr	Reset Value	Description
27:16	RW	0x000	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 enabled, 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. 12 bits –1 bit less than FIFO-count of status register, which is 13 bits. Limitation: RX_WMark <= FIFO_DEPTH-2 Recommended: (FIFO_DEPTH/2) - 1; (means greater than (FIFO_DEPTH/2) - 1) NOTE: In DMA mode during CCS time-out, the DMA does not generate the request at the end of packet, even if remaining bytes are less than threshold. In this case, there will be some data left in the FIFO. It is the responsibility of the application to reset the FIFO after the CCS timeout.
15:12	RO	0x0	reserved

Bit	Attr	Reset Value	Description
11:0	RW	0x000	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 enabled, 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. 12 bits –1 bit less than FIFO-count of status register, which is 13 bits. Limitation: TX_WMark >= 1; Recommended: FIFO_DEPTH/2; (means less than or equal to FIFO_DEPTH/2)

SDMMC_CDETECT
Address: Operational Base + offset (0x0050)
Card-detect register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	card_detect_n Value on card_detect_n input ports; read-only bits. 0 represents presence of card.

SDMMC_WRTPRT
Address: Operational Base + offset (0x0054)
Write-protect register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	write_protect Value on card_write_prt input port. 1 represents write protection.

SDMMC_TCBCNT
Address: Operational Base + offset (0x005c)
Transferred CIU card byte count

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	trans_card_byte_count Number of bytes transferred by CIU unit to card. In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied. Both TCBCNT and TBBCNT share same coherency register. When AREA_OPTIMIZED parameter is 1, register should be read only after data transfer completes; during data transfer, register returns 0.

SDMMC_TBBCNT
Address: Operational Base + offset (0x0060)
Transferred host/DMA to/from BIU-FIFO byte count

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	trans_fifo_byte_count Number of bytes transferred between Host/DMA memory and BIU FIFO. In 32-bit or 64-bit AMBA data-bus-width modes, register should be accessed in full to avoid read-coherency problems. In 16-bit AMBA data-bus-width mode, internal 16-bit coherency register is implemented. User should first read lower 16 bits and then higher 16 bits. When reading lower 16 bits, higher 16 bits of counter are stored in temporary register. When higher 16 bits are read, data from temporary register is supplied. Both TCBCNT and TBBCNT share same coherency register.

SDMMC_DEBNCE
Address: Operational Base + offset (0x0064)
Card detect debounce register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:0	RW	0xffffff	debounce_count Number of host clocks (clk) used by debounce filter logic; typical debounce time is 5-25 ms.

SDMMC_USRID
Address: Operational Base + offset (0x0068)
User ID register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	USRID User identification register; value set by user. Default reset value can be picked by user while configuring core before synthesis. Can also be used as scratch pad register by user. the default value is determined by Configuration Value.

SDMMC_RST_n
Address: Operational Base + offset (0x0078)
Hardware reset register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x1	CARD_RESET Hardware reset. 1 –Active mode 0 –Reset These bits cause the cards to enter pre-idle state, which requires them to be re-initialized. CARD_RESET[0] should be set to 1'b1 to reset card

SDMMC_BACK_END_POWER
Address: Operational Base + offset (0x0104)
Back-end Power

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	Back_End_Power Back end power 1'b0 –Off; Reset 1'b1 –Back-end Power supplied to card application

SDMMC_FIFO_BASE
Address: Operational Base + offset (0x0200)

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	fifo_base_addr fifo base addr

11.5 Application Notes

11.5.1 Software/Hardware Restriction

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.

If the card is enumerated in SDR12 or SDR25 mode, the application must program the use_hold_reg bit[29] in the CMD register to 1'b1.

This programming should be done for all data transfer commands and non-data commands that are sent to the card. When the use_hold_reg bit is programmed to 1'b0, the SD/MMC Controller bypasses the Hold Registers in the transmit path. The value of this bit should not be changed when a Command or Data Transfer is in progress. For more details on

using use_hold_reg and the implementation requirements for meeting the Card input hold time, refer to "Recommended Usage" and Table 11-1.

Table 11-1Recommended Usage of use_hold_reg

No.	Speed Mode	use_hold_reg	cclk_in	clk_in_drv	clk_divider
1	SDR25	1'b1	50	50	0
2	SDR12	1'b1	50	50	1

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

5. 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.
6. 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.
7. 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.
8. 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.

11.5.2 Programming Sequence

Initialization

Fig. 11-2 illustrates the initialization flow.

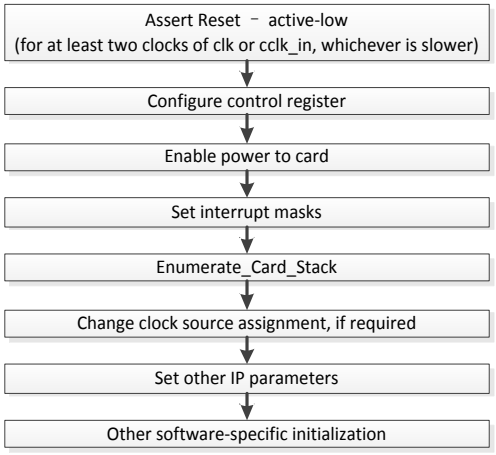


Fig 11-2 Initialization Sequence

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 pull-up 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 sdmmc_clkout according to SDMMC specifications.
   - ResponseTimeOut = 0x64
   - Data_timeout = 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

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.

7. Check if the card is connected.
8. 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.
9. 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.
10. Identify the card type; that is, SD, MMC, or SDIO.
    f. Send CMD5 first. If a response is received, then the card is SDIO
    g. 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
    h. 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
    i. If Response is received for ACMD41 then the card is SD. Otherwise the card is MMC.
    j. 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

11. Enumerate the card according to the card type.
12. 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 disabled while switching off the power.

Clock Programming

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_cmdis 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 along response is copied to all four response registers @0x30, 0x34, 0x38, and 0x3C. The Response3register 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 11-2.

Table 11-2Command Settings for No-Data Command

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
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 Host 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:

7. Write the data size in bytes in the BYTCNT register @0x20.
8. 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.
9. 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 11-3. 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 11-3Command Register Setting for Single-Block or Multiple-Block Read

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
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	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, the SD/MMC controller starts executing the command; when the command is sent to the bus, the command_done interrupt is generated.
10. 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.
11. 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.
12. 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 11-4. 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 11-4Command Register Settings for Single-Block or Multiple-Block Write

Parameter	Value	Description
Default		
start_cmd	1	-
use_hold_reg	1/0	Choose value based on speed mode being used; ref to "use_hold_reg" on CMD register
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 Host 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 theAuto_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 ASx 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. 11-3:

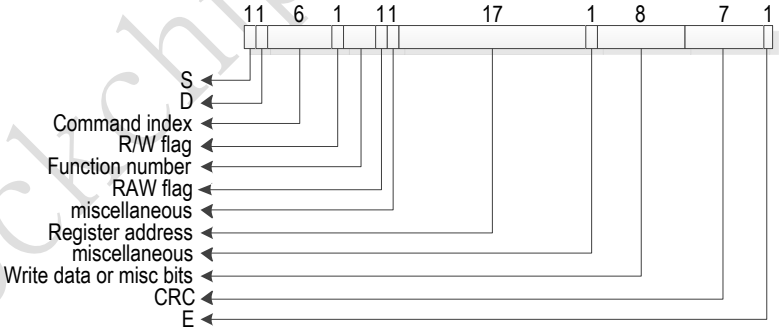


Fig 11-3 Command format for CMD52

e. Program the CMDARG register @0x28 with the appropriate command argument parameters listed in Table 11-5.

Table 11-5Parameters 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

f. 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.
g. Wait for command_transfer_over.
h. 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 CMD52command.

1. SUSPEND data transfer – Non-data command.
   g. 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.
   h. 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 FSx bits is valid.
   i. To suspend the transfer, set BR (bit 2) of the CCCR register @0x0C.
   j. 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.
   k. 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.
   l. Wait for data completion. Get pending bytes to transfer by reading the TCBCNT register @0x5C.

RESUME data transfer – This is a data command.

   j. Check that the card is not in a transfer state, which confirms that the bus is free for data transfer.
   k. If the card is in a disconnect state, select it using CMD7. The card status can be retrieved in response to CMD52/CMD53 commands.
   l. 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.
   m. To resume transfer, use CMD52 to write the function number at FSx 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 11-6.

Table 11-6CMDARG 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

n. Write the block size in the BLKSIZ register @0x1C; data will be transferred in units of this block size.
o. 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.
p. 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".
q. 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.
r. If the DF flag is 0, then in case of a read, the SD/MMC Host 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 Host 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 DMA controller that are interfaced to the SD/MMC Host 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.

Notes: 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.

11.5.3 ProgrammingEMMC Controller for Boot Operation

Boot Operation

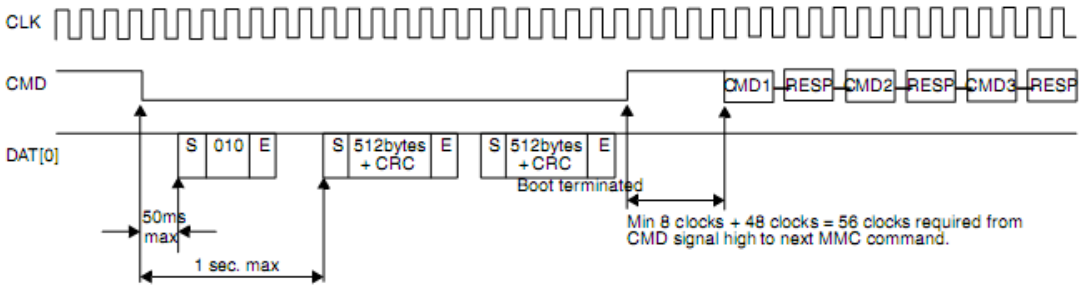


Fig 11-4 illustrates timing for Boot operation

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 following:

- Registers
- Ports
- FIFO-pointers
- DMA interface controls
- State-machines in the design

After power-on reset, the software should perform the appropriate steps described in the following sections for the respective types of cards.

Following are the steps that the software driver must follow when working with eMMC cards for Boot operation.

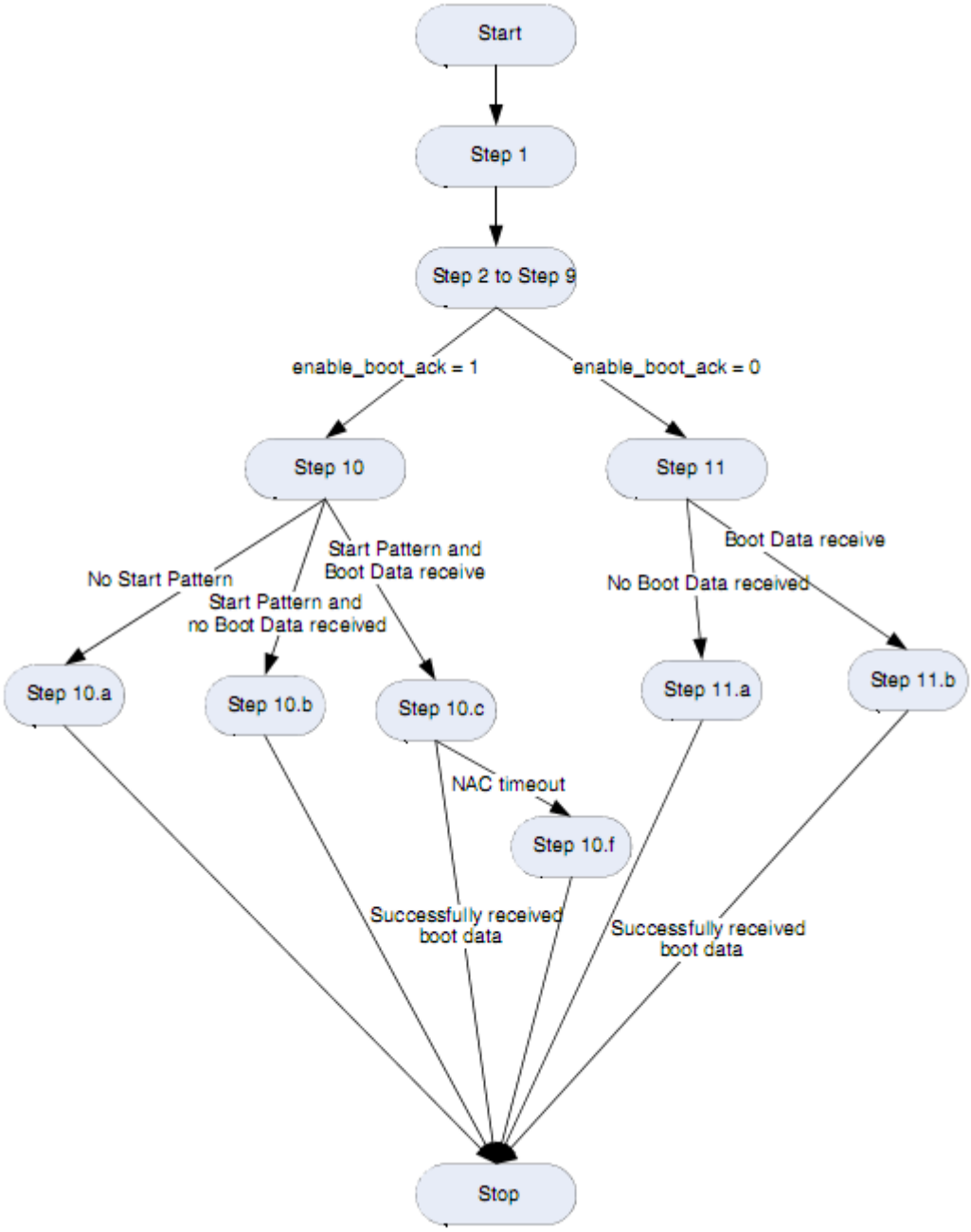


Fig 11-5 SD/MMC Controller Flow for Boot Operation

1. The software driver is aware:
   - That the card supports boot operation—BOOT_PARTITION_ENABLE bit set in the card.
   - Of the BOOT_SIZE_MULT value in the card and the data bus width to use during boot operation—Extend CSD register byte[177] bit[0:1].
2. Set the following:
   - Masks for interrupts by clearing appropriate bits in the Interrupt Mask register @0x024.
   - Global int_enable bit of the Control register @0x00.

   It is recommended that you write 0xffff_ffff to the Raw Interrupt register @0x044 and IDSTS @0x8C in order to clear any pending interrupts before setting the int_enable bit.

   For Internal DMAC mode, the software driver needs to unmask all the relevant fields in the IDINTEN register.

3. Configure control register (CTRL):
   - int_enable = 1'b1
   - Other fields should be 1'b0.
4. Change clock source assignment – Set the card frequency to 400 KHz using the clock-divider and clock-source registers; for details, refer to "Clock Programming".
5. Set Data_timeout = (10 * ((TAAC * Fop) + (100 * NSAC)); this is NAC.
6. Program the BLKSIZ register with 0x200 (512 bytes).
7. Program the BYTCNT register with multiples of 128K bytes, as indicated by the BOOT_SIZE_MULT value in the card.
8. Program the Rx FIFO threshold value in bytes in the FIFOTH register @0x04C.Typically, the threshold value can be set to half the FIFO depth; that is, RX_WMark = (FIFO_DEPTH/2) - 1.
9. Program the CMD register with the following fields:
   - start_cmd = 1'b1
   - enable_boot = 1'b1
   - enable_boot_ack – depends on whether a start-acknowledge pattern is expected from the card
   - Card_number = appropriate_card_number; obtained by referring to CDETECT register
   - Data_expected = 1'b1
   - Remainder of CMD register fields = 1'b0
10. If enable_boot_ack = 1'b1, the software driver should start a timer after step #9; the terminal value is 50ms.
    - Before this timer elapses, the BAR interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver must program the CMD register with the following fields:
      - start_cmd = 1'b1
      - disable_boot = 1'b1
      - All other fields = 0

      The SD/MMC Controller generates a Command Done (CD) interrupt after de-asserting the CMD line of the card.
    - If the BAR interrupt is received, the software driver should clear this interrupt by writing a 1 to it. The software driver should then start another timer with a terminal value of 1 - 0.05 = 0.95 seconds. Before this timer elapses, the BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver must program the CMD register with the following fields:
      - start_cmd = 1'b1
      - disable_boot = 1'b1
      - All other fields = 0

      The SD/MMC Controller generates a Command Done (CD) interrupt after de-asserting the CMD line of the card.
    - If the BDS interrupt is received, it indicates that the boot data is being received from the card. The software driver can then initiate a data read from the SD/MMC Controller based on the RXDR interrupt bit in the RINTSTS register. At the end of a successful boot data transfer from the card, the following interrupts are generated:
      - Command Done (CD) in RINTSTS register

- Data Transfer Over (DTO) in RINTSTS register
- If an Error occurs in Boot Ack pattern (010) or an end bit Error occurs:
  - RTL automatically aborts boot by pulling CMD line high
  - RTL generates Command done interrupt
  - RTL does not generate BAR interrupt
  - Application aborts boot transfer
- If between data block transfers NAC is violated, DRTO (Data Read Timeout) is asserted. Apart from this, if there are errors associated with Start/End bits, SBE/EBE interrupts are also generated.

11. If enable_boot_ack = 1'b0, the software driver should start a timer after the step #9 where the terminal value is 1 second.
    - Before this timer elapses, a BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver must program the CMD register with the following fields:
      - start_cmd = 1'b1
      - disable_boot = 1'b1
      - All other fields = 0

    The SD/MMC Controller generates a Command Done (CD) interrupt after de-asserting the CMD line of the card.
    - If a BDS interrupt is received, it indicates that the boot data is being received from the card. The software driver can then initiate a data read from the SD/MMC Controller based on the RXDR interrupt bit in the RINTSTS register.

    At the end of a successful boot data transfer from card, the following interrupts are generated.
      - Command Done (CD) in RINTSTS.
      - Data Transfer Over (DTO) in RINTSTS.

Alternative Boot Operation

The Alternative Boot Operation differs from the Boot Operation in that CMD0 is used to boot the card rather than holding down the CMD-line of the card. The Alternative Boot Operation can be done only if bit 0 in the extended CSD byte[228] (BOOT_INFO) is set to 1.

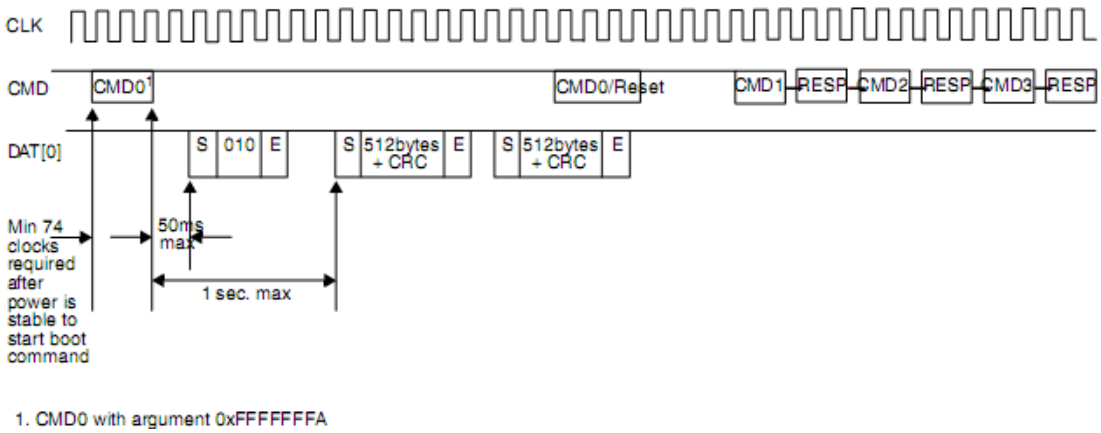


Fig 11-6 Alternative Boot Operation

Following are the steps that the software driver must follow when working with eMMC for the Alternative Boot operation.

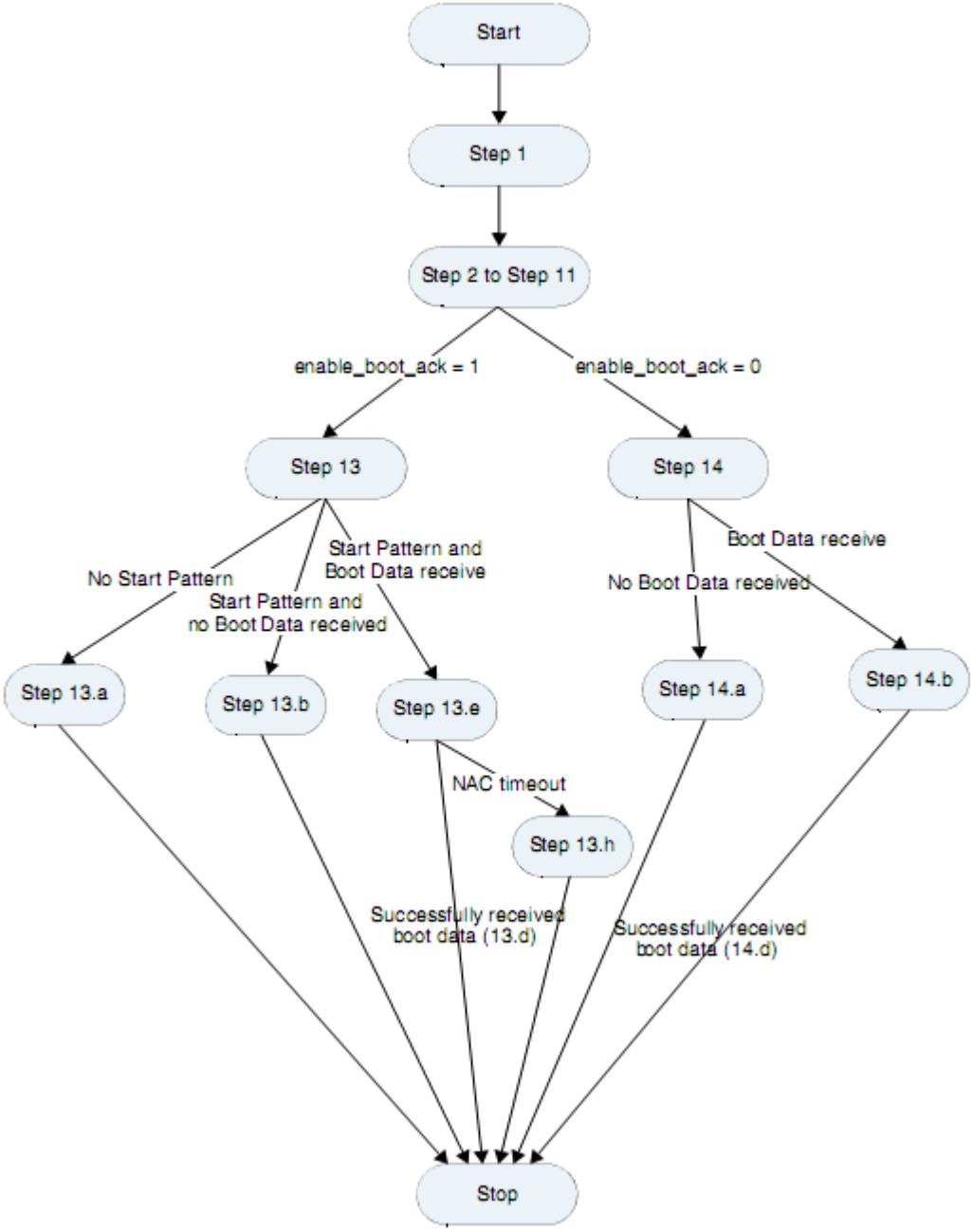


Fig 11-7 Host Controller Flow for Alternative Boot Mode

1. The software driver is aware:
   - That the card supports the Alternative Boot operation—BOOT_INFO bit is set in the card.
   - Of the BOOT_SIZE_MULT value in the card and the data bus width to use during the boot operation—Extend CSD register byte[177] bit[0:1]..
2. Set the following:
   - Masks for interrupts by clearing appropriate bits in the Interrupt Mask register @0x024
   - Global int_enable bit of the Control register @0x00

   It is recommended that you write 0xffff_ffff to the Raw Interrupt register @0x044 and IDSTS @0x8C in order to clear any pending interrupts before setting the int_enable bit.

   For Internal DMAC mode, software driver needs to unmask all the relevant fields in IDINTEN register.
3. Configure control register (CTRL):
   - enable_OD_pullup = 1'b0

- int_enable = 1'b1
- Other fields should be 1'b0

4. Changing clock source assignment – Set the card frequency to 400 KHz using the clock-divider and clock-source registers; for details, refer to "Clock Programming".Ensure that the card clock—cclk_out—is running.
5. Wait for a time that ensures that at least 74 card clock cycles have occurred on the card interface.
6. Set Data_timeout = (10 * ((TAAC * Fop) + (100 * NSAC)); this is NAC.
7. Program the BLKSIZ register with 0x200—512 bytes.
8. Program the BYTCNT register with multiples of 128K bytes, as indicated by the BOOT_SIZE_MULT value in the card.
9. Program the Rx FIFO threshold value in bytes in the FIFOTH register @0x04C.Typically, the threshold value can be set to half the FIFO depth; that is, RX_WMark = (FIFO_DEPTH/2) - 1.
10. Program CMDARG = 0xFFFFFFFA.
11. Program the CMD register with the following fields.
    - start_cmd = 1'b1
    - boot_mode = 1'b1
    - enable_boot_ack – depends on whether a start-acknowledge pattern is expected from the card.
    - Card_number – appropriate_card_number, obtained by referring to CDETECT register
    - Data_expected = 1'b1
    - Cmd_index = 0
    - Remainder of CMD register fields = 1'b0
12. The software driver should wait for the Command Done (CD) interrupt.
13. If enable_boot_ack = 1'b1 in step 11, the software driver should start a timer after the above step with a terminal value of 50ms.
    - Before this timer elapses, the BAR interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver needs to infer that the start-pattern has not been received and should discontinue the boot process and start with normal enumeration.
    - If the BAR interrupt is received, the software driver should clear this interrupt by writing a 1 to it. The software driver should then start another timer with a terminal value of 1 - 0.05 = 0.95 seconds. Before this timer elapses, the BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver should discontinue the boot process and start with normal enumeration.
    - If the BDS interrupt is received, it indicates that the boot data is being received from the card. In non-IDMAC mode, the software driver can then initiate a data read from the SD/MMC Controller based on the RXDR interrupt bit in the RINTSTS register.
    - It is the responsibility of the software driver to terminate the boot operation by programming the SD/MMC Controller to send a CMD0 by programming the registers CMDARG = 0 and CMD = {start_cmd = 1, card number = appropriate_card_number, cmd_index = 0, all_other_fields = 0}.
    - At the end of a successful boot data transfer from the card, the following interrupts are:
      - Command Done (CD) in RINTSTS
      - Data Transfer Over (DTO) in RINTSTS
      - Receive Interrupt (RI) in IDSTS in IDMAC mode only
    - If an Error occurs in Boot Ack pattern (010) or an end bit Error occurs:
      - RTL does not generate BAR interrupt
      - RTL detects Boot Data Start and generates BDS interrupt
      - RTL continues to receive Boot Data
      - Application must abort boot after receiving BDS interrupt
    - If between data block transfers NAC is violated, DRTO (Data Read Timeout) is

asserted. Apart from this, if there are errors associated with Start/End bits, SBE/EBE interrupts are also generated.

14. If enable_boot_ack = 1'b0 in step 11, the software driver should start a timer after step #11with a terminal value of 1 second.
    - Before this timer elapses, the BDS interrupt should be received from the SD/MMC Controller. If this does not occur, the software driver should discontinue the boot process and start with normal enumeration.
    - If the BDS interrupt is received, it indicates that the boot data is being received from the card. In non-IDMAC mode, the software driver can then initiate a data read from the SD/MMC Controller based on the RXDR (in RINTSTS) interrupt.
    - It is the responsibility of the software driver to terminate the boot operation by programming the SD/MMC Controller to send a CMD0 by programming the registers CMDARG = 0 and CMD = {start_cmd=1,card number = appropriate card number, cmd_index =0, rest of the fields = 0}.
    - At the end of a successful boot data transfer from card, the following interrupts are generated.
        - Command Done (CD) in RINTSTS.
        - Data Transfer Over (DTO) in RINTSTS.
        - Receive Interrupt (RI) in IDSTS in IDMAC mode only.

11.5.4 H/W Reset Operation

When the RST_n signal goes low, the card enters a pre-idle state from any state other than the inactive state.

H/W Reset Programming Sequence

The following outlines the steps for the H/W reset programming sequence:

1. Program CMD12 to end any transfer in process.
2. Wait for DTO, even if no response is sent back by the card.
3. Set the following resets:
    - DMA reset– CTRL[2]
    - FIFO reset – CTRL[1] bits

Note: The above steps are required only if a transfer is in process.

4. Program the CARD_RESET register with a value of 0; this can be done at any time when the card is connected to the controller. This programming asserts the RST_n signal and resets the card.
5. Wait for minimum of 1 µs or cclk_in period, whichever is greater
6. After a minimum of 1 µs, the application should program a value of 0 into the CARD_RESET register. This de-asserts the RST_n signal and takes the card out of reset.
7. The application can program a new CMD only after a minimum of 200 µs after the de-assertion of the RST_n signal, as per the MMC 4.41 standard.
   Note: For backward compatibility, the RST_n signal is temporarily disabled in the card by default. The host may need to set the signal as either permanently enabled or permanently disabled before it uses the card.

11.6 Interface description

Table 11-7EMMC Interface Description

Module pin	Dir.	Pad name	IOMUX
EMMC Interface			
pwren	O	IO_EMMCpwren_I2S1Bclk_GPIO0a0	GRF_GPIO0A_IOMUX[1:0]=2'b01
rstn	O	IO_EMMCrstn_GPIO1b3	GRF_GPIO1B_IOMUX[7:6]=2'b01
cclk	O	IO_EMMCclk_I2S1Blrck_UART2Ctx_GPIO0a1	GRF_GPIO0A_IOMUX[3:2]=2'b01

ccmd	I/O	IO_EMMCcmd_I2S1Bsclk_UART2Crx_GPIO0a2	GRF_GPIO0A_IOMUX[5:4]=2'b01
Cdata0	I/O	IO_EMMCd0_I2S1Bsdo_UART2Ccts_GPIO0a3	GRF_GPIO0A_IOMUX[7:6]=2'b01
Cdata1	I/O	IO_EMMCd1_I2S1Bsdi_UART2Crts_GPIO0a4	GRF_GPIO0A_IOMUX[9:8]=2'b01
Cdata2	I/O	IO_EMMCd2_SFCd3_I2C0Csda_GPIO0a5	GRF_GPIO0A_IOMUX[11:10]=2'b01
Cdata3	I/O	IO_EMMCd3_SFCd2_I2C0Cscl_GPIO0a6	GRF_GPIO0A_IOMUX[13:12]=2'b01
Cdata4	I/O	IO_EMMCd4_SFCd1_GPIO0a7	GRF_GPIO0A_IOMUX[15:14]=2'b01
Cdata5	I/O	IO_EMMCd5_SFCd0_JTG1tdi_GPIO0b0	GRF_GPIO0B_IOMUX[1:0]=2'b01
Cdata6	I/O	IO_EMMCd6_SFCclk_JTG1tdo_GPIO0b1	GRF_GPIO0B_IOMUX[3:2]=2'b01
Cdata7	I/O	IO_EMMCd7_SFCcs_JTG1trst_GPIO0b2	GRF_GPIO0B_IOMUX[5:4]=2'b01

Chapter 12 Embedded SRAM

12.1 Overview

There're five embedded SRAMs for instruction and data storage. Dynamic configurable response cycle is supported to meet high frequency requirement.
Byte/HALF WORD/WORD transactions are also supported.
ALL the SRAMs consist of several 8192X32(32KB) sub blocks and the clock of each 32KB block can be disabled to save power when the corresponding address area is not used.CRU_CLK3~7_CON contains the fields for embedded SRAM clock gating.

Table 12-1embedded SRAM list

RAM	***ADDRESS***	***POWERDOMAIN***
SYSRAM0	0x0300_0000~0x0304_ffff(320KB)	PD_LOGIC
SYSRAM1	0x0305_0000~0x0308_ffff(256KB)	PD_LOGIC
HIGHRAM0	0x0100_0000~0x0101_ffff(128KB)	PD_HIGH
HIGHRAM1	0x0102_0000~0x0105_ffff(256KB)	PD_HIGH
PMUSRAM	0x0309_0000~0x0309_ffff(64KB)	PD_PMU

12.2 Block Diagram

- AHB2SRAM: convert AHB access to SRAM access.
- MBIST_TOP: MBIST logic
- SPRA8192X32_n: SMIC55 single port SRAM instance with 8192 depth and 32 bits width.

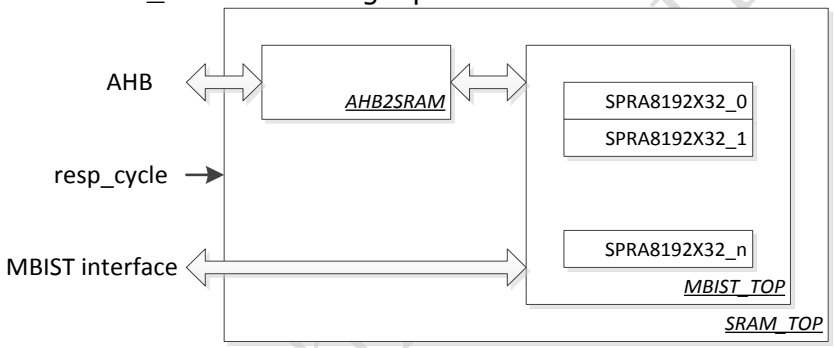


Figure 1.2-1 embedded SRAM architecture

12.3 Application Notes

12.3.1 Response cycle configuration

The maximum frequency of the five SRAMs is about 150 MHz by default. User should enable the "one more response cycle" feature by configuring the corresponding bit in GRF_INTER_CON0 to 1 when SRAM works at the frequency of about 300 MHz.

Chapter 13 I2S

13.1 Overview

The I2S/PCM controller is designed for interfacing between the AHB 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. Now it is widely used by many semiconductor manufacturers.
I2S bus is widely used in the devices such as 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.

13.1.1 Features

Not only I2S but also PCM mode stereo audio output and input are supported in I2S/PCM1/2 controller.

- Support two internal 32-bit wide and 32-location deep FIFOs, one for transmitting and the other for receiving audio data
- Support AHB bus interface
- Support 16 ~ 32 bits audio data transfer
- Support master and slave mode
- Support DMA handshaking interface and configurable DMA water level
- Support transmit FIFO empty, underflow, receive FIFO full, overflow interrupt and all interrupts can be masked
- Support configurable water level of transmit FIFO empty and receive FIFO full interrupt
- Support combined interrupt output
- Support 2-channel audio transmitting in I2S mode and PCM mode
- Support 2-channel audio receiving in I2S and PCM mode
- Support up to 192kHz sample rate
- Support I2S normal, left and right justified mode serial audio data transfer
- Support PCM early, late1, late2, late3 mode serial audio data transfer
- Support MSB or LSB first serial audio data transfer
- Support 16 to 31 bit audio data left or right justified in 32-bit wide FIFO
- Support two 16-bit audio data store together in one 32-bit wide location
- Support 2 independent LRCK signals, one for receiving and the other for transmitting audio data
- Support configurable SCLK and LRCK polarity
- Support SCLK is equivalent to MCLK divided by an even number range from 2 to 64 in master mode
- I2S0 / I2S1 are in peripheral sub-system

13.2 Block Diagram

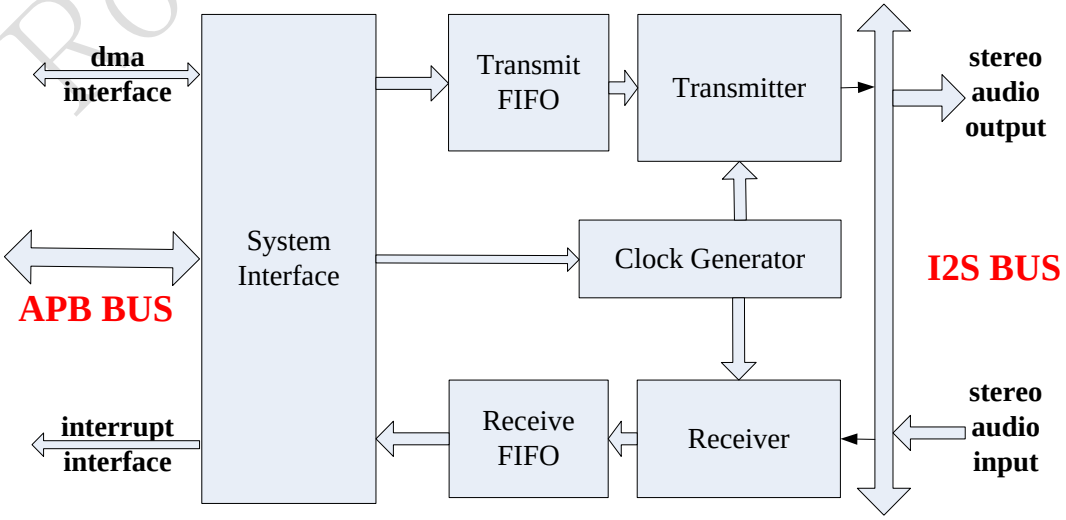


Fig 13-1 I2S/PCM1/2 controller (2 channel) Block Diagram

System Interface
The system interface implements the AHB slave operation. It contains not only control registers of transmitters and receiver inside but also interrupt and DMA handshaking interface.
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.
Transmitters
The Transmitters implement transmission operation. The transmitters can act as either a master or a slave, with I2S or PCM mode surround (up to 7.1 channel) serial audio interface.
Receiver
The Receiver implements receive operation. The receiver can act as either a master or a slave, with I2S or PCM mode stereo serial audio interface.
Transmit FIFO
The Transmit FIFO is the buffer to store transmitted audio data. The size of the FIFO is 32bits x 32.
Receive FIFO
The Receive FIFO is the buffer to store received audio data. The size of the FIFO is 32bits x 32.

13.3 Function description

In the I2S/PCM1/2 controller, there are four types: transmitter-master &receiver-master; transmitter-master & receiver-slave; transmitter-slave & receiver-master; transmitter-slave & receiver-slave.
In broadcasting application, the I2S/PCM1/2 controller is used as a transmitter and external or internal audio CODEC is used as a receiver. In recording application, the I2S/PCM1/2 controller is used as a receiver and external or internal audio CODEC is used as a transmitter. Either the I2S/PCM1/2 controller or the audio CODEC can act as a master or a slave, but if one is master, the other must be slave.

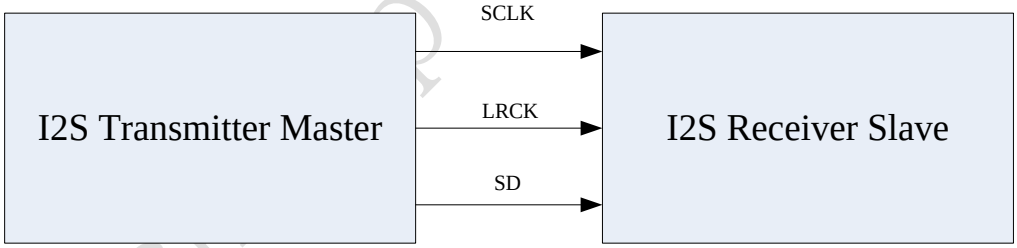


Fig 13-2 I2S transmitter-master & receiver-slave condition

When the transmitter acts as a master, it sends all signals to the receiver (the slave), and CPU controls when to send clock and data to the receiver. When acts as a slave, SD signal still goes from transmitter to receiver, but SCLK and LRCK signals are from the receiver (the master) to the 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 the transmitter to send data.

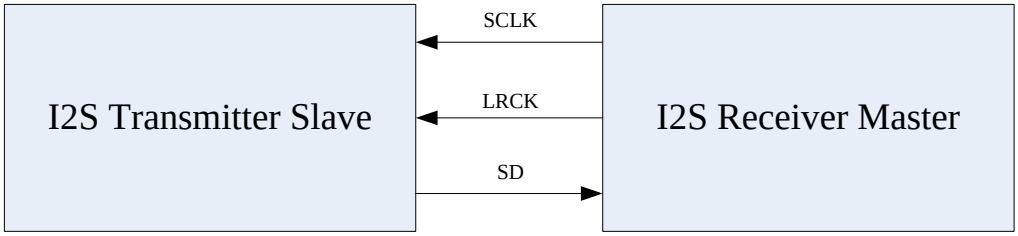


Fig 13-3 I2S transmitter-slave & receiver-master condition

When the receiver acts as a master, it sends SCLK and LRCK signals to the transmitter (the 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.
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.

13.3.1 I2S normal mode

This is the waveform of I2S normal mode. For LRCK (i2s1_lrck_rx/i2s1_lrck_tx) signal, it goes low to indicate left channel and high to right channel. For SD (i2s1_sdo, i2s1_sdi) signal, it starts sending the first bit (MSB or LSB) one SCLK clock cycle after LRCK changes. The range of SD signal width is from 16 to 32bits.

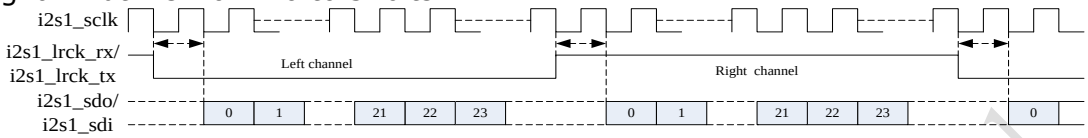


Fig 13-4 I2S normal mode timing format

13.3.2 I2S left justified mode

This is the waveform of I2S left justified mode. For LRCK (i2s1_lrck_rx / i2s1_lrck_tx) signal, it goes high to indicate left channel and low to right channel. For SD (i2s1_sdo, i2s1_sdi) signal, it starts sending the first bit (MSB or LSB) at the same time when LRCK changes. The range of SD signal width is from 16 to 32bits.

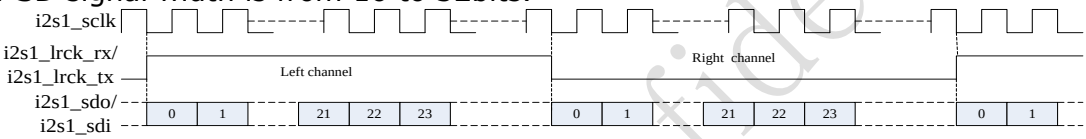


Fig 13-5 I2S left justified mode timing format

13.3.3 I2S right justified mode

This is the waveform of I2S right justified mode. For LRCK (i2s1_lrck_rx/ i2s1_lrck_tx) signal, it goes high to indicate left channel and low to right channel. For SD (i2s1_sdo, i2s1_sdi) signal, it transfers MSB or LSB first; but what is different from I2S normal or left justified mode, the last bit of the transferred data is aligned to the transition edge of the LRCK signal while one bit is transferred at one SCLK cycle. The range of SD signal width is from 16 to 32bits.

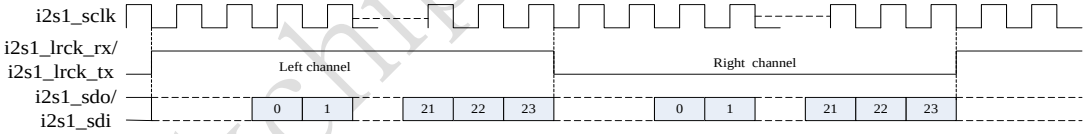


Fig 13-6 I2S right justified mode timing format

13.3.4 PCM early mode

This is the waveform of PCM early mode. For LRCK (i2s1_lrck_rx/i2s1_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s1_sdo, i2s1_sdi) signal, it sends the first bit (MSBor LSB) at the same time when LRCK goes high. The range of SD signal width is from 16 to 32bits.

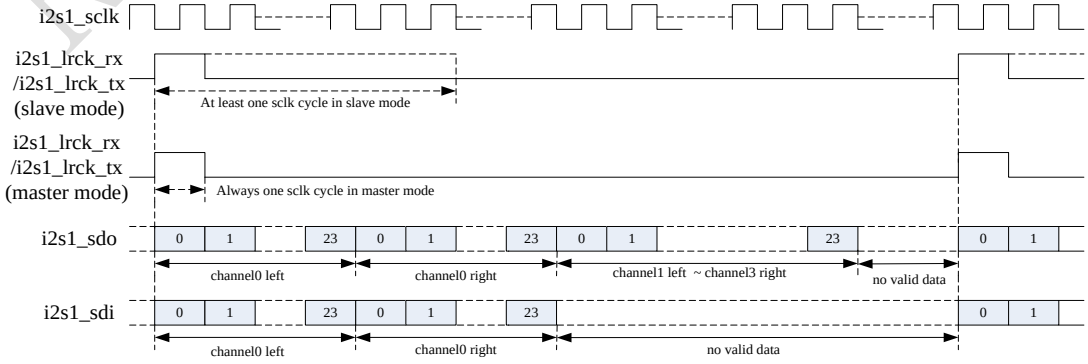


Fig 13-7 PCM early mode timing format

13.3.5 PCM late1 mode

This is the waveform of PCM early mode. For LRCK (i2s1_lrck_rx/i2s1_lrck_tx) signal, it goes

high to indicate the start of a group of audio channels. For SD (i2s1_sdo, i2s1_sdi) signal, it sends the first bit (MSB or LSB) one SCLK clock cycle after LRCK goes high. The range of SD signal width is from 16 to 32bits.

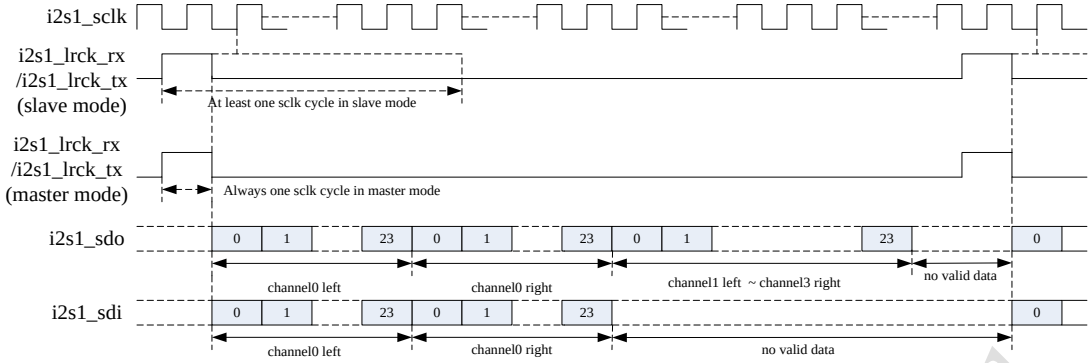


Fig 13-8 PCM late1 mode timing format

13.3.6 PCM late2 mode

This is the waveform of PCM early mode. For LRCK (i2s1_lrck_rx/i2s1_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s1_sdo, i2s1_sdi) signal, it sends the first bit (MSB or LSB)two SCLK clock cycles after LRCK goes high. The range of SD signal width is from 16 to 32bits.

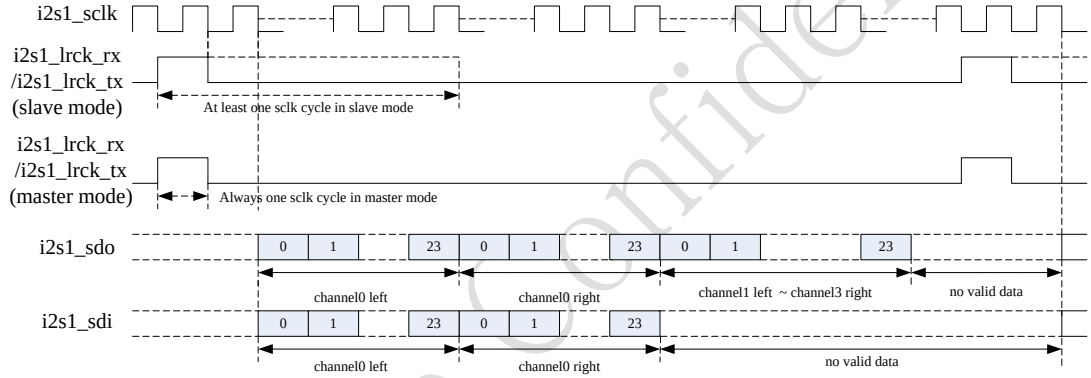


Fig 13-9 PCM late2 mode timing format

13.3.7 PCM late3 mode

This is the waveform of PCM early mode. For LRCK (i2s1_lrck_rx/i2s1_lrck_tx) signal, it goes high to indicate the start of a group of audio channels. For SD (i2s1_sdo, i2s1_sdi) signal, it sends the first bit (MSB or LSB) three SCLK clock cycles after LRCK goes high. The range of SD signal width is from 16 to 32bits.

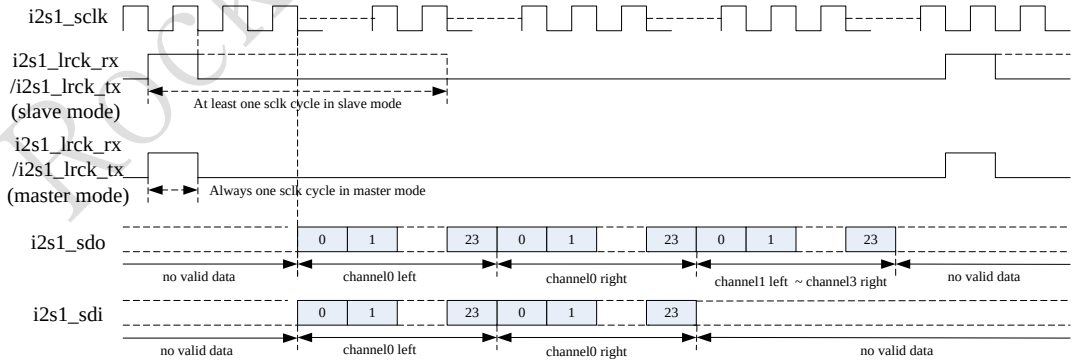


Fig 13-10 PCM late3 mode timing format

13.4 Register description

13.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
I2Sx_TXCR	0x0000	W	0x0000000f	transmit operation control register

Name	Offset	Size	Reset Value	Description
I2Sx_RXCR	0x0004	W	0x0000000f	receive operation control register
I2Sx_CKR	0x0008	W	0x00071f1f	clock generation register
I2Sx_FIFOLR	0x000c	W	0x00000000	FIFO level register
I2Sx_DMACR	0x0010	W	0x001f0000	DMA control register
I2Sx_INTCR	0x0014	W	0x00000000	interrupt control register
I2Sx_INTSR	0x0018	W	0x00000000	interrupt status register
I2Sx_XFER	0x001c	W	0x00000000	Transfer Start Register
I2Sx_CLR	0x0020	W	0x00000000	SCLK domain logic clear Register
I2Sx_TXDR	0x0400 ~0x7FC	W	0x00000000	Transmit FIFO Data Register
I2Sx_RXDR	0x0800 ~0xBFC	W	0x00000000	Receive FIFO Data Register

*Notes: B- Byte (8 bits) access, **HW**- Half WORD (16 bits) access, **W**-WORD (32 bits) access, x=1,2*

13.4.2 Detail Register Description

I2Sx_TXCR

Address: Operational Base + offset (0x0000)
transmit operation control register

Bit	Attr	Reset Value	Description
31:23	RO	0x0	reserved
22:17	RW	0x00	RCNT right justified counter (Can be written only when XFER[0] bit is 0.) Only valid in I2S Right justified format and slave tx mode is selected. Start to transmit data RCNT sclk cycles after left channel valid.
16:15	RW	0x0	CSR Channel select register Must be 2'b00.
14	RW	0x0	HWT Halfword word transform (Can be written only when XFER[0] bit is 0.) Only valid when VDW select 16bit data. 0:32 bit data valid from AHB/APB bus. Low 16 bit for left channel and high 16 bit for right channel. 1:low 16bit data valid from AHB/APB bus, high 16 bit data invalid.
13	RO	0x0	reserved

Bit	Attr	Reset Value	Description
12	RW	0x0	SJM Store justified mode (Can be written only when XFER[0] bit is 0.) 16bit~31bit DATA stored in 32 bits width fifo. If VDW select 16bit data, this bit is valid only when HWT select 0.Because if HWT is 1, every fifo unit contain two 16bit data and 32 bit space is full, it is impossible to choose justified mode. 0:right justified 1:left justified
11	RW	0x0	FBM First Bit Mode (Can be written only when XFER[0] bit is 0.) 0:MSB 1:LSB
10:9	RW	0x0	IBM I2S bus mode (Can be written only when XFER[0] bit is 0.) 0:I2S normal 1:I2S Left justified 2:I2S Right justified 3:reserved
8:7	RW	0x0	PBM PCM bus mode (Can be written only when XFER[0] bit is 0.) 0:PCM no delay mode 1:PCM delay 1 mode 2:PCM delay 2 mode 3:PCM delay 3 mode
6	RO	0x0	reserved
5	RW	0x0	TFS Transfer format select (Can be written only when XFER[0] bit is 0.) 0: I2S format 1: PCM format

Bit	Attr	Reset Value	Description
4:0	RW	0x0f	VDW Valid Data width (Can be written only when XFER[0] bit is 0.) 0~14:reserved 15:16bit 16:17bit 17:18bit 18:19bit …… n:(n+1)bit …… 28:29bit 29:30bit 30:31bit 31:32bit

I2Sx_RXCR
Address: Operational Base + offset (0x0004)
receive operation control register

Bit	Attr	Reset Value	Description
31:15	RO	0x0	reserved
14	RW	0x0	HWT Halfword word transform (Can be written only when XFER[1] bit is 0.) Only valid when VDW select 16bit data. 0:32 bit data valid to AHB/APB bus. Low 16 bit for left channel and high 16 bit for right channel. 1:low 16bit data valid to AHB/APB bus, high 16 bit data invalid.
13	RO	0x0	reserved
12	RW	0x0	SJM Store justified mode (Can be written only when XFER[1] bit is 0.) 16bit~31bit DATA stored in 32 bits width fifo. If VDW select 16bit data, this bit is valid only when HWT select 0.Because if HWT is 1, every fifo unit contain two 16bit data and 32 bit space is full, it is impossible to choose justified mode. 0:right justified 1:left justified

Bit	Attr	Reset Value	Description
11	RW	0x0	FBM First Bit Mode (Can be written only when XFER[1] bit is 0.) 0:MSB 1:LSB
10:9	RW	0x0	IBM I2S bus mode (Can be written only when XFER[1] bit is 0.) 0:I2S normal 1:I2S Left justified 2:I2S Right justified 3:reserved
8:7	RW	0x0	PBM PCM bus mode (Can be written only when XFER[1] bit is 0.) 0:PCM no delay mode 1:PCM delay 1 mode 2:PCM delay 2 mode 3:PCM delay 3 mode
6	RO	0x0	reserved
5	RW	0x0	TFS Transfer format select (Can be written only when XFER[1] bit is 0.) 0:i2s 1:pcm
4:0	RW	0x0f	VDW Valid Data width (Can be written only when XFER[1] bit is 0.) 0~14:reserved 15:16bit 16:17bit 17:18bit 18:19bit …… n:(n+1)bit …… 28:29bit 29:30bit 30:31bit 31:32bit

I2Sx_CKR

Address: Operational Base + offset (0x0008)

clock generation register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:28	RO	0x0	reserved
27	RW	0x0	MSS Master/slave mode select (Can be written only when XFER[1] or XFER[0] bit is 0.) 0:master mode(sclk output) 1:slave mode(sclk input)
26	RW	0x0	CKP Sclk polarity (Can be written only when XFER[1] or XFER[0] bit is 0.) 0: sample data at posedgesclk and drive data at negedgesclk 1: sample data at negedgesclk and drive data at posedgesclk
25	RW	0x0	RLP Receive lrck polarity (Can be written only when XFER[1] or XFER[0] bit is 0.) 0:normal polarity (I2S normal: low for left channel, high for right channel I2S left/right just: high for left channel, low for right channel PCM start signal: high valid) 1:oppsite polarity (I2S normal: high for left channel, low for right channel I2S left/right just: low for left channel, high for right channel PCM start signal: low valid)

Bit	Attr	Reset Value	Description
24	RW	0x0	TLP Transmit lrck polarity (Can be written only when XFER[1] or XFER[0] bit is 0.) 0:normal polarity (I2S normal: low for left channel, high for right channel I2S left/right just: high for left channel, low for right channel PCM start signal: high valid) 1:oppsite polarity (I2S normal: high for left channel, low for right channel I2S left/right just: low for left channel, high for right channel PCM start signal: low valid)
23:16	RW	0x07	MDIV mclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.) Serial Clock Divider = Fmclk / Ftxsclk-1.(mclk frequecy / txsclk frequecy-1) 0 :Fmclk=Ftxsclk; 1 :Fmclk=2*Ftxsclk; 2,3 :Fmclk=4*Ftxsclk; 4,5 :Fmclk=6*Ftxsclk; …… 2n,2n+1:Fmclk=(2n+2)*Ftxsclk; …… 60,61:Fmclk=62*Ftxsclk; 62,63:Fmclk=64*Ftxsclk; …… 252,253:Fmclk=254*Ftxsclk; 254,255:Fmclk=256*Ftxsclk;

Bit	Attr	Reset Value	Description
15:8	RW	0x1f	RSD Receive sclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.) Receive sclk divider= Fsclk/Frxlrck 0~30:reserved 31: 32fs 32: 33fs 33: 34fs 34: 35fs …… n: (n+1)fs …… 253: 254fs 254: 255fs 255: 256fs
7:0	RW	0x1f	TSD Transmit sclk divider (Can be written only when XFER[1] or XFER[0] bit is 0.) Transmit sclk divider=Ftxsclk/Ftxlrck 0~30:reserved 31: 32fs 32: 33fs 33: 34fs 34: 35fs …… n: (n+1)fs …… 253: 254fs 254: 255fs 255: 256fs

I2Sx_FIFOLR

Address: Operational Base + offset (0x000c)

FIFO level register

Bit	Attr	Reset Value	Description
31:30	RO	0x0	reserved
29:24	RO	0x00	RFL Receive FIFO Level Contains the number of valid data entries in the receive FIFO.
23:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5:0	RO	0x00	TFL Transmit FIFO Level Contains the number of valid data entries in the transmit FIFO0.

I2Sx_DMACR
Address: Operational Base + offset (0x0010)
DMA control register

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24	RW	0x0	RDE Receive DMA Enable 0 : Receive DMA disabled 1 : Receive DMA enabled
23:21	RO	0x0	reserved
20:16	RW	0x1f	RDL 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.
15:9	RO	0x0	reserved
8	RW	0x0	TDE Transmit DMA Enable 0 : Transmit DMA disabled 1 : Transmit DMA enabled
7:5	RO	0x0	reserved
4:0	RW	0x00	TDL 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 TXFIFO(TXFIFO0 if CSR=00;TXFIFO1 if CSR=01,TXFIFO2 if CSR=10,TXFIFO3 if CSR=11)is equal to or below this field value.

I2Sx_INTCR
Address: Operational Base + offset (0x0014)
interrupt control register

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved

Bit	Attr	Reset Value	Description
24:20	RW	0x00	RFT Receive FIFO Threshold When the number of receive FIFO entries is more than or equal to this threshold plus 1, the receive FIFO full interrupt is triggered.
19	RO	0x0	reserved
18	WO	0x0	RXOIC RX overrun interrupt clear Write 1 to clear RX overrun interrupt.
17	RW	0x0	RXOIE RX overrun interrupt enable 0:disable 1:enable
16	RW	0x0	RXFIE RX full interrupt enable 0:disable 1:enable
15:9	RO	0x0	reserved
8:4	RW	0x00	TFT Transmit FIFO Threshold When the number of transmit FIFO (TXFIFO0 if CSR=00; TXFIFO1 if CSR=01, TXFIFO2 if CSR=10, TXFIFO3 if CSR=11) entries is less than or equal to this threshold, the transmit FIFO empty interrupt is triggered.
3	RO	0x0	reserved
2	WO	0x0	TXUIC TX underrun interrupt clear Write 1 to clear TX underrun interrupt.
1	RW	0x0	TXUIE TX underrun interrupt enable 0:disable 1:enable
0	RW	0x0	TXEIE TX empty interrupt enable 0:disable 1:enable

I2Sx_INTSR
Address: Operational Base + offset (0x0018)
interrupt status register

Bit	Attr	Reset Value	Description
31:18	RO	0x0	reserved

Bit	Attr	Reset Value	Description
17	RO	0x0	RXOI RX overrun interrupt 0:inactive 1:active
16	RO	0x0	RXFI RX full interrupt 0:inactive 1:active
15:2	RO	0x0	reserved
1	RO	0x0	TXUI TX underrun interrupt 0:inactive 1:active
0	RO	0x0	TXEI TX empty interrupt 0:inactive 1:active

I2Sx_XFER
Address: Operational Base + offset (0x001c)
Transfer Start Register

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	synchronize TX and RX LRCK 0:disable 1:enable
1	RW	0x0	RXS RX Transfer start bit 0:stop RX transfer 1:start RX transfer
0	RW	0x0	TXS TX Transfer start bit 0:stop TX transfer 1:start TX transfer

I2Sx_CLR
Address: Operational Base + offset (0x0020)
SCLK domain logic clear Register

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	RXC RX logic clear This is a self cleared bit. Write 1 to clear all receive logic.

Bit	Attr	Reset Value	Description
0	RW	0x0	TXC TX logic clear This is a self cleared bit. Write 1 to clear all transmit logic.

I2Sx_TXDR
Address: Operational Base + offset (0x0024)
Transmit FIFO Data Register

Bit	Attr	Reset Value	Description
31:0	WO	0x00000000	TXDR Transmit FIFO Data Register When it is written to, data are moved into the transmit FIFO.

I2Sx_RXDR
Address: Operational Base + offset (0x0028)
Receive FIFO Data Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	RXDR Receive FIFO Data Register When the register is read, data in the receive FIFO is accessed.

13.5 Interface description

I2S0 has 1 IOMUX and 1 connection to the internal ACODEC, which is controlled by GRF_IOMUX_CON[12].

When GRF_IOMUX_CON[12] is 1'b0, the IOMUX is as follow.

Module Pin	Direction	Pad Name	IOMUX Setting
I2S0			
i2s0_clk	O	IO_LCDcsn_I2S0clk_EBCgdoe_GPIO0b7	GRF_GPIO0B_IOMUX[15:14]=2'b10
i2s0_sclk	I/O	IO_UART2Arx_I2S0sclk_EBCgdclk_GPIO0b5	GRF_GPIO0B_IOMUX[11:10]=2'b10
i2s0_lrck	I/O	IO_UART2Atx_I2S0lrck_EBCgdsp_GPIO0b6	GRF_GPIO0B_IOMUX[13:12]=2'b10
i2s0_sdo	O	IO_UART2Acts_I2S0sdo_EBCsdclk_GPIO0b4	GRF_GPIO0B_IOMUX[9:8]=2'b10
i2s0_sdi	I	IO_UART2Arts_I2S0sdi_EBCvcom_GPIO0b3	GRF_GPIO0B_IOMUX[7:6]=2'b10

When GRF_IOMUX_CON[12] is 1'b1, I2S0is connected to the internal ACODEC.

I2S1 has 2 IOMUX, which is controlled by GRF_IOMUX_CON[13].

When GRF_IOMUX_CON[13] is 1'b0, the IOMUX is as follow.

Module Pin	Direction	Pad Name	IOMUX Setting

I2S1			
i2s1_clk	O	IO_I2S1Aclk_EBCgdrl_GPIO1a0	GRF_GPIO1A_IOMUX[1:0]=2'b01
i2s1_sclk	I/O	IO_I2S1Asclk_UART1Btx_EBCsdce2_GPIO1a2	GRF_GPIO1A_IOMUX[5:4]=2'b01
i2s1_lrck	I/O	IO_I2S1Alrck_UART1Brx_EBCsdshr_GPIO1a1	GRF_GPIO1A_IOMUX[3:2]=2'b01
i2s1_sdo	O	IO_I2S1Asdo_UART1Bcts_EBCsdce1_GPIO1a3	GRF_GPIO1A_IOMUX[7:6]=2'b01
i2s1_sdi	I	IO_I2S1Asdi_UART1Brts_EBCsdce4_GPIO1a4	GRF_GPIO1A_IOMUX[9:8]=2'b01

When GRF_IOMUX_CON[13] is 1'b1, the IOMUX is as follow.

Module Pin	Direction	Pad Name	IOMUX Setting
I2S1			
i2s1_clk	O	IO_EMMCpwren_I2S1Bclk_GPIO0a0	GRF_GPIO0A_IOMUX[1:0]=2'b10
i2s1_sclk	I/O	IO_EMMCcmd_I2S1Bsclk_UART2Crx_GPIO0a2	GRF_GPIO0A_IOMUX[5:4]=2'b10
i2s1_lrck	I/O	IO_EMMCclk_I2S1Blrck_UART2Ctx_GPIO0a1	GRF_GPIO0A_IOMUX[3:2]=2'b10
i2s1_sdo	O	IO_EMMCd0_I2S1Bsdo_UART2Ccts_GPIO0a3	GRF_GPIO0A_IOMUX[7:6]=2'b10
i2s1_sdi	I	IO_EMMCd1_I2S1Bsdi_UART2Crts_GPIO0a4	GRF_GPIO0A_IOMUX[9:8]=2'b10

13.6 Application Notes

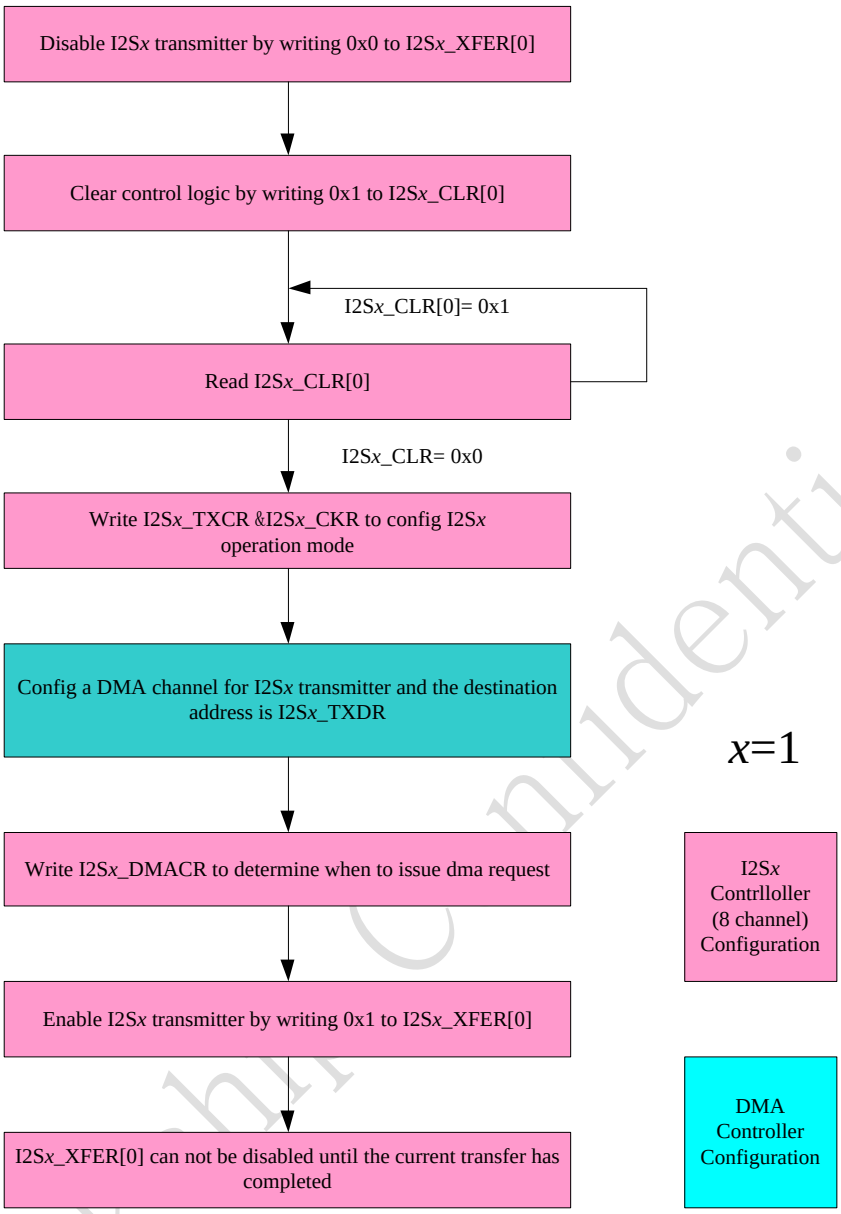


Fig 13-11 I2S/PCM1/2 controller transmit operation flow chart

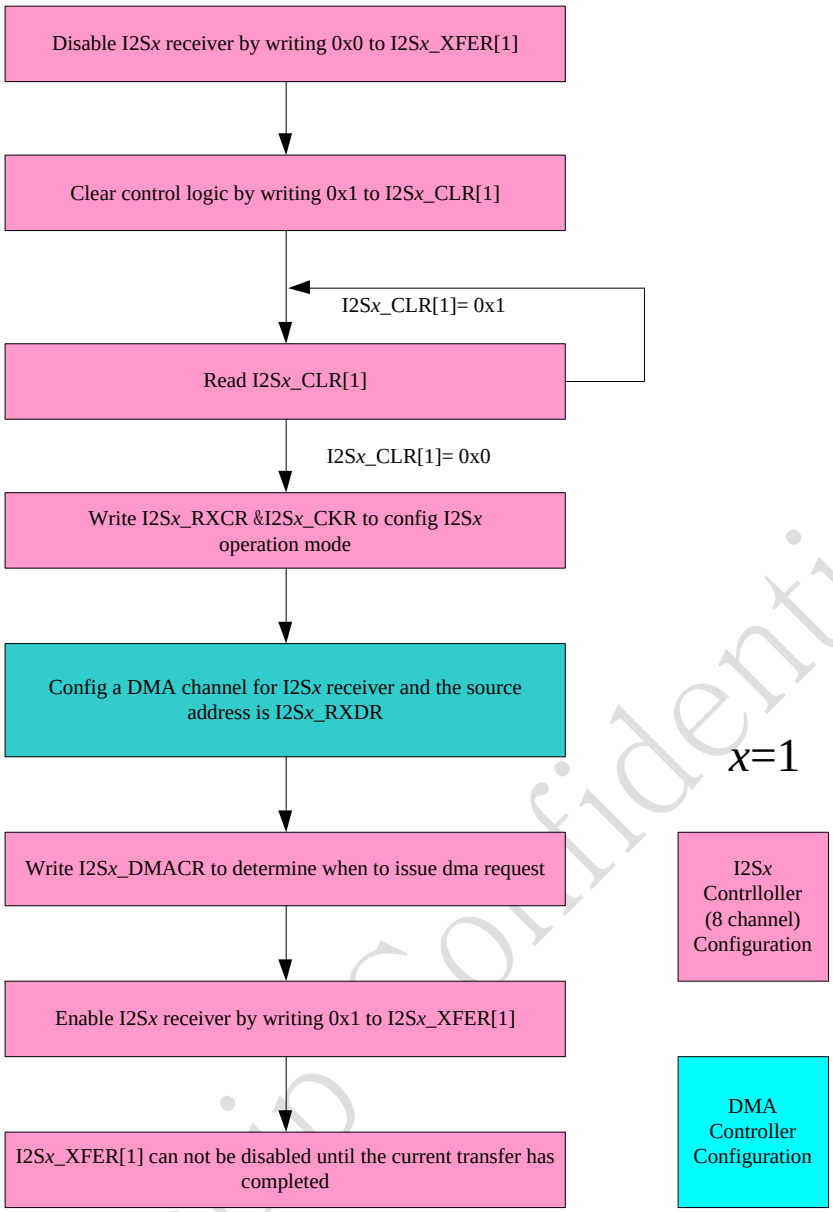


Fig 13-12 I2S/PCM1/2 controller receive operation flow chart

Chapter 14 USB OTG

14.1 Overview

This USB OTG Controller is compliant withUSB2.0 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 connection to device) and multi-point applications to devices.

14.1.1 Features

- Non-HNP- and Non-SRP-Capable OTG compliant with the USB2.0 Specification.
- AHB slave port only, has no internal DMA.
- Operates in High-Speed and Full-Speed mode.
- Supports UTMI+ Level 3 with 16bit data bus.
- One USB root hub for HOST mode.
- 3 Device mode endpoints in addition to control endpoint 0.
- 1 Device mode periodic IN endpoint up to 64 bytes.
- 4 Host mode channels.
- 308 x 35bits total data FIFO.

14.2 Block Diagram

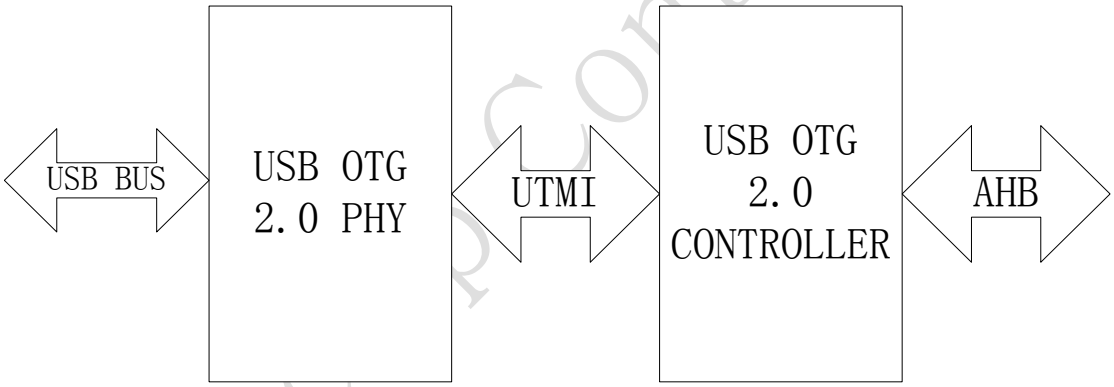


Fig 14-1 USB OTG 2.0 Architecture

The figure above shows the architecture of USB OTG 2.0. It is broken up into two separate units: USB OTG 2.0 controller and USB OTG 2.0 PHY. The two units are interconnected with UTMI interface.

14.2.1 USB OTG 2.0 Controller Function

The USB OTG 2.0 Controller consists of SIE(Serial Interface Engine) logic, the endpoint logic, the channel logic and the internal DMA logic.

The SIE logic contains the USB PID and address recognition logic, and other sequencing and state machine logic to handle USB packets and transactions. Generally the SIE Logic is required for any USB implementation while the number and types of endpoints will vary as function of application and performance requirements.

The endpoint logic contains the endpoint specific logic: endpoint number recognition, FIFOs and FIFO control, etc.

The channel Logic contains the channel tasks schedule, FIFOs and FIFO control, etc.

14.2.2 USB OTG 2.0 PHY Function

The USB OTG 2.0 PHY handles the low level USB protocol and signaling. This includes features such as; data serialization and deserialization, bit stuffing and clock recovery and synchronization. The primary focus of this block is to shift the clock domain of the data from the USB 2.0 rate to the frequency of UTMI clock which is 30MHz.

14.2.3 UTMI Interface

- Transmit

Transmit must be asserted to enable any transmissions.
The USB OTG2.0 CONTROLLER asserts TXValid to begin a transmission and negates TXValid to end a transmission. After the USB OTG2.0 CONTROLLER asserts TXValid it can assume that the transmission has started when it detect sTXReady asserted.
The USB OTG2.0 CONTROLLER assumes that the USB OTG2.0 PHY has consumed a data byte if TXReady and TXValid are asserted.
The USB OTG2.0 CONTROLLER must have valid packet information (PID) asserted on the DataIn bus coincident with the assertion of TXValid. Depending on the USB OTG2.0 PHY implementation, TXReady may be asserted by the Transmit State Machine as soon as one CLK after the assertion of TXValid. TXValid and TXReady are sampled on the rising edge of CLK.
The Transmit State Machine does NOT automatically generate Packet ID's (PIDs) or CRC.
When transmitting, the USB OTG2.0 CONTROLLER is always expected to present a PID as the first byte of the data stream and if appropriate, CRC as the last bytes of the data stream.
The USB OTG2.0 CONTROLLER must use LineState to verify a Bus Idle condition before asserting TXValid in the TX Waitstate.
The state of TXReady in the TX Wait and Send SYNC states is undefined. An MTU implementation may prepare for the next transmission immediately after the Send EOP state and assert TXReady inthe TX Wait state. An MTU implementation may also assert TXReady in the Send SYNC state. The first assertion of TXReady is Macrocell implementation dependent.
The USB OTG2.0 CONTROLLER must prepare DataIn for the first byte to be transmitted before asserting TXValid.

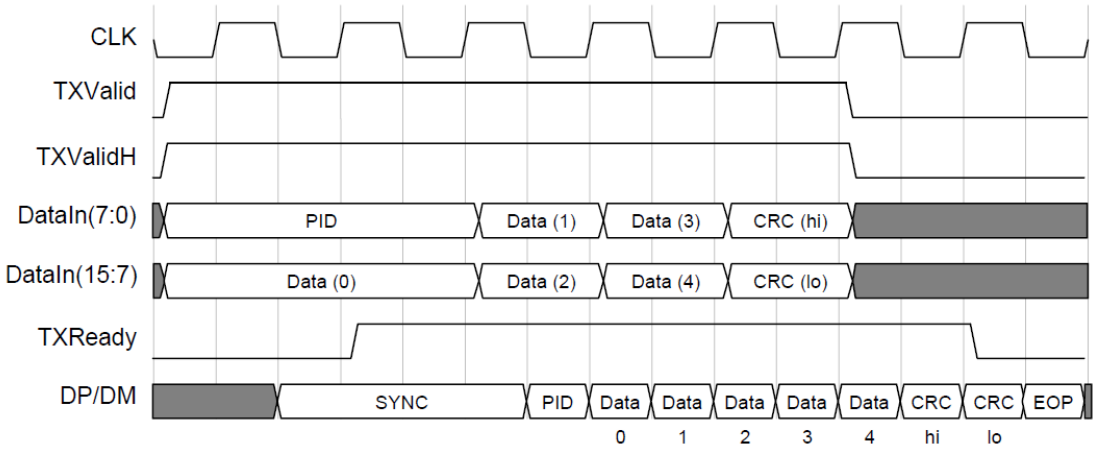


Fig 14-2 UTMI interface – Transmit timing for a data packet

- Receive

RXActive and RXValid are sampled on the rising edge of CLK.
In the RX Wait state the receiver is always looking for SYNC.
The USB OTG 2.0 PHY asserts RXActive when SYNC is detected (Strip SYNC state).
The USB OTG 2.0 PHY negates RXActive when an EOP is detected (Strip EOP state).
When RxActive is asserted, RXValid will be asserted if the RX Holding Register is full.
RXValid will be negated if the RX Holding Register was not loaded during the previous byte time.
This will occur if 8 stuffed bits have been accumulated.
The USB OTG2.0 Controller must be ready to consume a data byte if RXActive and RXValid are asserted (RX Data state).
In FS mode, if a bit stuff error is detected then the Receive State Machine will negate RXActive and RXValid, and return to the RXWait state.

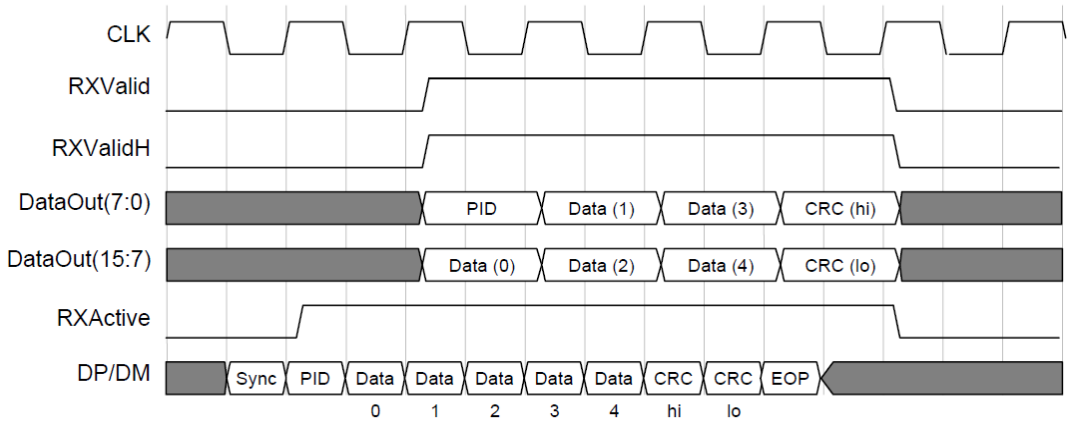


Fig 14-3 UTMI interface – Receive timing for a data packet

14.3 UART BYPASS FUNCITON

USB PADs IO_USB0PP and IO_USB0PN can be used as UART0_tx and UART0_rx when USB OTG PHY is configured in bypass mode.

Signal	CONNECT	I/O	Description
BYPASSDMDATA0	uart0_sout	I	Data for DM0 Transmitter Digital Bypass
BYPASSDMEN0	grf_uoc_con0[1]	I	DM0 Transmitter Digital Bypass Enable
BYPASSSEL0	grf_uoc_con0[2]	I	Transmitter Digital Bypass mode Enable
FSVPLUS0	uart0_sin	O	Single-Ended D- Indicator The controller signal indicates the state of the DP during normal operation or UART data reception
OTGDISABLE0	grf_uoc_con0[0]	I	1'b1: OTG0 disable; 1'b0: OTG0 normal mode
COMMONONN	uoc0_con0[0]	I	Common Block Power-Down Control This signal controls the power-down signals in PLL blocks when the USBPHY is in Suspend Mode. 1: PLL blocks are powered down. 0: PLL blocks remain powered This signal is a strapping option that must be set prior to a power-on reset and remain static during normal operation.

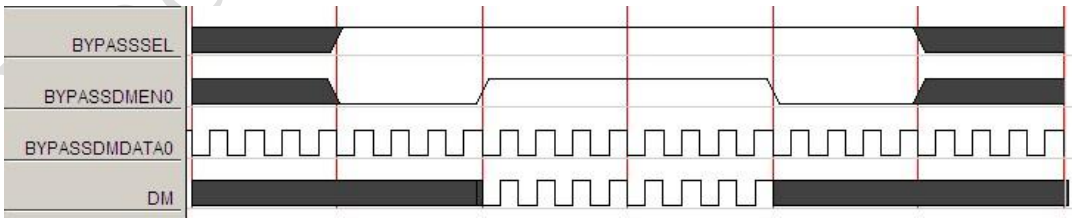


Fig 14-4 UART Timing Sequence

To use UART and Auto résumé functions:
1. Disable the OTG block by setting OTGDISABLE0 to 1'b1.
2. Disable the pull-up resistance on the D+ line by setting OPMODE0[1:0] to 2'b01.
3. To ensure that the XO, Bias, and PLL blocks are powered down in Suspend mode, set COMMONONN to 1'b1.
4. Place the USB PHY in Suspend mode by setting SUSPENDM0 to 1'b0.
5. Set BYPASSSEL0 to 1'b1.
6. To transmit data, controls BYPASSDMEN0, and BYPASSDMDATA0.
To receive data, monitor FSVPLUS0.

To return to normal operating mode:
1. Ensure that there is no activity on the USB.
2. Set BYPASSSEL0 to 1'b0.
3. Set SUSPENDM0 to 1’b1.Resume the USB PHY.
4. Set COMMONONN to 1’b0.
5. set OTGDISABLE0 to 1'b0.

Please reference “CH14 USB OTG.pdf” for detail.

Chapter 15 VOP

15.1 Overview

VOP is the display interface from Direct Memory Access (DMA) to MCU-LCD. It's connected to an AHB bus. After configuring by CPU, VOP is filled with pixel data transferred by DMA through AHB bus.

15.1.1 Features

- Bus interface
  - Support AMBA 2.0 AHB slave interface for accessing internal registers ， 32-bit data bus width
- Support source data format
  - RGB565
  - YUV420
- Support UV swap
- Support YUV2RGB
  - Support BT601 limited range
  - Support BT709 limited range
  - Support BT601 full range
- Support allegro dither down for RGB888 to RGB565
- Support RGB565 display data format
- Support display data swap
  - Half-word swap
  - Byte swap
- Support max output resolution400x400
- Built-in i8080 MCU interface
  - 8-bit data bus width
  - Timing configurable
  - Data direction control when idle
  - Write split transfer mode support

15.2 Block Diagram

The VOP architecture is shown in figure 1-1 .There are 4 data paths as shown below:

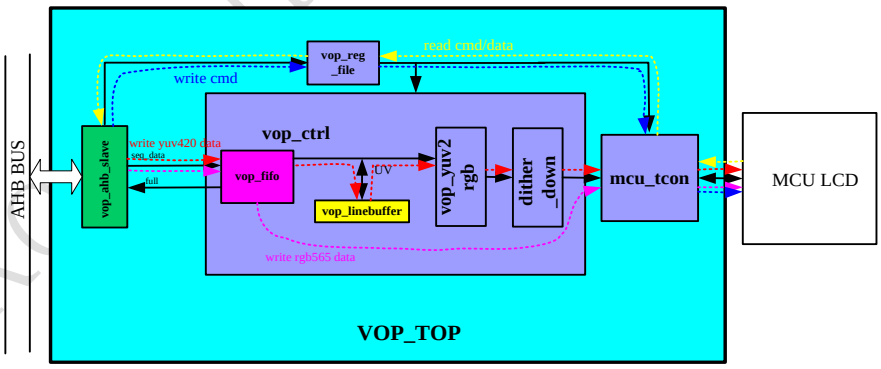


Fig 15-1 VOP Block Diagram

1. RGB format through register and command path(blue dotted line): in order to be compatible with NanoC, RGB data can go through the registers directly.
2. RGB format through FIFO (pink dotted line): RGB data go through to MCU_TCON.
3. YUV format path (red dotted line): UV data go through from FIFO to LineBuffer, and will be fetched out at the time with Y data from FIFO, after translating from YUV to RGB888, dither down, data will be sent out.
4. Read path (yellow dotted line): RGB data or command read path

15.3 Function Description

15.3.1 Pixel format

1. RGB

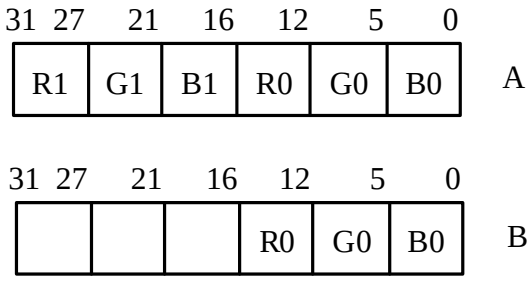


Fig 15-2 RGB565 data format

2. YUV

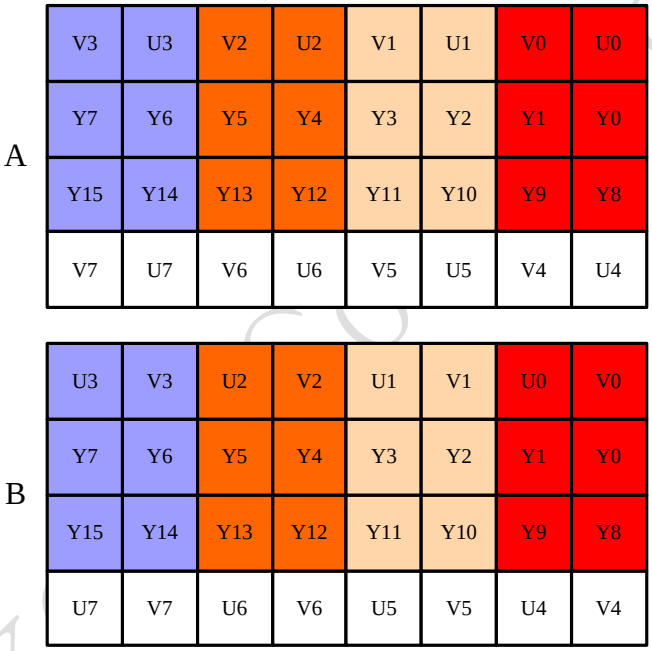


Fig 15-3 YUV420 data format

15.3.2 Pixel Data Path

Pixel data is passed from memory to VOP through DMA as shown in figure1-4. It's required that YUV420 data transferred from DMA is as shown in figure1-3.

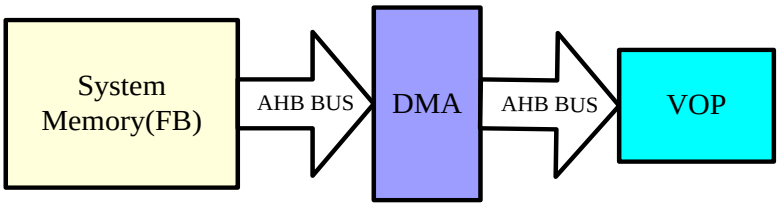


Fig 15-4 pixel data path

15.3.3 Display process

1. Color space conversion

Support three standards for YUV2RGB as shown below.

a) yuv to rgb (REC-601) range 0 (Y[16:235], UV[16:240], RGB[0:255])

R = 1.164(Y-16) + 1.596(V-128)

G = 1.164(Y-16) - 0.391(U-128) - 0.813(V-128)

B = 1.164(Y-16) + 2.018(U-128)

b) yuv to rgb (REC-601) range 1 (YUV[0:255], RGB[0:255])
R = (Y-16) + 1.402(V-128)
G = (Y-16) - 0.344(U-128) - 0.714(V-128)
B = (Y-16) + 1.772(U-128)

c) yuv to rgb (REC-709) range 0 (Y[16:235], UV[16:240], RGB[0:255])
R = 1.164(Y-16) + 1.793(V-128)
G = 1.164(Y-16) - 0.213(U-128) - 0.534(V-128)
B = 1.164(Y-16) + 2.115(U-128)

2. Allegro Dither Down

Dithering is an intentional applied form of noise, using to randomize quantization error, and thereby preventing large-scaling patterns such as "banding".
The pixel value is used by dithering process to display the data in a lower color depth on the LCD panel. The source input format is RGB888 and display output format is RGB565. When dithering is enabled (sw_mcu_dither_en), the output data is generated by dithering algorithm based on the pixel position and the value of removed bits. Otherwise, the MSBs of the pixel color components are outputted as display data as shown in figure1-5.

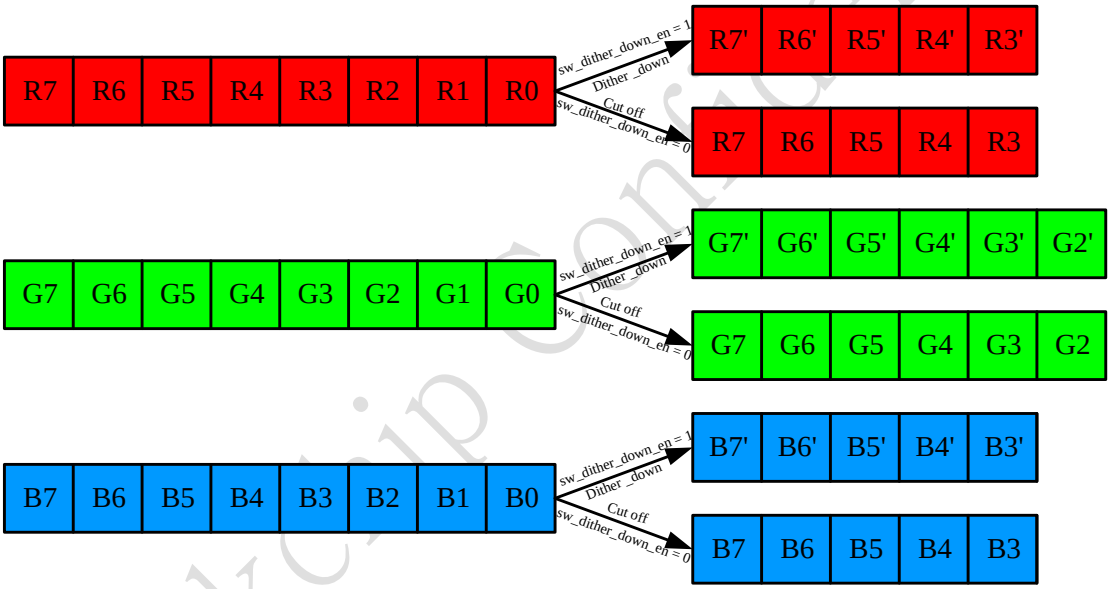


Fig 15-5 dither down block

15.3.4 I/O description

VOP have 11 outputs connected to pad, which are controlled by IOMUX as shown in Table 15-1.

Table 15-1VOP output pins

Module pin	Dir	Pad name	IOMUX
lcd_csn	O	IO_LCDcsn_I2S0clk_EBCgdoe_GPIO0b7	GRF_GPIO2B_IOMUX[15:14]=2'b01
lcd_wrn	O	IO_LCDwrn_I2C2Bscl_EBCsdle_GPIO0d0	GRF_GPIO0D_IOMUX[1:0]=2'b01
lcd_rs	O	IO_LCDrs_I2C2Bsda_EBCsdoe_GPIO0d1	GRF_GPIO0D_IOMUX[3:2]=2'b01
lcd_d0	O	IO_LCDd0_SPI0Atx_EBCsddo0_GPIO0c0	GRF_GPIO0C_IOMUX[1:0]=2'b01
lcd_d1	O	IO_LCDd1_SPI0Atx_EBCsddo1_GPIO0c1	GRF_GPIO0C_IOMUX[3:2]=2'b01
lcd_d2	O	IO_LCDd2_SPI0Aclk_EBCsddo2_GPIO0c	GRF_GPIO0C_IOMUX[5:4]=2'b01

		2	
lcd_d3	O	IO_LCDd3_SPI0Acs_EBCsddo3_GPIO0c3	GRF_GPIO0C_IOMUX[7:6]=2'b01
lcd_d4	O	IO_LCDd4_UART2Brx_EBCsddo4_GPIO0c4	GRF_GPIO0C_IOMUX[9:8]=2'b01
lcd_d5	O	IO_LCDd5_UART2Brx_EBCsddo5_GPIO0c5	GRF_GPIO0C_IOMUX[11:10]=2'b01
lcd_d6	O	IO_LCDd6_UART2Brts_EBCsddo6_GPIO0c6	GRF_GPIO0C_IOMUX[13:12]=2'b01
lcd_d7	O	IO_LCDd7_UART2Brts_EBCsddo7_GPIO0c7	GRF_GPIO0C_IOMUX[15:14]=2'b01

15.4 Register Description

15.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
VOP_MCU_CON	0x00000	W	0x00000000	Mode control
VOP_MCU_VERSION	0x00004	W	0x00000000	VOP Version
VOP_MCU_TIMING	0x00008	W	0x00000000	MCU timing control
VOP_MCU_LCD_SIZE	0x0000C	W	0x00000000	LCD size
VOP_MCU_FIFO_WATERMARK	0X00010	w	0X00000000	FIFO almost_full or almost_emptu watermark
VOP_MCU_SRT	0x00014	W	0x00000000	Soft reset
VOP_MCU_INT_EN	0X00018	W	0x00000000	Interrupt enable
VOP_MCU_INT_CLEAR	0x0001C	W	0x00000000	Interrupt clear
VOP_MCU_INT_STATUS	0x00020	W	0x00000000	Interrupt status
VOP_MCU_STATUS	0x00024	W	0x00000000	Indicate vop_mcu current status.
VOP_MCU_CMD	0x00028	W	0x00000000	VOP command/index r/w entry
VOP_MCU_DATA	0x0002C	W	0x00000000	VOP data r/w entry
VOP_MCU_START	0x00030	W	0x00000000	Start VOP

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

15.4.2 Detail Register Description

VOP_MCU_CON

Address: Operational Base + offset (0x00000)

Mode control register

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:13	RW	0x0	sw_mcu_burst mask dummy fifo writing for burst stride 3'b000 : SINGLE 3'b001 : INCR4 3'b010 : INCR8 3'b111 : INCR16
12	RW	0x0	sw_auto_ckg auto clock gating 0: disable 1: enable

Bit	Attr	Reset Value	Description
11	RW	0x0	sw_wdata_bypass_en 0: no bypass 1: bypass
10	RW	0x0	sw_dither_down_en dither down enable 0: disable 1: enable
9	RW	0x0	sw_mcu_uv_swap write yuv420 data uv swap
8	RW	0x0	sw_mcu_input_format 0: rgb565 1: yuv420
7:6	RW	0x0	sw_mcu_y2r_mode yuv2rgb mode selection 2'b00 : BT601 limited 2'b01 : BT709 limited 2'b10 : BT601 fulled 2'b11 : reserved
5	RW	0x0	sw_mcu_byte_swap mcu write data half-word swap
4	RW	0x0	sw_mcu_hw_swap mcu write data half-word swap
3	RW	0x0	sw_mcu_bits mcu data width 0: 8bits 1: 16bits
2:1	RW	0x0	sw_mcu_wr_phase write data split 2'b00 : no split 2'b01 : two phase split 2'b10 : three phase split 2'b11 : four phase split
0	RW	0x0	sw_mcu_idle_dir data out valid , when there is no write/read operation, data pin(D0 ~ D15) as input or output 0: D0 ~ D15 input 1: D0 ~ D15 output valid

VOP_MCU_VERSION
Address: Operational Base + offset (0x00004)
VOP version

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	sw_mcu_version Indicate VOP version

VOP_MCU_TIMING
Address: Operational Base + offset (0x00008)
MCU interface timing control

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:12	RW	0x0	sw_mcu_csrw Chip select to R/W strobe leading edge this field specifies the number of processor clock cycles from the falling edge of CSN to the falling edge of RDN or WRN. If this bit is set to 0x0 , the processor uses a csrw valus of 0x1
11	RO	0x0	reserved
10:5	RW	0x00	sw_mcu_rwpw this field controls the width of RDN or WRN in processor clock cycles . If this bit is set to 0x0 , the processor uses an rwpw value of 0x1
4:0	RW	0x00	sw_mcu_rwcs this field specifies the number of processor clock cycles from the rising edge of RDN or WRN to the rising edge of CSN . If this is set to 0x0 , the processor uses an rwcs valus of 0x1

VOP_MCU_LCD_SIZE
Address: Operational Base + offset (0x000C)
Configure LCD size

Bit	Attr	Reset Value	Description
31:21	RO	0x0	reserved
20:12	RW	0x0	sw_mcu_lcd_height. LCD height, max value is 400,real -1
11:9	RO	0x0	reserved
8:0	RW	0x0	sw_mcu_lcd_width. LCD width, max value is 400, real -1

VOP_MCU_FIFO_WATERMARK
Address: Operational Base + offset (0x0010)
Configure FIFO Watermark

Bit	Attr	Reset Value	Description
31:13	RO	0x0	reserved
12:8	RW	0x00	sw_almost_empty_watermark Recommended value is not greater than 16
7:5	RO	0x0	reserved
4:0	RW	0x00	sw_almost_full_watermark Recommended value is not greater than 16, and if it's set to a value less than 2, real valus is 2.

VOP_MCU_SRT
Address: Operational Base + offset (0x00014)
VOP soft_reset

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	W1C	0x0	sw_soft_reset auto clear 0 : not clear 1 : clear

VOP_MCU_INT_EN
Address: Operational Base + offset (0x00018)
VOP Interrupt enable

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	sw_int_en_fifo_full 0 : fifo_full interrupt disable 1 : fifo_full interrupt enable
1	RW	0x0	sw_int_en_fifo_empty 0 : fifo_empty interrupt disable 1 : fifo_empty interrupt enable
0	RW	0x0	sw_int_en_frame_done 0 : frame_done interrupt disable 1 : frame_done interrupt enable

VOP_MCU_INT_CLEAR
Address: Operational Base + offset (0x0001C)
VOP Interrupt clear

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	RW	0x0	sw_int_clear_fifo_full 0 : no clear 1 : clear
1	RW	0x0	sw_int_clear_fifo_empty 0 : no clear 1 : clear
0	RW	0x0	sw_int_clear_frame_done 0 : no clear 1 : clear

VOP_MCU_INT_STATUS
Address: Operational Base + offset (0x00020)
VOP Interrupt status

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5	RW	0x0	sw_int_raw_fifo_full 0: invalid 1: valid
4	RW	0x0	sw_int_raw_fifo_empty 0: invalid 1: valid

Bit	Attr	Reset Value	Description
3	RW	0x0	sw_int_raw_frame_done 0: invalid 1: valid
2	RW	0x0	sw_int_fifo_full 0: invalid 1: valid
1	RW	0x0	sw_int_fifo_empty 0: invalid 1: valid
0	RW	0x0	sw_int_frame_done 0: invalid 1: valid

VOP_MCU_STATUS
Address: Operational Base + offset (0x00024)
VOP status register

Bit	Attr	Reset Value	Description
31:25	RO	0x0	reserved
24:16	RW	0x000	sw_mcu_current_line mcu current line
15:13	RO	0x0	reserved
12:4	RW	0x000	sw_mcu_current_row mcu current row
3:1	RO	0x0	reserved
0	RW	0x0	sw_mcu_working 0 : idle 1 : working

VOP_MCU_CMD
Address: Operational Base + offset (0x00028)
VOP command/index r/w entry

Bit	Attr	Reset Value	Description
31:8	RW	0x000000	mcu_cmd_high_bits VOP command/index write (only in split mode)
7:0	RW	0x00	mcu_cmd_low_bits VOP command/index write VOP status read (only use low 8-bit)

VOP_MCU_DATA
Address: Operational Base + offset (0x0002C)
VOP data r/w entry

Bit	Attr	Reset Value	Description
31:8	RW	0x000000	mcu_data_high_bits VOP data write(only in split mode)

Bit	Attr	Reset Value	Description
7:0	RW	0x00	mcu_data_low_bits VOP data write VOP data read (only use low 8-bit)

VOP_MCU_START
Address: Operational Base + offset (0x00030)
Start VOP

Bit	Attr	Reset Value	Description
0	RW	0x0	sw_mcu_start start VOP 0 : no start 1 : start

15.5 Timing Diagram

15.5.1 i8080 Timing

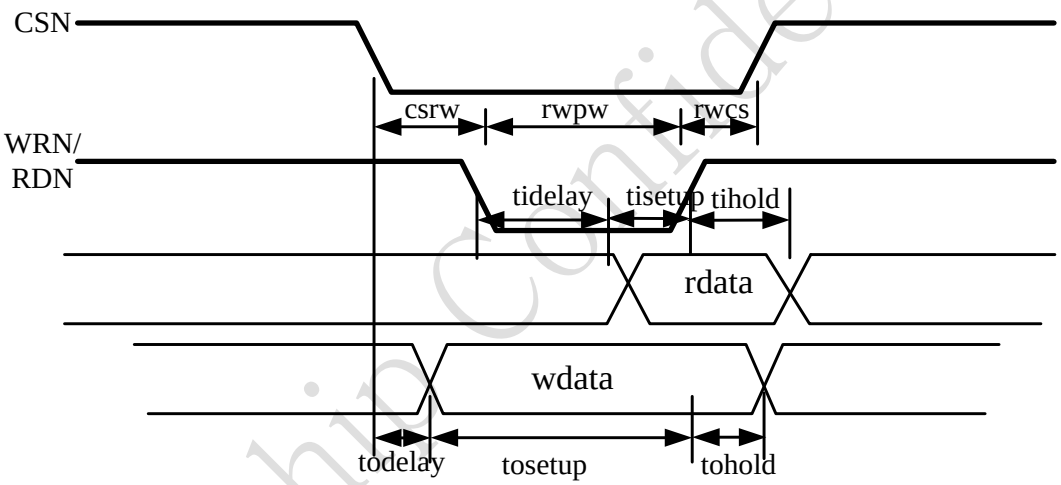


Fig 15-6 i8080 r/w timing

15.5.2 write data split

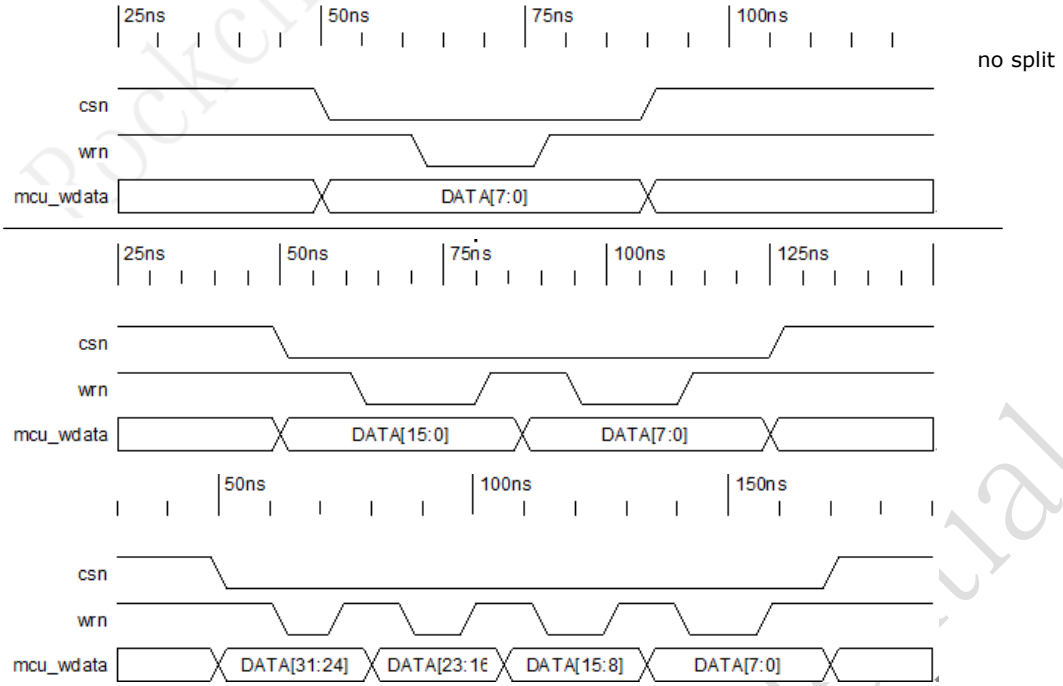


Fig 15-7 write data split

15.6 Application Notes

15.6.1 Configure registers

The flow of configuring register is shown in figure 4-1.

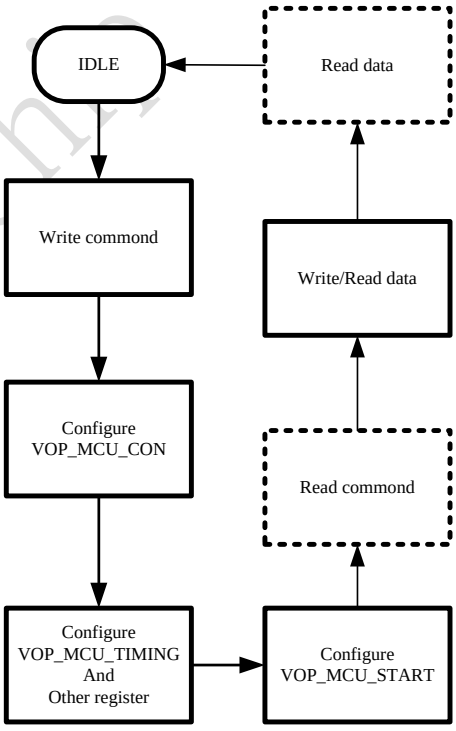


Fig 15-8 operation flow

Note1: Registers can't be configured again until VOP is idle when sw_mcu_working is 1'b0.
Note2: Can't read data/command or write command until VOP is idle.
Note3: Before a new picture is sent, vop_mcu_start must be configured again.

15.6.2 Input RGB565 data

Note1: If the source data format is RGB565, set sw_mcu_input_format to 0. Otherwise, sw_mcu_input_format is set to 1. Furthermore, if sw_mcu_input_format is set to 0 and sw_wdata_bypass_en is asserted, the VOP will act as the VOP for NanoC.

Note2：If the RGB565 source data is stored as figure1-2a，sw_mcu_hw_swap must be set to 1'b1 and sw_mcu_wr_phase must be set to 2'b11.

Note3：If the RGB565 source data is stored as figure1-2b，sw_mcu_hw_swap must be set to 1'b0 and sw_mcu_wr_phase must be set to 2'b01.

15.6.3 Input YUV420 data

Note1: If sw_dither_down_en is not asserted, the LBS of output data will be cut off directly.

Note2: If the YUV420 source data is stored as figure1-3b, sw_mcu_uv_swap must be asserted.

Note3: If the source data format is YUV420, sw_mcu_wr_phase must be set 2'b01.

15.6.4 Read command or data

Note1：Set sw_mcu_uv_swap to be 2'b00 to read command or data.

Chapter 16 SPI

16.1 Overview

The serial peripheral interface is an APB slave device. 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 slave select signals or the first edge of the serial clock. The slave select line is held high when the SPI is idle or disabled. This SPI controller can work as either master or slave mode.

16.1.1 Features

SPI Controller supports the following features:

- Support Motorola SPI,TI Synchronous Serial Protocol and National Semiconductor Microwire interface
- Support 32-bit APB bus
- Support two internal 16-bit wide and 32-location deep FIFOs, one for transmitting and the other for receiving serial data
- Support one chip select signals in master mode
- Support 4,8,16 bit serial data transfer
- Support configurable interrupt polarity
- Support asynchronous APB bus and SPI clock
- Support master and slave mode
- Support DMA handshake interface and configurable DMA water level
- Support transmit FIFO empty, underflow, receive FIFO full, overflow, interrupt and all interrupts can be masked
- Support configurable water level of transmit FIFO empty and receive FIFO full interrupt
- Support combine interrupt output
- Support up to half of SPI clock frequency transfer in master mode and one sixth of SPI clock frequency transfer in slave mode
- Support full and half duplex mode transfer
- Stop transmitting SCLK if transmit FIFO is empty or receive FIFO is full in master mode
- Support configurable delay from chip select active to SCLK active in master mode
- Support configurable period of chip select inactive between two parallel data in master mode
- Support big and little endian, MSB and LSB first transfer
- Support two 8-bit audio data store together in one 16-bit wide location
- Support sample RXD 0~3 SPI clock cycles later
- Support configurable SCLK polarity and phase
- Support fix and incremental address access to transmit and receive FIFO

16.2 Block Diagram

The SPI Controller comprises with:

- AMBA APB interface and DMA Controller Interface
- Transmit and receive FIFO controllers and an FSM controller
- Register block
- Shift control and interrupt

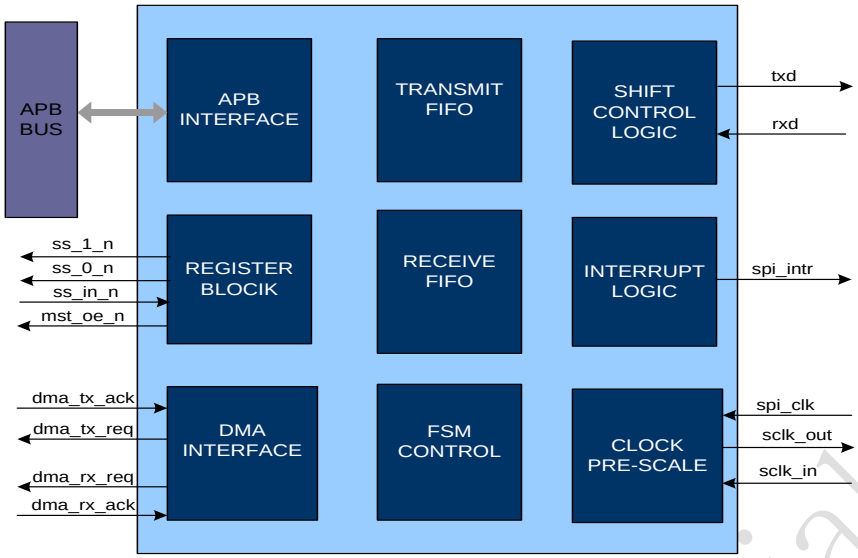


Fig 16-1 SPI Controller Block diagram

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 serialdevice into the SPI is pushed into the receive FIFO. Both FIFOs are 32x16bits.

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 result of all other SPI interrupts after masking.

16.3 Function description

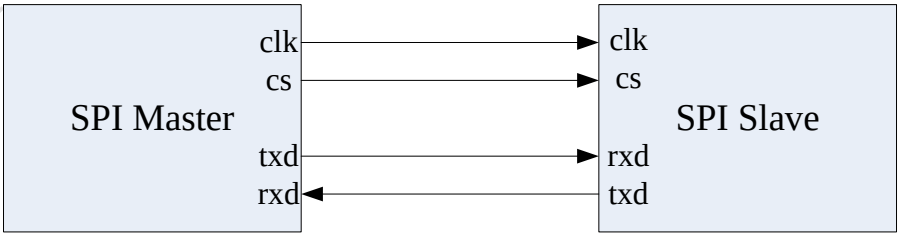


Fig 16-2 SPI Master and Slave Interconnection

The SPI controller support dynamic switching between master and slave in a system. The diagram show how the SPI controller connects with other SPI devices.

Operation Modes

The SPI can be configured in the following two fundamental modes of operation: Master Mode when SPI_CTRLR0 [20] is 1'b0, Slave Mode when SPI_CTRLR0 [20] is 1'b1.

Transfer Modes

The SPI operates in the following three modes when transferring data on the serial bus.

1. Transmit and Receive

When SPI_CTRLR0 [19:18]== 2'b00, both transmit and receive logic are valid.

2. Transmit Only

When SPI_CTRLR0 [19:18] == 2'b01, the receive data are invalid and should not be stored in the receive FIFO.

3. Receive Only

When SPI_CTRLR0 [19:18]== 2'b10, the transmit data are invalid.

Clock Ratios

A summary of the frequency ratio restrictions between the bit-rate clock (sclk_out/sclk_in) and the SPI peripheral clock (spi_clk) are described as,

When SPI Controller works as master, theF_{spi_clk}>= 2 × (maximum F_{sclk_out})

When SPI Controller works as slave, the F_{spi_clk}>= 6 × (maximum F_{sclk_in})

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 4/8/16 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. The following two figures show 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.

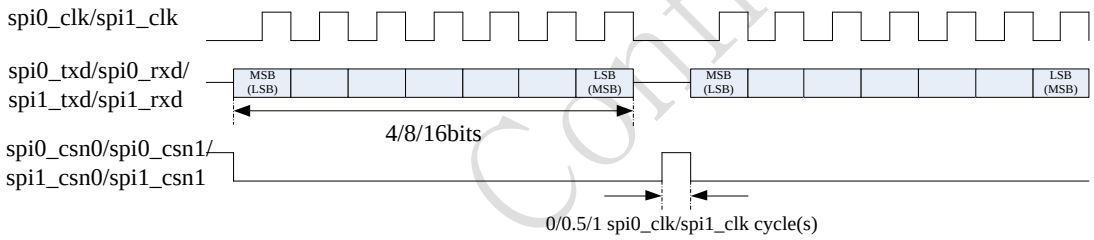


Fig 16-3 SPI Format (SCPH=0 SCPOL=0)

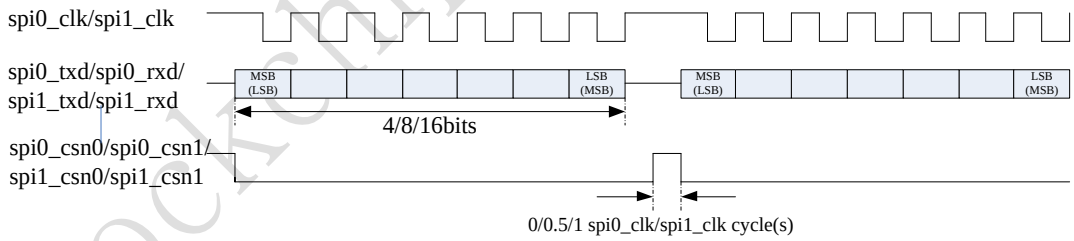


Fig 16-4 SPI Format (SCPH=0 SCPOL=1)

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. The following two figures show the timing diagram for the SPI format when the configuration parameter SCPH = 1.

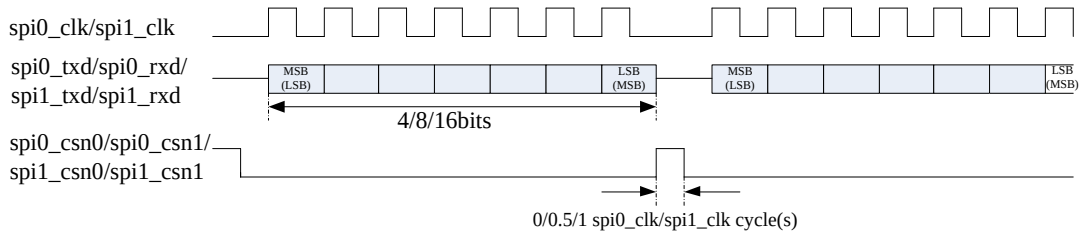


Fig 16-5 SPI Format (SCPH=1 SCPOL=0)

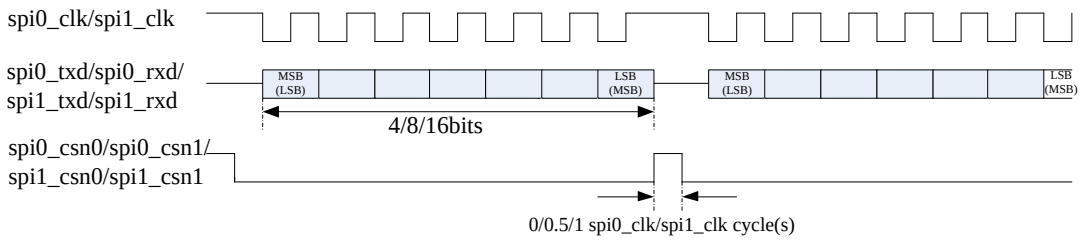


Fig 16-6 SPI Format (SCPH=1 SCPOL=1)

16.4 Register Description

This section describes the control/status registers of the design. Pay attention that there are two SPI controllers in the chip: spi0 & spi1, so the base address in the following register descriptions can be either spi0 or spi1 base address.

16.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SPI_CTRLR0	0x0000	W	0x00000002	Control Register 0
SPI_CTRLR1	0x0004	W	0x00000000	Control Register 1
SPI_ENR	0x0008	W	0x00000000	SPI Enable
SPI_SER	0x000c	W	0x00000000	Slave Enable Register
SPI_BAUDR	0x0010	W	0x00000000	Baud Rate Select
SPI_TXFTLR	0x0014	W	0x00000000	Transmit FIFO Threshold Level
SPI_RXFTLR	0x0018	W	0x00000000	Receive FIFO Threshold Level
SPI_TXFLR	0x001c	W	0x00000000	Transmit FIFO Level
SPI_RXFLR	0x0020	W	0x00000000	Receive FIFO Level
SPI_SR	0x0024	W	0x0000000c	SPI Status
SPI_IPR	0x0028	W	0x00000000	Interrupt Polarity
SPI_IMR	0x002c	W	0x00000000	Interrupt Mask
SPI_ISR	0x0030	W	0x00000000	Interrupt Status
SPI_RISR	0x0034	W	0x00000001	Raw Interrupt Status
SPI_ICR	0x0038	W	0x00000000	Interrupt Clear
SPI_DMACR	0x003c	W	0x00000000	DMA Control
SPI_DMATDLR	0x0040	W	0x00000000	DMA Transmit Data Level
SPI_DMARDLR	0x0044	W	0x00000000	DMA Receive Data Level
SPI_TXDR	0x0048	W	0x00000000	Transmit FIFO Data
SPI_RXDR	0x004c	W	0x00000000	Receive FIFO Data

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

16.4.2 Detail Register Description

SPI_CTRLR0

Address: Operational Base + offset (0x0000)
Control Register 0

Bit	Attr	Reset Value	Description
31:22	RO	0x0	reserved

Bit	Attr	Reset Value	Description
21	RW	0x0	MTM Microwire Transfer Mode Valid when frame format is set to National Semiconductors Microwire. 0: non-sequential transfer 1: sequential transfer
20	RW	0x0	OPM Operation Mode 0: Master Mode 1: Slave Mode
19:18	RW	0x0	XFM Transfer Mode 00 :Transmit & Receive 01 : Transmit Only 10 : Receive Only 11 :reserved
17:16	RW	0x0	FRF Frame Format 00: Motorola SPI 01: Texas Instruments SSP 10: National Semiconductors Microwire 11 : Reserved
15:14	RW	0x0	RSD Rxd Sample Delay When SPI is configured as a master, if the rxd data cannot be sampled by the sclk_out edge at the right time, this register should be configured to define the number of the spi_clk cycles after the active sclk_out edge to sample rxd data later when SPI works at high frequency. 00:do not delay 01:1 cycle delay 10:2 cycles delay 11:3 cycles delay
13	RW	0x0	BHT Byte and Halfword Transform Valid when data frame size is 8bit. 0:apb 16bit write/read, spi 8bit write/read 1: apb 8bit write/read, spi 8bit write/read
12	RW	0x0	FBM First Bit Mode 0:first bit is MSB 1:first bit is LSB

Bit	Attr	Reset Value	Description
11	RW	0x0	EM Endian Mode Serial endian mode can be configured by this bit. Apb endian mode is always little endian. 0:little endian 1:big endian
10	RW	0x0	SSD ss_n to sclk_out delay Valid when the frame format is set to Motorola SPI and SPI used as a master. 0: the period between ss_n active and sclk_out active is half sclk_out cycles. 1: the period between ss_n active and sclk_out active is one sclk_out cycle.
9:8	RW	0x0	CSM Chip Select Mode Valid when the frame format is set to Motorola SPI and SPI used as a master. 00: ss_n keep low after every frame data is transferred. 01:ss_n be high for half sclk_out cycles after every frame data is transferred. 10: ss_n be high for one sclk_out cycle after every frame data is transferred. 11:reserved
7	RW	0x0	SCPOL Serial Clock Polarity Valid when the frame format is set to Motorola SPI. 0: Inactive state of serial clock is low 1: Inactive state of serial clock is high
6	RW	0x0	SCPH Serial Clock Phase Valid when the frame format is set to Motorola SPI. 0: Serial clock toggles in middle of first data bit 1: Serial clock toggles at start of first data bit

Bit	Attr	Reset Value	Description
5:2	RW	0x0	CFS Control Frame Size Selects the length of the control word for the Microwire frame format. 0000~0010:reserved 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
1:0	RW	0x2	DFS Data Frame Size Selects the data frame length. 00:4bit data 01:8bit data 10:16bit data 11:reserved

SPI_CTRLR1
Address: Operational Base + offset (0x0004)
Control Register 1

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0000	NDM Number of Data Frames When Transfer Mode is receive only, 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.

SPI_ENR
Address: Operational Base + offset (0x0008)
SPI Enable

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	ENR SPI Enable Enables and disables all SPI operations. Transmit and receive FIFO buffers are cleared when the device is disabled.

SPI_SER
Address: Operational Base + offset (0x000c)
Slave Enable Register

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1:0	RW	0x0	SER Slave Select Enable This register is valid only when SPI is configured as a master device.

SPI_BAUDR
Address: Operational Base + offset (0x0010)
Baud Rate Select

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0000	BAUDR Baud Rate Select SPI Clock Divider. This register is valid only when the SPI is configured as a master device. 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: Fsclk_out = Fspi_clk/ SCKDV Where SCKDV is any even value between 2 and 65534. For example: for Fspi_clk = 3.6864MHz and SCKDV =2 Fsclk_out = 3.6864/2= 1.8432MHz

SPI_TXFTLR
Address: Operational Base + offset (0x0014)
Transmit FIFO Threshold Level

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	TXFTLR Transmit FIFO Threshold Level When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.

SPI_RXFTLR
Address: Operational Base + offset (0x0018)
Receive FIFO Threshold Level

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	RXFTLR Receive FIFO Threshold Level When the number of receive FIFO entries is greater than or equal to this value + 1, the receive FIFO full interrupt is triggered.

SPI_TXFLR
Address: Operational Base + offset (0x001c)
Transmit FIFO Level

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RO	0x00	TXFLR Transmit FIFO Level Contains the number of valid data entries in the transmit FIFO.

SPI_RXFLR
Address: Operational Base + offset (0x0020)
Receive FIFO Level

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RO	0x00	RXFLR Receive FIFO Level Contains the number of valid data entries in the receive FIFO.

SPI_SR
Address: Operational Base + offset (0x0024)
SPI Status

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4	RO	0x0	RFF Receive FIFO Full 0: Receive FIFO is not full 1: Receive FIFO is full
3	RO	0x1	RFE Receive FIFO Empty 0: Receive FIFO is not empty 1: Receive FIFO is empty
2	RO	0x1	TFE Transmit FIFO Empty 0: Transmit FIFO is not empty 1: Transmit FIFO is empty
1	RO	0x0	TFF Transmit FIFO Full 0: Transmit FIFO is not full 1: Transmit FIFO is full
0	RO	0x0	BSF 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

SPI_IPR
Address: Operational Base + offset (0x0028)
Interrupt Polarity

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	IPR Interrupt Polarity Interrupt Polarity Register 0:Active Interrupt Polarity Level is HIGH 1: Active Interrupt Polarity Level is LOW

SPI_IMR
Address: Operational Base + offset (0x002c)
Interrupt Mask

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RW	0x0	RFFIM Receive FIFO Full Interrupt Mask 0: spi_rxf_intr interrupt is masked 1: spi_rxf_intr interrupt is not masked

Bit	Attr	Reset Value	Description
3	RW	0x0	RFOIM Receive FIFO Overflow Interrupt Mask 0: spi_rxo_intr interrupt is masked 1: spi_rxo_intr interrupt is not masked
2	RW	0x0	RFUIM Receive FIFO Underflow Interrupt Mask 0: spi_rxu_intr interrupt is masked 1: spi_rxu_intr interrupt is not masked
1	RW	0x0	TFOIM Transmit FIFO Overflow Interrupt Mask 0: spi_txo_intr interrupt is masked 1: spi_txo_intr interrupt is not masked
0	RW	0x0	TFEIM Transmit FIFO Empty Interrupt Mask 0: spi_txe_intr interrupt is masked 1: spi_txe_intr interrupt is not masked

SPI_ISR
Address: Operational Base + offset (0x0030)
Interrupt Status

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RO	0x0	RFFIS Receive FIFO Full Interrupt Status 0: spi_rxf_intr interrupt is not active after masking 1: spi_rxf_intr interrupt is full after masking
3	RO	0x0	RFOIS Receive FIFO Overflow Interrupt Status 0: spi_rxo_intr interrupt is not active after masking 1: spi_rxo_intr interrupt is active after masking
2	RO	0x0	RFUIS Receive FIFO Underflow Interrupt Status 0: spi_rxu_intr interrupt is not active after masking 1: spi_rxu_intr interrupt is active after masking
1	RO	0x0	TFOIS Transmit FIFO Overflow Interrupt Status 0: spi_txo_intr interrupt is not active after masking 1: spi_txo_intr interrupt is active after masking

Bit	Attr	Reset Value	Description
0	RO	0x0	TFEIS Transmit FIFO Empty Interrupt Status 0: spi_txe_intr interrupt is not active after masking 1: spi_txe_intr interrupt is active after masking

SPI_RISR
Address: Operational Base + offset (0x0034)
Raw Interrupt Status

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RO	0x0	RFFRIS Receive FIFO Full Raw Interrupt Status 0: spi_rxf_intr interrupt is not active prior to masking 1: spi_rxf_intr interrupt is full prior to masking
3	RO	0x0	RFORIS Receive FIFO Overflow Raw Interrupt Status 0 = spi_rxo_intr interrupt is not active prior to masking 1 = spi_rxo_intr interrupt is active prior to masking
2	RO	0x0	RFURIS Receive FIFO Underflow Raw Interrupt Status 0: spi_rxu_intr interrupt is not active prior to masking 1: spi_rxu_intr interrupt is active prior to masking
1	RO	0x0	TFORIS Transmit FIFO Overflow Raw Interrupt Status 0: spi_txo_intr interrupt is not active prior to masking 1: spi_txo_intr interrupt is active prior to masking
0	RO	0x1	TFERIS Transmit FIFO Empty Raw Interrupt Status 0: spi_txe_intr interrupt is not active prior to masking 1: spi_txe_intr interrupt is active prior to masking

SPI_ICR
Address: Operational Base + offset (0x0038)

Interrupt Clear

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	WO	0x0	CTFOI Clear Transmit FIFO Overflow Interrupt Write 1 to Clear Transmit FIFO Overflow Interrupt
2	WO	0x0	CRFOI Clear Receive FIFO Overflow Interrupt Write 1 to Clear Receive FIFO Overflow Interrupt
1	WO	0x0	CRFUI Clear Receive FIFO Underflow Interrupt Write 1 to Clear Receive FIFO Underflow Interrupt
0	WO	0x0	CCI Clear Combined Interrupt Write 1 to Clear Combined Interrupt

SPI_DMACR

Address: Operational Base + offset (0x003c)

DMA Control

Bit	Attr	Reset Value	Description
31:2	RO	0x0	reserved
1	RW	0x0	TDE Transmit DMA Enable 0: Transmit DMA disabled 1: Transmit DMA enabled
0	RW	0x0	RDE Receive DMA Enable 0: Receive DMA disabled 1: Receive DMA enabled

SPI_DMATDLR

Address: Operational Base + offset (0x0040)

DMA Transmit Data Level

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved

Bit	Attr	Reset Value	Description
4:0	RW	0x00	TDL 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 Transmit DMA Enable (DMACR[1]) = 1.

SPI_DMARDLR
Address: Operational Base + offset (0x0044)
DMA Receive Data Level

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	RDL 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 Receive DMA Enable(DMACR[0])=1.

SPI_TXDR
Address: Operational Base + offset (0x0048)
Transmit FIFO Data

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	WO	0x0000	TXDR Transmit FIFO Data Register. When it is written to, data are moved into the transmit FIFO.

SPI_RXDR
Address: Operational Base + offset (0x004c)
Receive FIFO Data

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0000	RXDR Receive FIFO Data Register. When the register is read, data in the receive FIFO is accessed.

16.5 Interface description

SPI0 have 2 IOMUX, which is controlled by GRF_IOMUX_CON[6].
When GRF_IOMUX_CON[6] is 1'b0. The IOMUX is as follow.

Table 16-1SPI0 Interface Description

Module pin	Dir.	Pad name	IOMUX
SPI0 Interface			
Spi0_txd	O	IO_LCDd0_SPI0Atx_EBCsddo0_GPIO0c0	GRF_GPIO0C_IOMUX[1:0]=2'b10
Spi0_rxd	I	IO_LCDd1_SPI0Arx_EBCsddo1_GPIO0c1	GRF_GPIO0C_IOMUX[3:2]=2'b10
Clk_spi0	I/O	IO_LCDd2_SPI0Aclk_EBCsddo2_GPIO0c2	GRF_GPIO0C_IOMUX[5:4]=2'b10
Spi0_ss	I/O	IO_LCDd3_SPI0Acs_EBCsddo3_GPIO0c3	GRF_GPIO0C_IOMUX[7:6]=2'b10

When GRF_IOMUX_CON[6] is 1'b1. The IOMUX is as follow.

Table 16-2 SPI0 Interface Description

Module pin	Dir.	Pad name	IOMUX
SPI0 Interface			
Spi0_txd	O	IO_I2C0Ascl_SPI0Btx_EBCsdce5_GPIOP2b2	GRF_GPIO2B_IOMUX[5:4]=2'b10
Spi0_rxd	I	IO_I2C0Asda_SPI0Brx_EBCborder0_GPIOP2b3	GRF_GPIO2B_IOMUX[7:6]=2'b10
Clk_spi0	I/O	IO_I2C1Ascl_SPI0Bclk_EBCborder1_GPIOP2b1	GRF_GPIO2B_IOMUX[3:2]=2'b10
Spi0_ss	I/O	IO_I2C1Asda_SPI0Bcs_EBCsdce3_GPIOP2b0	GRF_GPIO2B_IOMUX[1:0]=2'b10

SPI1 have 2 IOMUX, which is controlled by GRF_IOMUX_CON[7].
When GRF_IOMUX_CON[7] is 1'b0. The IOMUX is as follow.

Table 16-3PI1 Interface Description

Module pin	Dir.	Pad name	IOMUX
SPI1 Interface			
Spi1_txd	O	IO_SDMMCd1_SPI1Atx_UART4rx_GPIO1b0	GRF_GPIO1B_IOMUX[1:0]=2'b10
Spi1_rxd	I	IO_SDMMCd0_SPI1Arx_UART4tx_GPIO1a7	GRF_GPIO1A_IOMUX[15:14]=2'b10
Clk_spi1	I/O	IO_SDMMCclk_SPI1Aclk_UART3rx_GPIO1a6	GRF_GPIO1A_IOMUX[13:12]=2'b10
Spi1_ss	I/O	IO_SDMMCcmd_SPI1Acs_UART3tx_GPIO1a5	GRF_GPIO1A_IOMUX[11:10]=2'b10

When GRF_IOMUX_CON[7] is 1'b1. The IOMUX is as follow.

Table 16-4 SPI1 Interface Description

Module pin	Dir.	Pad name	IOMUX
SPI1 Interface			
Spi1_txd	O	IO_UART1Arx_I2C0Bscl_SPI1Btx_GPIOP2c1	GRF_GPIO2C_IOMUX[3:2]=2'b11
Spi1_rxd	I	IO_UART1Actx_JTG0tck_SPI1Brx_GPIOP2b7	GRF_GPIO2B_IOMUX[15:14]=2'b11
Clk_spi	I/O	IO_UART1Atx_I2C0Bsda_SPI1Bclk_GPIOP	GRF_GPIO2C_IOMUX[1:0]=2'b1

1		2c0	1
Spi1_ss	I/O	IO_UART1Arts_JTG0tms_SPI1Bcs_GPIOP2b6	GRF_GPIO2B_IOMUX[13:12]=2'b11

16.6 Application Notes

Clock Ratios

A summary of the frequency ratio restrictions between the bit-rate clock (sclk_out/sclk_in) and the SPI peripheral clock (spi_clk) are described as,

When SPI Controller works as master, theF_{spi_clk}>= 2 × (maximum F_{sclk_out})

When SPI Controller works as slave,theF_{spi_clk}>= 6 × (maximum F_{sclk_in})

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_ENR = 0), no serial transfers can occur and sclk_out is held in "inactive" state, as defined by the serial protocol under which it operates.

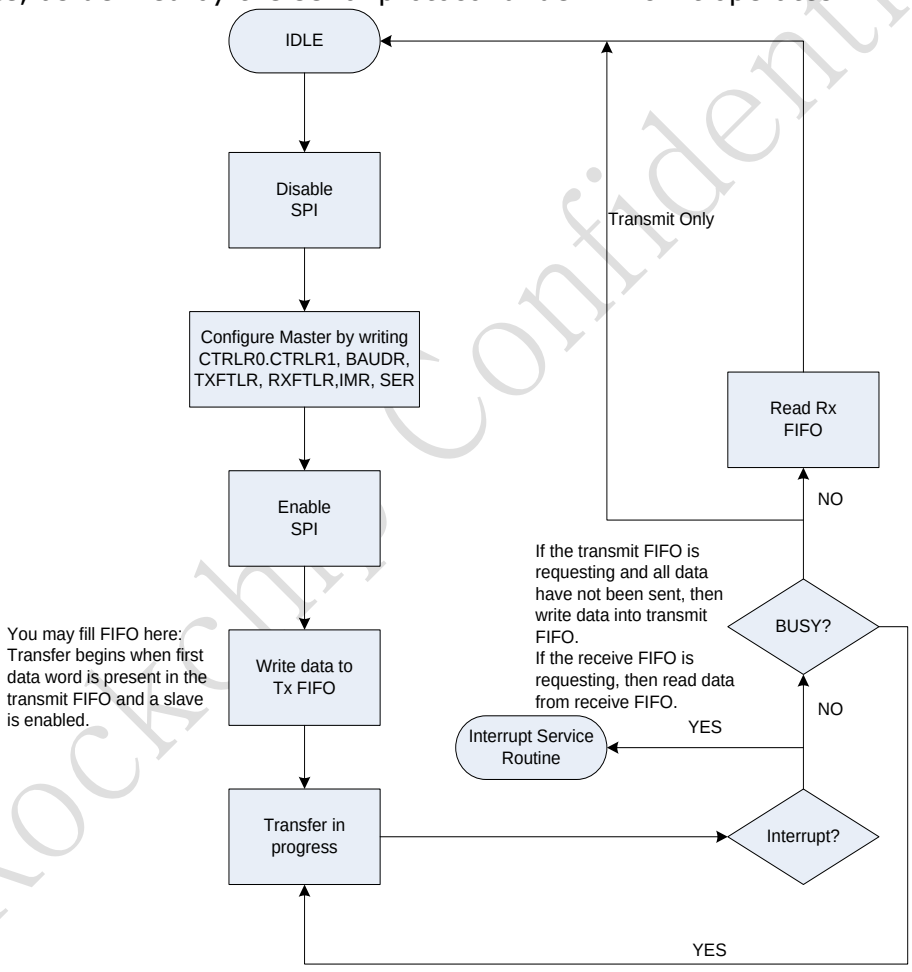


Fig 16-7 SPI Master transfer flow diagram

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.

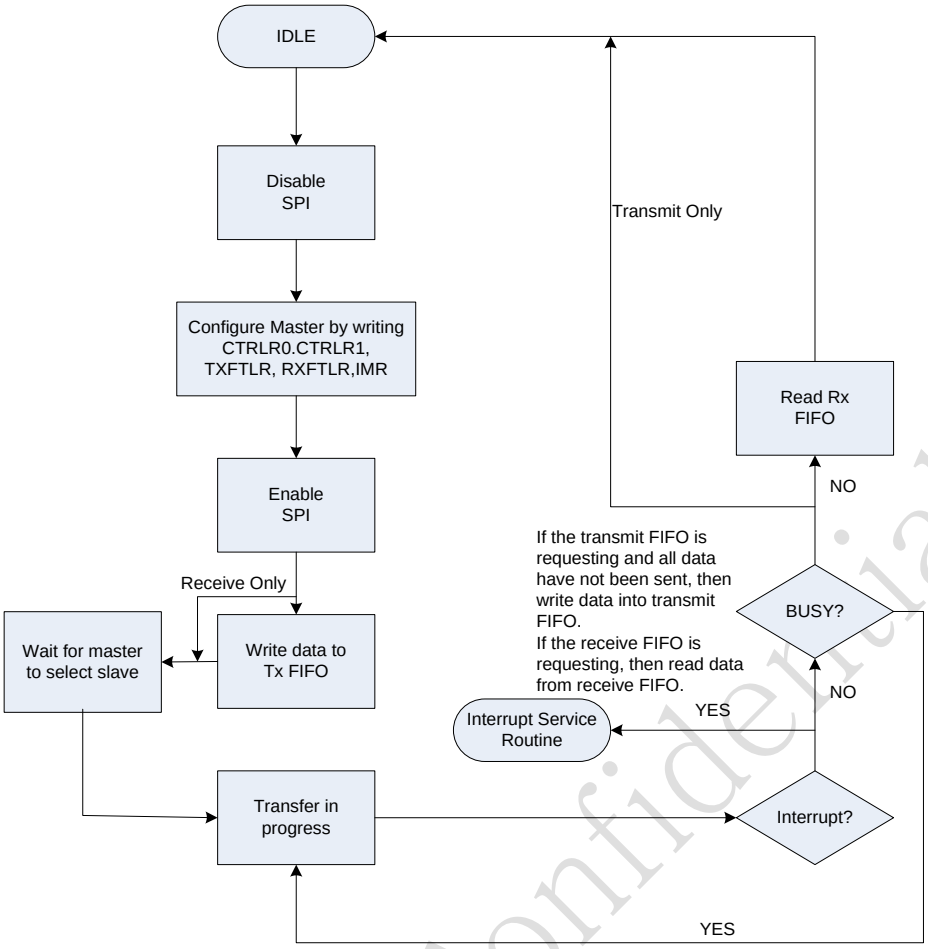


Fig 16-8 SPI Slave transfer flow diagram

Chapter 17 SAR-ADC

17.1 Overview

17.1.1 Features

The ADC is an 8-channelsignal-ended 10-bit Successive Approximation Register (SAR) A/D Converter. It uses the supply and ground as it reference which avoid use of any external reference. It converts the analog input signal into 10-bit binary digital codes at maximum conversion rate of 100KSPS with 1MHz A/D converter clock.

17.2 Block Diagram

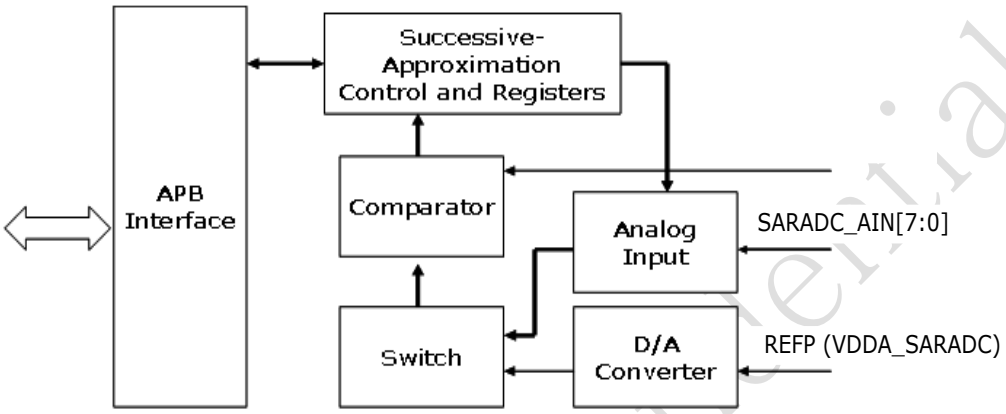


Fig 17-1 SAR-ADC block diagram

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 SARADC_AIN[7:0] 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.

17.3 Function description

In the chip, SAR-ADC works at single-sample operation mode.

This mode is useful to sample an analog input when there is a gap between two samples to be converted. In this mode START is asserted only on the rising edge of CLKIN where conversion is needed. At the end of every conversion EOC signal is made high and valid output data is available at the rising edge of EOC. The detailed timing diagram will be shown in the following.

17.4 Register Description

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

17.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
SARADC_DATA	0x0000	W	0x00000000	This register contains the data after A/D Conversion.
SARADC_STAS	0x0004	W	0x00000000	The status register of A/D Converter.
SARADC_CTRL	0x0008	W	0x00000000	The control register of A/D Converter.
GRF_VREF_CON	GRF_BASE+0x00cc	W	0x00000018	The control register of ADC_REF.

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

17.4.2 Detail Register Description

SARADC_DATA

Address: Operational Base + offset (0x0000)

This register contains the data after A/D Conversion.

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved
9:0	RO	0x000	adc_data A/D value of the last conversion (DOUT[9:0]).

SARADC_STAS

Address: Operational Base + offset (0x0004)

The status register of A/D Converter.

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	adc_status ADC status (EOC) 1'b0: ADC stop 1'b1: Conversion in progress

SARADC_CTRL

Address: Operational Base + offset (0x0008)

The control register of A/D Converter.

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	int_status Interrupt status. This bit will be set to 1 when end of conversion. Set 0 to clear the interrupt.
5	RW	0x0	int_en Interrupt enable. 1'b0: Disable 1'b1: Enable
4	RW	0x0	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	0x0	adc_power_ctrl ADC power down control bit 1'b0: ADC power down 1'b1: ADC power up and reset start signal will be asserted (DLY_PU_SOC+2) sclk clock period later after power up

Bit	Attr	Reset Value	Description
2:0	RW	0x0	adc_input_src_sel ADC input source selection(CH_SEL[2:0]). 3'b000: Input source 0 (SARADC_AIN[0]) 3'b001: Input source 1 (SARADC_AIN[1]) 3'b010: Input source 2 (SARADC_AIN[2]) 3'b011: Input source 3 (SARADC_AIN[3]) 3'b100: Input source 4 (SARADC_AIN[4]) 3'b101: Input source 5 (SARADC_AIN[5]) 3'b110: Input source 6 (SARADC_AIN[6]) 3'b111: Input source 7 (SARADC_AIN[7])

GRF_VREF_CON
Address: GRF Base + offset (0x00cc)
The control register of ADC_REF.

Bit	Attr	Reset Value	Description
31:16	WO	0x0	Write enable
15:7	RO	0x0	reserved
6:4	RW	0x1	adc_vbg_sel select the VBG voltage which should has the best temperature characteristics
3:1	RW	0x4	vref_trim the 2.5V output voltage trim register, the turning range is 2.4V~2.6V. 000-> 2.4V 111-> 2.6V
0	RW	0x0	ref_pwd when high the block is power down

17.5 Timing Diagram

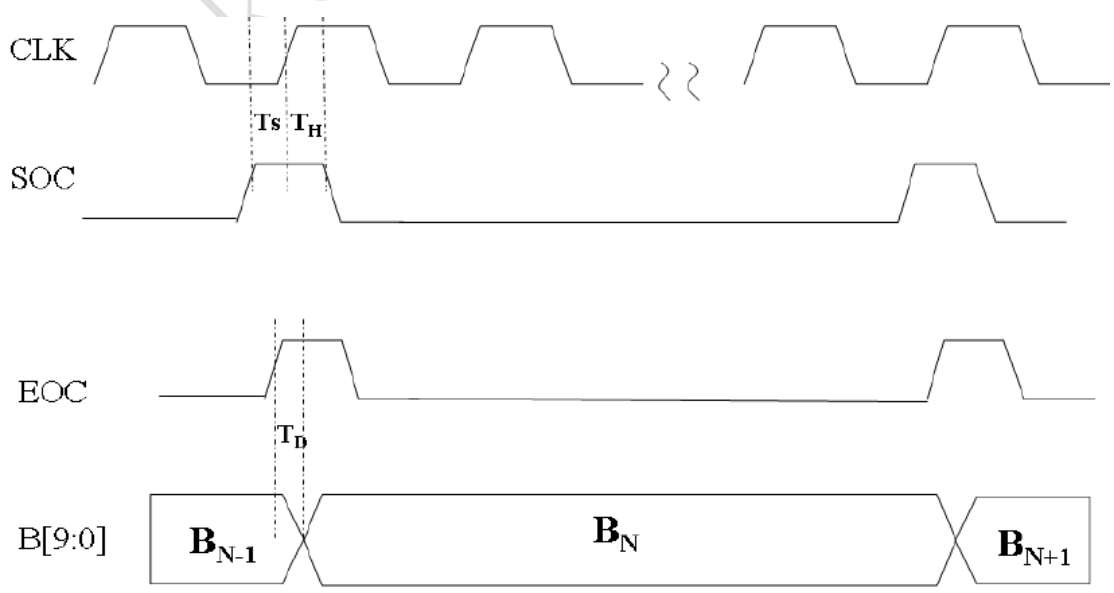


Fig 17-2 SAR-ADC timing diagram in single-sample conversion mode

The following table has shown the detailed value for timing parameters in the above diagram.

Table 17-1SAR-ADC timing parameters list

Timing	Symbol	Value			Unit	Description
		Min	Typ	Max		
Data Setup	T_S	0			ns	Setup time for SOC
Data Hold	T_H	5			ns	Hold time for SOC
Data delay	T_D			0.33	ns	B[9:0] delay from rising edge of EOC
CLK period		227			ns	CLK period

17.6 Application Notes

Steps of adc conversion:

- Write SARADC_CTRL[3] as 0 to power down adc converter.
- Write SARADC_CTRL[2:0] as n to select adc channel(n).
- Write SARADC_CTRL[5] as 1 to enable adc interrupt.
- Write SARADC_CTRL[3] as 1 to power up adc converter.
- Wait for adc interrupt or poll SARADC_STAS register to assert whether the conversion is completed
- Read the conversion result from SARADC_DATA[9:0]

Note: The A/D converter was designed to operate at maximum 1MHZ.

Chapter 18 Timer

18.1 Overview

Timer is a programmable timer peripheral. This component is a APBs lave device. Timer count down from a programmed value and generate an interrupt when the count reaches zero.

18.1.1 Features

Timer supports the following features:

- One APB timers in the SOC system two operation modes: free-running and user-defined count.
- Maskable for each individual interrupt
- 2 timer channels

18.2 Block Diagram

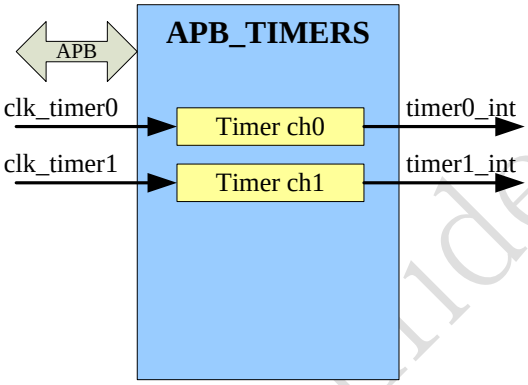


Fig 18-1 Timers Block Diagram

The above figure shows the architecture of the APB timers (include two programmable timer channel) that in the peripheral subsystem.

18.3 Function description

18.3.1 Timer clock

TIMER0~5 are in the PERI subsystem, using timer ch0 ~ ch5 respectively. The timer clock is 24MHz OSC.

18.3.2 Programming sequence

1. Initialize the timer by the TIMERn_CONTROLREG (0≤n≤1)register:

- Disable the timer by writing a "0" to the timer enable bit (bit 0). Accordingly, the timer_enoutput signal is de-asserted.
- Program the timer mode—user-defined or free-running—by writing a "0" or "1" respectively, to the timer mode bit (bit 1).
- 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 count value into the TIMERn_LOAD_COUNT1 (0≤n≤5) and TIMERn_ LOAD_COUNT0 (0≤n≤5)register.

3. Enable the timer by writing a "1" to bit 0 of TIMERn_CONTROLREG (0≤n≤5).

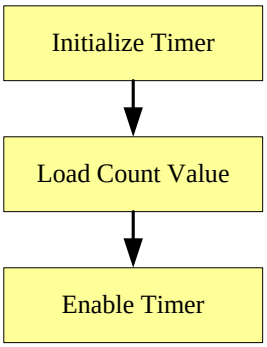


Fig 18-2 Timer Usage Flow

18.3.3 Loading a timer count value

The initial value for each timer—that is, the value from which it counts down—is loaded into the timer using the load count register (TIMERn_LOAD_COUNT1 (0≤n≤1) and TIMERn_ LOAD_COUNT0 (0≤n≤1)). Two events can cause a timer to load the initial value from its load count register:

- Timer is enabled after reset or disabled.
- Timer counts down to 0, when timer is configured into free-running mode.

18.3.4 Timer mode selection

- User-defined count mode – Timer loads TIMERn_LOAD_COUNT1(0≤n≤1)and TIMERn_ LOAD_ COUNT0(0≤n≤1)register as initial value. Timer will not automatically load the count register, when timer counts down to 0. User need to disable timer firstly and follow the programming sequence to make timer work again.
- Free-running mode – Timer loads the TIMERn_LOAD_COUNT1(0≤n≤1)and TIMERn_ LOAD_ COUNT0(0≤n≤1)register as initial value. Timer will automatically load the count register, when timer counts down to 0.

18.4 Register Description

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

18.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
TIMER0_LOAD_COUNT0	0x0000	W	0x00000000	Timer0 Load Count Register
TIMER0_LOAD_COUNT1	0x0004	W	0x00000000	Timer0 Load Count Register
TIMER0_CURRENT_VALUE0	0x0008	W	0x00000000	Timer0 Current Value Register
TIMER0_CURRENT_VALUE1	0x000C	W	0x00000000	Timer0 Current Value Register
TIMER0_CONTROLREG	0x0010	W	0x00000000	Timer0 Control Register
TIMER0_INTSTATUS	0x0018	W	0x00000000	Timer0 Interrupt Status Register
TIMER1_LOAD_COUNT0	0x0020	W	0x00000000	Timer1 Load Count Register
TIMER1_LOAD_COUNT1	0x0024	W	0x00000000	Timer1 Load Count Register
TIMER1_CURRENT_VALUE0	0x0028	W	0x00000000	Timer1 Current Value Register
TIMER1_CURRENT_VALUE1	0x002c	W	0x00000000	Timer1 Current Value Register
TIMER1_CONTROLREG	0x0030	W	0x00000000	Timer1 Control Register
TIMER1_INTSTATUS	0x0038	W	0x00000000	Timer1 Interrupt Status Register

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

18.4.2 Detail Register Description

TIMERn_LOAD_COUNT0

Address: Operational Base + offset(0x00+n*0x20)
Timer n Load Count Register(0≤n≤1)

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Low 32 bits Value to be loaded into Timer n. This is the value from which counting commences.

TIMERn_LOAD_COUNT1
Address: Operational Base + offset(0x04+n*0x20)
Timer n Load Count Register(0≤n≤1)

Bit	Attr	Reset Value	Description
31:0	RW	0x0	High 32 bits Value to be loaded into Timer n. This is the value from which counting commences.

TIMERn_CURRENT_VALUE0
Address: Operational Base + offset(0x08+n*0x20)
Timer n Current Value Register(0≤n≤1)

Bit	Attr	Reset Value	Description
31:0	R	0x0	Low 32 bits of Current Value of Timer n.

TIMERn_CURRENT_VALUE1
Address: Operational Base + offset(0x0c+n*0x20)
Timer n Current Value Register(0≤n≤1)

Bit	Attr	Reset Value	Description
31:0	R	0x0	High 32 bits of Current Value of Timer n.

TIMERn_CONTROLREG
Address: Operational Base + offset(0x10+n*0x20)
Timer n Control Register(0≤n≤1)

Bit	Attr	Reset Value	Description
31:3	-	-	Reserved
2	RW	0x0	Timer interrupt mask. 1'b0: mask 1'b1: not mask
1	RW	0x0	Timer mode. 1'b0: free-running mode 1'b1: user-defined count mode
0	RW	0x0	Timer enable. 1'b0: disable 1'b1: enable

TIMERn_INTSTATUS
Address: Operational Base + offset(0x18+n*0x20)
Timer n Interrupt Status Register(0≤n≤1)

Bit	Attr	Reset Value	Description
31:1	-	-	Reserved
0	RW	0x0	This register contains the interrupt status for timer n Write 1 to this register will clear the interrupt

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

18.5 Application Notes

In the chip, the timer_clk is from 24MHz OSC, asynchronous to the pclk. When user disable the timer enable bit (bit 0 of TIMERn_CONTROLREG(0≤n≤5)), the timer_en output signal is de-asserted, and timer_clk will stop. When user enables the timer, the timer_en signal is asserted and timer_clkwill start running.

The application is only allowed to re-config registers when timer_en is low.

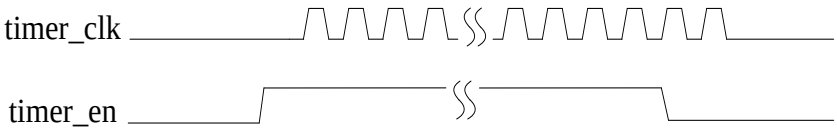


Fig 18-3 Timing between timer_en and timer_clk

Please refer to function description section for the timer usage flow.

Chapter 19 GPIO

19.1 Overview

GPIO is a programmable General Purpose Programming I/O peripheral. This component is an 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.
GPIO supports the following features:

- 32 bits APB bus width
- 32 independently configurable signals
- Separate data registers and data direction registers for each signal
- Software control for each signal, or for each bit of each signal
- Configurable interrupt mode, and combinational interrupt output
- 32 bits for GPIO's IO port

19.2 Block Diagram

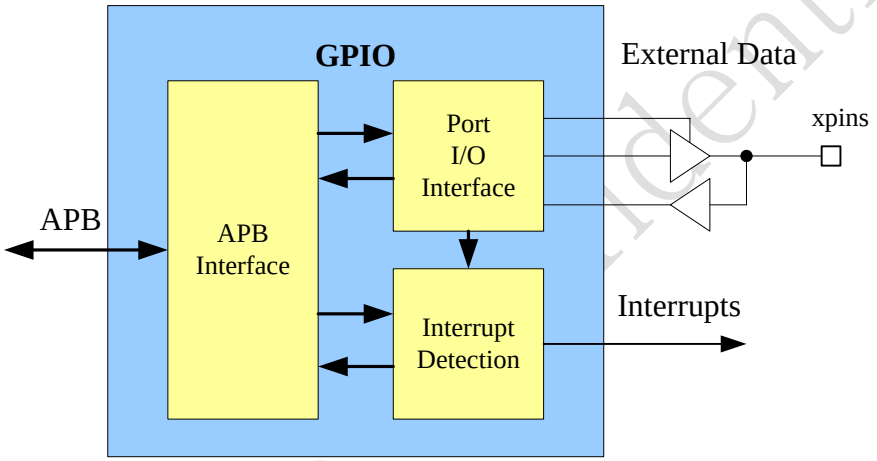


Fig 19-1 GPIO Block Diagram

Block Descriptions:
APB Interface
The APB Interface implements the APB slave operation. Its 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.

19.3 Register Description

19.3.1 Register Summary for GPIO0

Name	Offset	Size	Reset Value	Description
GPIO_SWPORT_DR	0x0000	W	0x00000000	I/O Port data register
GPIO_SWPORT_DDR	0x0004	W	0x00000000	I/O Port data direction register
GPIO_INTEN	0x0030	W	0x00000000	Interrupt enable register
GPIO_INTMASK	0x0034	W	0x00000000	Interrupt mask register
GPIO_INTTYPE_LEVEL	0x0038	W	0x00000000	Interrupt level register
GPIO_INT_POLARITY	0x003C	W	0x00000000	Interrupt polarity register
GPIO_INT_STATUS	0x0040	W	0x00000000	Interrupt status register
GPIO_INT_RAWSTATUS	0x0044	W	0x00000000	Raw Interrupt status
GPIO_DEBOUNCE	0x0048	W	0x00000000	Debounce enable register
GPIO_PORT_EOI	0x004C	W	0x00000000	I/O Port clear interrupt register

GPIO_EXT_PORT	0x0050	W	0x00000000	I/O Port 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*

19.3.2 Register Summary for GPIO1

Name	Offset	Size	Reset Value	Description
GPIO_SWPORT_DR	0x0000	W	0x00000000	I/O Port data register
GPIO_SWPORT_DDR	0x0004	W	0x00000000	I/O Port data direction register
GPIO_INTEN	0x0030	W	0x00000000	Interrupt enable register
GPIO_INTMASK	0x0034	W	0x00000000	Interrupt mask register
GPIO_INTTYPE_LEVEL	0x0038	W	0x00000000	Interrupt level register
GPIO_INT_POLARITY	0x003C	W	0x00000000	Interrupt polarity register
GPIO_INT_STATUS	0x0040	W	0x00000000	Interrupt status register
GPIO_INT_RAWSTATUS	0x0044	W	0x00000000	Raw Interrupt status
GPIO_DEBOUNCE	0x0048	W	0x00000000	Debounce enable register
GPIO_PORT_EOI	0x004C	W	0x00000000	I/O Port clear interrupt register
GPIO_EXT_PORT	0x0050	W	0x00000000	I/O Port 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*

19.3.3 Register Summary for GPIO2

Name	Offset	Size	Reset Value	Description
GPIO_SWPORT_DR	0x0000	W	0x00000000	I/O Port data register
GPIO_SWPORT_DDR	0x0004	W	0x00000000	I/O Port data direction register
GPIO_INTEN	0x0030	W	0x00000000	Interrupt enable register
GPIO_INTMASK	0x0034	W	0x00000000	Interrupt mask register
GPIO_INTTYPE_LEVEL	0x0038	W	0x00000000	Interrupt level register
GPIO_INT_POLARITY	0x003C	W	0x00000000	Interrupt polarity register
GPIO_INT_STATUS	0x0040	W	0x00000000	Interrupt status register
GPIO_INT_RAWSTATUS	0x0044	W	0x00000000	Raw Interrupt status
GPIO_DEBOUNCE	0x0048	W	0x00000000	Debounce enable register
GPIO_PORT_EOI	0x004C	W	0x00000000	I/O Port clear interrupt register
GPIO_EXT_PORT	0x0050	W	0x00000000	I/O Port 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*

19.3.4 Detailed Register Description

Operational Base Address of GPIO0 is 0x40160000.
Operational Base Address of GPIO1 is 0x40170000.
Operational Base Address of GPIO2 is0x50030000.

GPIO_SWPORT_DR

Address: Operational Base + offset(0x0000)
I/O Port data register

bit	Attr	Reset Value	Description
31:0	RW	0x00	Values written to this register are output on the I/O signals for the I/O Port if the corresponding data direction bits for the I/O Port are set to Output mode.

			The value read back is equal to the last value written to this register.

GPIO_SWPORT_DDR
Address: Operational Base + offset(0x0004)
I/O Port data direction register

bit	Attr	Reset Value	Description
31:0	RW	0x00	Values written to this register independently control the direction of the corresponding data bit in the I/O Port. 0: Input (default) 1: Output

GPIO_INTEN
Address: Operational Base + offset(0x0030)
Interrupt enable register

bit	Attr	Reset Value	Description
31:0	RW	0x00	Allows each bit of I/O port to be configured for interrupts. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on I/O Porto become an interrupt; otherwise, I/O Port operates as a normal GPIO signal. Interrupts are disabled on the corresponding bits of I/O Port if the corresponding data direction register is set to Output. 0: Configure I/O Port bit as normal GPIO signal (default) 1: Configure I/O Port bit as interrupt

GPIO_INTMASK
Address: Operational Base + offset(0x0034)
Interrupt mask register

bit	Attr	Reset Value	Description
31:0	RW	0x00	Controls whether an interrupt on I/O Port 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(0x0038)
Interrupt level register

bit	Attr	Reset Value	Description
31:0	RW	0x00	Controls the type of interrupt that can occur on I/O Port. 0: Level-sensitive (default) 1: Edge-sensitive

GPIO_INT_POLARITY

Address: Operational Base + offset(0x003C)
Interrupt polarity register

bit	Attr	Reset Value	Description
31:0	RW	0x00	Controls the polarity of edge or level sensitivity that can occur on input of I/O Port. 0: Active-low (default) 1: Active-high

GPIO_INT_STATUS
Address: Operational Base + offset(0x0040)
Interrupt status register

bit	Attr	Reset Value	Description
31:0	R	0x00	Interrupt status of I/O Port

GPIO_INT_RAWSTATUS
Address: Operational Base + offset(0x0044)
Raw Interrupt status register

bit	Attr	Reset Value	Description
31:0	R	0x00	Raw interrupt of status of I/O Port (premasking bits)

GPIO_DEBOUNCE
Address: Operational Base + offset(0x0048)
Debounce enable register

bit	Attr	Reset Value	Description
31:0	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_PORT_EOI
Address: Operational Base + offset(0x004C)
I/O Port clear interrupt register

bit	Attr	Reset Value	Description
31:0	W	0x00	Controls the clearing of edge type interrupts from I/O Port. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when I/O Port is not configured for interrupts. 0: No interrupt clear (default) 1: Clear interrupt

GPIO_EXT_PORT
Address: Operational Base + offset(0x0050)
I/O Port external port register

bit	Attr	Reset Value	Description
31:0	R	0x00	When I/O Port is configured as Input, then reading this location reads the values on the signal. When the data direction of I/O Port is set as Output, reading this

			location reads the data register for I/O Port.

GPIO_LS_SYNC
Address: Operational Base + offset(0x0060)
Level_sensitive synchronization enable register

bit	Attr	Reset Value	Description
31:1	-	-	Reserved
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

19.4 Function Description

19.4.1 Operation

Control Mode(software)

Under software control, the data and direction control for the signal are sourced from the data register (GPIO_SWPORTA_DR) and direction control register (GPIO_SWPORTA_DDR).

The direction of the external I/O pad is controlled by a write to the Port data direction register (GPIO_SWPORTA_DDR). The data written to this memory-mapped register gets mapped onto an output signal, GPIO_PORTA_DDR, of the GPIO peripheral. This output signal controls the direction of an external I/O pad.

The data written to the Porta data register (GPIO_SWPORTA_DR) drives the output buffer of the I/O pad. External data are input on the external data signal, GPIO_EXT_PORTA. Reading the external signal register(GPIO_EXT_PORTA) shows the value on the signal, regardless of the direction. This register is read-only, meaning that it cannot be written from the APB software interface.

Reading External Signals

The data on the GPIO_EXT_PORTA 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_PORTA.

An APB read to the GPIO_EXT_PORTA register yields a value equal to that which is on the GPIO_EXT_PORTA 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.

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_INT_RAWSTATUS 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.

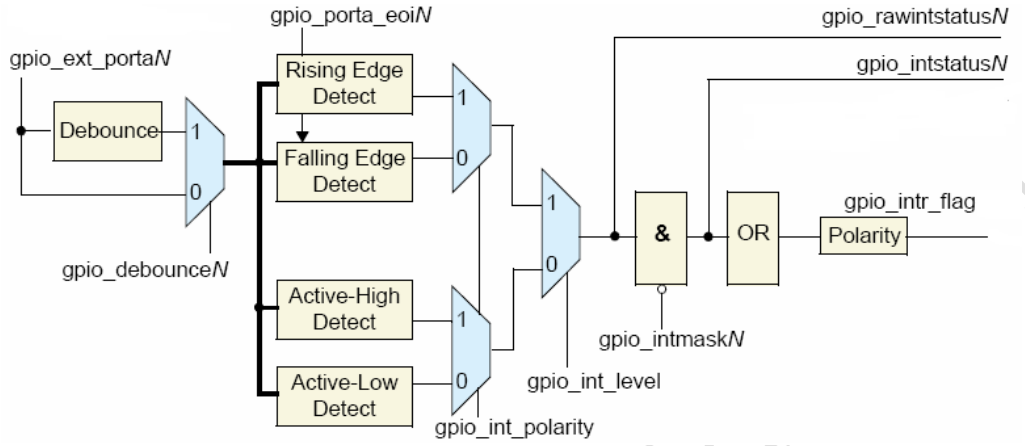


Fig 19-2 GPIO Interrupt RTL Block Diagram

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 topclk must occur for edge-detect signals. With level-sensitive interrupts, synchronization is optional and under software control(GPIO_LS_SYNC).

19.4.2 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 20 Watchdog

20.1 Overview

Watchdog Timer (WDT) is an APB slave peripheral that can be used to prevent system lockup that may be caused by conflicting parts or programs in a SoC. The WDT would generate interrupt or reset signal when its counter reaches zero, then a reset controller would reset the system.

WDT supports the following 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:
  - Generate a system reset
  - 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

20.2 Block Diagram

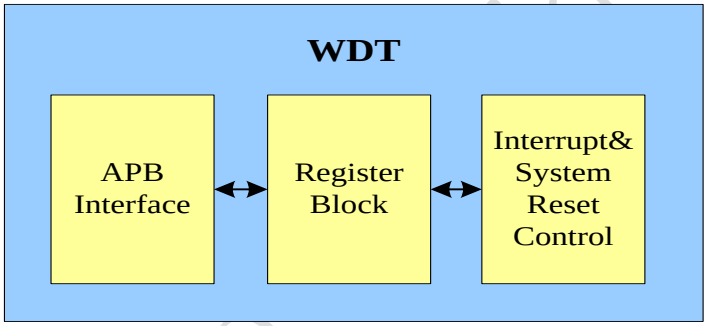


Fig 20-1 WDT block diagram

Block Descriptions:

APB Interface

The APB Interface implements the APB slave operation. Its data bus width is 32 bits.

Register Block

A register block that read coherence for the current count register.

Interrupt & system reset control

An interrupt/system reset generation block comprising of a decrementing counter and control logic.

20.3 Function description

20.3.1 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 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

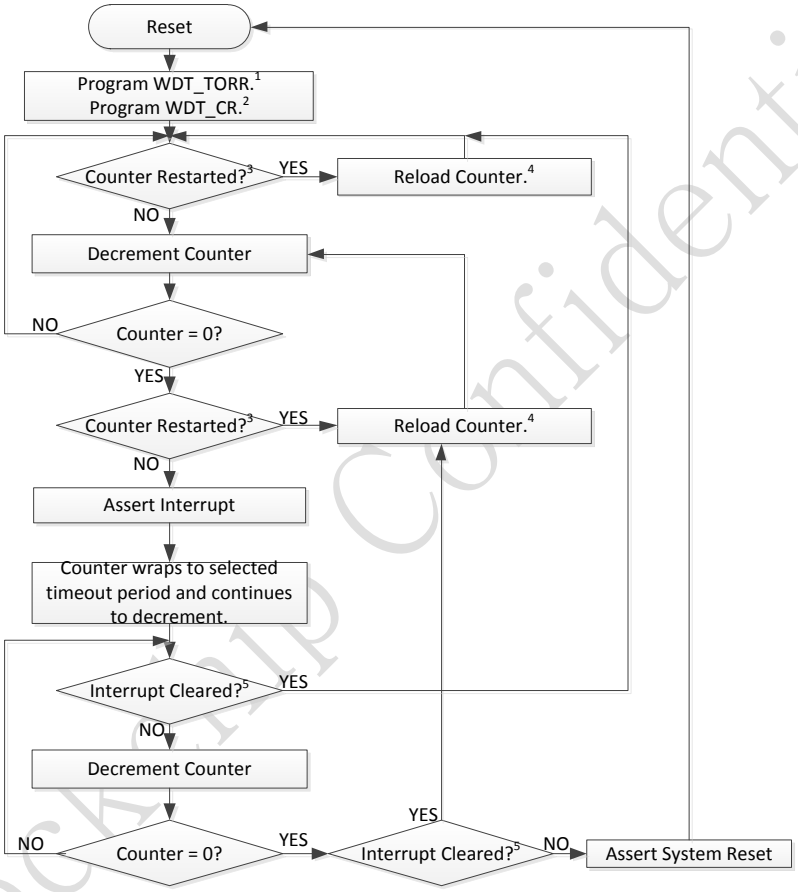
When a 0 is written to the output response mode field (RMOD, bit 1) of the Watchdog Timer Control Register (WDT_CR), the WDT generates a system reset when a timeout occurs.

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.

20.3.2 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 20-2 WDT Operation Flow

20.4 Register Description

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

20.4.1 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	0x00000000	Current counter value Register
WDT_CRR	0x000c	W	0x00000000	Counter restart Register

Name	Offset	Size	Reset Value	Description
WDT_STAT	0x0010	W	0x00000000	Interrupt status Register
WDT_EOI	0x0014	W	0x00000000	Interrupt clear Register

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

20.4.2 Detail Register Description

WDT_CR

Address: Operational Base + offset (0x0000)

Control Register

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:2	RW	0x2	rst_pluse_lenth Reset pulse length. This is used to select the number of pclk cycles for which the system reset stays asserted. 3'b000: 2 pclk cycles 3'b001: 4 pclk cycles 3'b010: 8 pclk cycles 3'b011: 16 pclk cycles 3'b100: 32 pclk cycles 3'b101: 64 pclk cycles 3'b110: 128 pclk cycles 3'b111: 256 pclk cycles
1	RW	0x1	resp_mode Response mode. Selects the output response generated to a timeout. 1'b0: Generate a system reset 1'b1: 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_en WDT enable 1'b0: WDT disabled 1'b1: WDT enabled

WDT_TORR

Address: Operational Base + offset (0x0004)

Timeout range Register

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3:0	RW	0x0	timeout_period 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: 4'b0000: 0x0000ffff 4'b0001: 0x0001ffff 4'b0010: 0x0003ffff 4'b0011: 0x0007ffff 4'b0100: 0x000fffff 4'b0101: 0x001fffff 4'b0110: 0x003fffff 4'b0111: 0x007fffff 4'b1000: 0x00ffffff 4'b1001: 0x01ffffff 4'b1010: 0x03ffffff 4'b1011: 0x07ffffff 4'b1100: 0x0fffffff 4'b1101: 0x1fffffff 4'b1110: 0x3fffffff 4'b1111: 0x7fffffff

WDT_CCVR
Address: Operational Base + offset (0x0008)
Current counter value Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	cur_cnt Current counter value This register, when read, is the current value of the internal counter. This value is read coherently whenever it is read

WDT_CRR
Address: Operational Base + offset (0x000c)
Counter restart Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:0	W1C	0x00	cnt_restart Counter restart 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 (0x0010)
Interrupt status Register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	wdt_status This register shows the interrupt status of the WDT. 1'b1: Interrupt is active regardless of polarity. 1'b0: Interrupt is inactive.

WDT_EOI
Address: Operational Base + offset (0x0014)
Interrupt clear Register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	wdt_int_clr Clears the watchdog interrupt. This can be used to clear the interrupt without restarting the watchdog counter.

Chapter 21 Pulse Width Modulation(PWM)

21.1 Overview

The pulse-width modulator (PWM) feature is very common in embedded systems. It provides a way to generate a pulse periodic waveform for motor control or can act as a digital-to-analog converter with some external components.

21.1.1 Features

The PWM Module supports the following features:

- 5-built-in PWM channels
- Configurable to operate in capture mode
  - Measures the high/low polarity effective cycles of this input waveform
  - Generates a single interrupt at the transition of input waveform polarity
  - 32-bit high polarity capture register
  - 32-bit low polarity capture register
  - 32-bit current value register
- Configurable to operate in continuous mode or one-shot mode
  - 32-bit period counter
  - 32-bit duty register
  - 32-bit current value register
  - Configurable PWM output polarity in inactive state and duty period pulse polarity
  - Period and duty cycle are shadow buffered. Change takes effect when the end of the effective period is reached or when the channel is disabled
  - Programmable center or left aligned outputs, and change takes effect when the end of the effective period is reached or when the channel is disabled
  - 8-bit repeat counter for one-shot operation. One-shot operation will produce N + 1 periods of the waveform, where N is the repeat counter value, and generates a single interrupt at the end of operation
  - Continuous mode generates the waveform continuously, and do not generates any interrupts
- pre-scaled operation to bus clock and then further scaled
- Available low-power mode to reduce power consumption when the channel is inactive
- PWM0 / PWM1 are in peripheral sub-system, only 1 channel of PWM1 is used, so, 5 PWM channels totally are used

21.2 Block Diagram

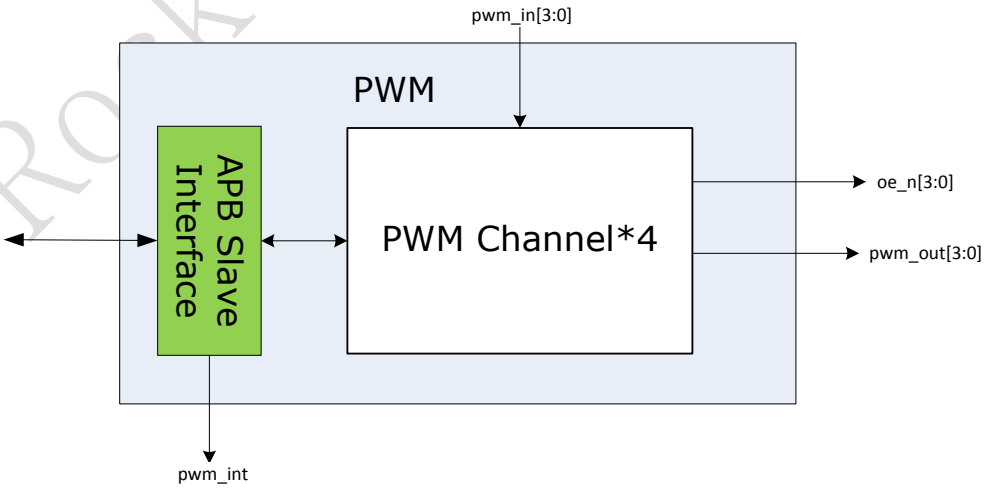


Fig 21-1 PWM architecture

21.2.1 PWM APB Slave Interface

The host processor gets access to PWM Register Block through the APB slave interface with 32-bit bus width, and asserts the active-high level interrupt.

21.2.2 PWM Channels

This is the control logic of PWM module, and controls the operation of PWM module according to the configured working mode.

21.3 Function description

The PWM supports three operation modes: reference mode, one-shot mode and continuous mode. For the one-shot mode and the continuous mode, the PWM output can be configured as the left-aligned mode or the center-aligned mode.

21.3.1 Reference mode

The reference mode is used to measure the PWM channel input waveform high/low effective cycles with the PWM channel clock, and asserts an interrupt when the polarity of the input waveform changes. The number of the high effective cycles is recorded in the PWMx_PERIOD_HPC register, while the number of the low effective cycles is recorded in the PWMx_DUTY_LPC register.

Note: the PWM input waveform is doubled buffered when the PWM channel is working in order to filter unexpected shot-time polarity transition, and therefore the interrupt is asserted several cycles after the input waveform polarity changes, and so does the change of the values of PWMx_PERIOD_HPC and PWMx_DUTY_LPC.

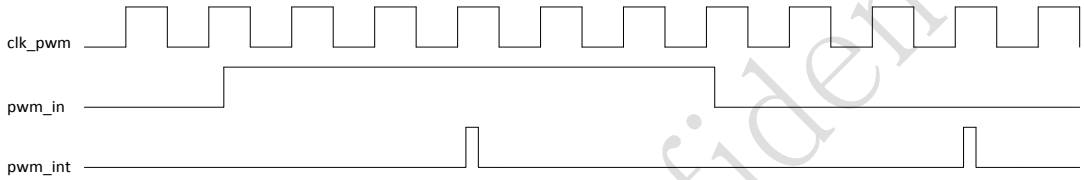


Fig 21-2 PWM Reference Mode

21.3.2 Continuous Mode

The PWM channel generates a series of the pulses continuously as expected once the channel is enabled with continuous mode.

In the continuous mode, the PWM output waveforms can be in one form of the two output mode: left-aligned mode or center-aligned mode.

For the left-aligned output mode, the PWM channel firstly starts the duty cycle with the configured duty polarity (PWMx_CTRL.duty_pol). Once duty cycle number (PWMx_DUTY_LPC) is reached, the output is switched to the opposite polarity. After the period number (PWMx_PERIOD_HPC) is reached, the output is again switched to the opposite polarity to start another period of desired pulse.

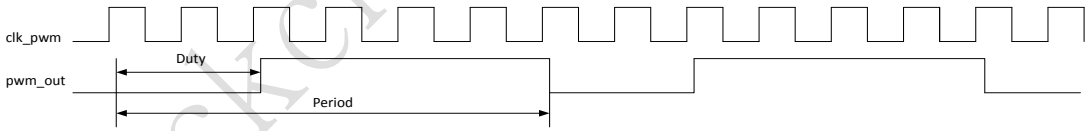


Fig 21-3 PWM Left-aligned Output Mode

For the center-aligned output mode, the PWM channel firstly starts the duty cycle with the configured duty polarity (PWMx_CTRL.duty_pol). Once one half of duty cycle number (PWMx_DUTY_LPC) is reached, the output is switched to the opposite polarity. Then if there is one half of duty cycle left for the whole period, the output is again switched to the opposite polarity. Finally after the period number (PWMx_PERIOD_HPC) is reached, the output is again switched to the opposite polarity to start another period of desired pulse.

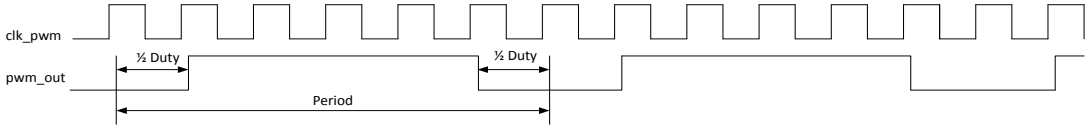


Fig 21-4 PWM Center-aligned Output Mode

Disable the PWM channel, the channel stops generating the output waveforms and output polarity is fixed as the configured inactive polarity (PWMx_CTRL.inactive_pol).

21.3.3 One-shot Mode

Unlike the continuous mode, the PWM channel generates the output waveforms within the configured periods (PWM_CTRL.rpt + 1), and then stops. At the same times, an interrupt is asserted to inform that the operation has been finished.

There are also two output modes for the one-shot mode: the left-aligned mode and the center-aligned mode.

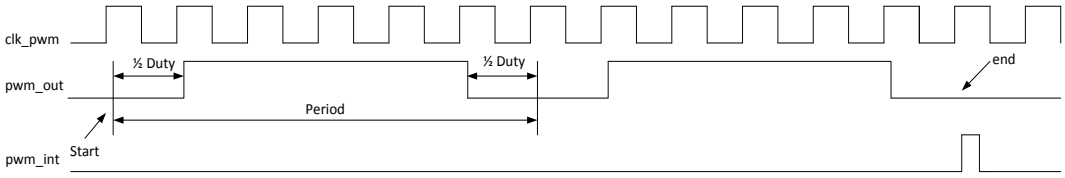


Fig 21-5 PWM Center-aligned Output Mode

21.4 Register description

21.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
PWM_PWM0_CNT	0x0000	W	0x00000000	PWM Channel 0 Counter Register
PWM_PWM0_PERIOD_HPR	0x0004	W	0x00000000	PWM Channel 0 Period Register/High Polarity Capture Register
PWM_PWM0_DUTY_LPR	0x0008	W	0x00000000	PWM Channel 0 Duty Register/Low Polarity Capture Register
PWM_PWM0_CTRL	0x000c	W	0x00000000	PWM Channel 0 Control Register
PWM_PWM1_CNT	0x0010	W	0x00000000	PWM Channel 1 Counter Register
PWM_PWM1_PERIOD_HPR	0x0014	W	0x00000000	PWM Channel 1 Period Register/High Polarity Capture Register
PWM_PWM1_DUTY_LPR	0x0018	W	0x00000000	PWM Channel 1 Duty Register/Low Polarity Capture Register
PWM_PWM1_CTRL	0x001c	W	0x00000000	PWM Channel 1 Control Register
PWM_PWM2_CNT	0x0020	W	0x00000000	PWM Channel 2 Counter Register
PWM_PWM2_PERIOD_HPR	0x0024	W	0x00000000	PWM Channel 2 Period Register/High Polarity Capture Register
PWM_PWM2_DUTY_LPR	0x0028	W	0x00000000	PWM Channel 2 Duty Register/Low Polarity Capture Register
PWM_PWM2_CTRL	0x002c	W	0x00000000	PWM Channel 2 Control Register
PWM_PWM3_CNT	0x0030	W	0x00000000	PWM Channel 3 Counter Register
PWM_PWM3_PERIOD_HPR	0x0034	W	0x00000000	PWM Channel 3 Period Register/High Polarity Capture Register
PWM_PWM3_DUTY_LPR	0x0038	W	0x00000000	PWM Channel 3 Duty Register/Low Polarity Capture Register
PWM_PWM3_CTRL	0x003c	W	0x00000000	PWM Channel 3 Control Register
PWM_INTSTS	0x0040	W	0x00000000	Interrupt Status Register
PWM_INT_EN	0x0044	W	0x00000000	Interrupt Enable Register

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

21.4.2 Detail Register Description

PWM_PWM0_CNT

Address: Operational Base + offset (0x0000)

PWM Channel 0 Counter Register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 0 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM0_PERIOD_HPR
Address: Operational Base + offset (0x0004)
PWM Channel 0 Period Register/High Polarity Capture Register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM0_DUTY_LPR
Address: Operational Base + offset (0x0008)
PWM Channel 0 Duty Register/Low Polarity Capture Register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM0_CTRL
Address: Operational Base + offset (0x000c)
PWM Channel 0 Control Register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This field defines the scale factor applied to prescaled clock. The value N means the clock is divided by 2*N. If N is 0, it means that the clock is divided by 512(2*256).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by 2^N.
11:10	RO	0x0	reserved

Bit	Attr	Reset Value	Description
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt. 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

PWM_PWM1_CNT
Address: Operational Base + offset (0x0010)
PWM Channel 1 Counter Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 1 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM1_PERIOD_HPR
Address: Operational Base + offset (0x0014)
PWM Channel 1 Period Register/High Polarity Capture Register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM1_DUTY_LPR
Address: Operational Base + offset (0x0018)
PWM Channel 1 Duty Register/Low Polarity Capture Register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM1_CTRL
Address: Operational Base + offset (0x001c)
PWM Channel 1 Control Register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This field defines the scale factor applied to prescaled clock. The value N means the clock is divided by 2*N. If N is 0, it means that the clock is divided by 512(2*256).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by 2^N.
11:10	RO	0x0	reserved

Bit	Attr	Reset Value	Description
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

PWM_PWM2_CNT
Address: Operational Base + offset (0x0020)
PWM Channel 2 Counter Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 2 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM2_PERIOD_HPR
Address: Operational Base + offset (0x0024)
PWM Channel 2 Period Register/High Polarity Capture Register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM2_DUTY_LPR
Address: Operational Base + offset (0x0028)
PWM Channel 2 Duty Register/Low Polarity Capture Register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM2_CTRL
Address: Operational Base + offset (0x002c)
PWM Channel 2 Control Register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This field defines the scale factor applied to prescaled clock. The value N means the clock is divided by 2*N. If N is 0, it means that the clock is divided by 512(2*256).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by 2^N.
11:10	RO	0x0	reserved

Bit	Attr	Reset Value	Description
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt. 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

PWM_PWM3_CNT
Address: Operational Base + offset (0x0030)
PWM Channel 3 Counter Register

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	CNT Timer Counter The 32-bit indicates current value of PWM Channel 3 counter. The counter runs at the rate of PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM3_PERIOD_HPR
Address: Operational Base + offset (0x0034)
PWM Channel 3 Period Register/High Polarity Capture Register

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	PERIOD_LPR Output Waveform Period/Input Waveform High Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the period of the output waveform. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the bit [31:1] is taken into account and bit [0] always considered as 0. If PWM is operated at the capture mode, this value indicates the effective high polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM3_DUTY_LPR
Address: Operational Base + offset (0x0038)
PWM Channel 3 Duty Register/Low Polarity Capture Register

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DUTY_LPR Output Waveform Duty Cycle/Input Waveform Low Polarity Cycle If PWM is operated at the continuous mode or one-shot mode, this value defines the duty cycle of the output waveform. The PWM starts the output waveform with duty cycle. Note that, if the PWM is operated at the center-aligned mode, the period should be an even one, and therefore only the [31:1] is taken into account. If PWM is operated at the capture mode, this value indicates the effective low polarity cycles of input waveform. This value is based on the PWM clock. The value ranges from 0 to (2^32-1).

PWM_PWM3_CTRL
Address: Operational Base + offset (0x003c)
PWM Channel 3 Control Register

Bit	Attr	Reset Value	Description
31:24	RW	0x00	rpt Repeat Counter This field defines the repeated effective periods of output waveform in one-shot mode. The value N means N+1 repeated effective periods.
23:16	RW	0x00	scale Scale Factor This field defines the scale factor applied to prescaled clock. The value N means the clock is divided by 2*N. If N is 0, it means that the clock is divided by 512(2*256).
15	RO	0x0	reserved
14:12	RW	0x0	prescale Prescale Factor This field defines the prescale factor applied to input clock. The value N means that the input clock is divided by 2^N.
11:10	RO	0x0	reserved

Bit	Attr	Reset Value	Description
9	RW	0x0	clk_sel Clock Source Select 0: non-scaled clock is selected as PWM clock source. It means that the prescale clock is directly used as the PWM clock source 1: scaled clock is selected as PWM clock source
8	RW	0x0	lp_en Low Power Mode Enable 0: disabled 1: enabled When PWM channel is inactive state and Low Power Mode is enabled, the path to PWM Clock prescale module is blocked to reduce power consumption.
7:6	RO	0x0	reserved
5	RW	0x0	output_mode PWM Output mode 0: left aligned mode 1: center aligned mode
4	RW	0x0	inactive_pol Inactive State Output Polarity This defines the output waveform polarity when PWM channel is in inactive state. The inactive state means that PWM finishes the complete waveform in one-shot mode or PWM channel is disabled. 0: negative 1: positive
3	RW	0x0	duty_pol Duty Cycle Output Polarity This defines the polarity for duty cycle. PWM starts the output waveform with duty cycle. 0: negative 1: positive
2:1	RW	0x0	pwm_mode PWM Operation Mode 00: One shot mode. PWM produces the waveform within the repeated times defined by PWMx_CTRL_rpt 01: Continuous mode. PWM produces the waveform continuously 10: Capture mode. PWM measures the cycles of high/low polarity of input waveform. 11: reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	pwm_en PWM channel enable 0: disabled 1: enabled. If the PWM is worked the one-shot mode, this bit will be cleared at the end of operation

PWM_INTSTS
Address: Operational Base + offset (0x0040)
Interrupt Status Register

Bit	Attr	Reset Value	Description
31:12	RO	0x0	reserved
11	RO	0x0	CH3_Pol Channel 3 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM3_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM3_PERIOD_HPR to know the effective low cycle of Channel 3 input waveform. Write 1 to CH3_IntSts will clear this bit.
10	RO	0x0	CH2_Pol Channel 2 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM2_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM2_PERIOD_HPR to know the effective low cycle of Channel 2 input waveform. Write 1 to CH2_IntSts will clear this bit.
9	RO	0x0	CH1_Pol Channel 1 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM1_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM1_PERIOD_HPR to know the effective low cycle of Channel 1 input waveform. Write 1 to CH1_IntSts will clear this bit.

Bit	Attr	Reset Value	Description
8	RO	0x0	CH0_Pol Channel 0 Interrupt Polarity Flag This bit is used in capture mode in order to identify the transition of the input waveform when interrupt is generated. When bit is 1, please refer to PWM0_PERIOD_HPR to know the effective high cycle of Channel 0 input waveform. Otherwise, please refer to PWM0_PERIOD_HPR to know the effective low cycle of Channel 0 input waveform. Write 1 to CH0_IntSts will clear this bit.
7:4	RO	0x0	reserved
3	RW	0x0	CH3_IntSts Channel 3 Interrupt Status 0: Channel 3 Interrupt not generated 1: Channel 3 Interrupt generated
2	RW	0x0	CH2_IntSts Channel 2 Interrupt Status 0: Channel 2 Interrupt not generated 1: Channel 2 Interrupt generated
1	RW	0x0	CH1_IntSts Channel 1 Interrupt Status 0: Channel 1 Interrupt not generated 1: Channel 1 Interrupt generated
0	RW	0x0	CH0_IntSts Channel 0 Raw Interrupt Status 0: Channel 0 Interrupt not generated 1: Channel 0 Interrupt generated

PWM_INT_EN
Address: Operational Base + offset (0x0044)
Interrupt Enable Register

Bit	Attr	Reset Value	Description
31:4	RO	0x0	reserved
3	RW	0x0	CH3_Int_en Channel 3 Interrupt Enable 0: Channel 3 Interrupt disabled 1: Channel 3 Interrupt enabled
2	RW	0x0	CH2_Int_en Channel 2 Interrupt Enable 0: Channel 2 Interrupt disabled 1: Channel 2 Interrupt enabled

Bit	Attr	Reset Value	Description
1	RW	0x0	CH1_Int_en Channel 1 Interrupt Enable 0: Channel 1 Interrupt disabled 1: Channel 1 Interrupt enabled
0	RW	0x0	CH0_Int_en Channel 0 Interrupt Enable 0: Channel 0 Interrupt disabled 1: Channel 0 Interrupt enabled

21.5 Interface Description

Module Pin	Direction	Pad Name	IOMUX Setting
pwm4	O	IO_PWM4out_I2C2Asda_EBCgdpwr0_GPIOP2a0	GRF_GPIO2A_IOMUX[1:0]= 2'b01
pwm3	O	IO_PWM3out_I2C2Ascl_EBCsdce0_GPIOP2a1	GRF_GPIO2A_IOMUX[3:2]= 2'b01
pwm2	O	IO_PWM2out_EBCgdpwr2_PMUst3_GPIOP2a2	GRF_GPIO2A_IOMUX[5:4]= 2'b01
pwm1	O	IO_PWM1out_CLKobs_EBCgdpwr1_GPIOP2a3	GRF_GPIO2A_IOMUX[7:6]= 2'b01
pwm0	O	IO_PWM0out_JTG0tdi_PMUst2_GPIOP2a4	GRF_GPIO2A_IOMUX[9:8]= 2'b01

21.6 Application Notes

21.6.1 PWM Reference Mode Standard Usage Flow

1. Set PWMx_CTRL.pwm_en to '0' to disable the PWM channel.
2. Choose the prescale factor and the scale factor for pclk by programming PWMx_CTRL.prescale and PWMx_CTRL.scale, and select the clock needed by setting PWMx_CTRL.clk_sel.
3. Configure the channel to work in the reference mode.
4. Enable the INT_EN.chx_int_en to enable the interrupt generation.
5. Enable the channel by writing'1' to PWMx_CTRL.pwm_en bit to start the channel.
6. When an interrupt is asserted, refer to INTSTS register to know the raw interrupt status. If the corresponding polarity flag is set, turn to PWMx_PERIOD_HPC register to know the effective high cycles of input waveforms, otherwise turn to PWMx_DUTY_LPC register to know the effective low cycles.
7. Write '0' to PWMx_CTRL.pwm_en to disable the channel.

21.6.2 PWM One-shot Mode/Continuous Standard Usage Flow

1. Set PWMx_CTRL.pwm_en to '0' to disable the PWM channel.
2. Choose the prescale factor and the scale factor for pclk by programming PWMx_CTRL.prescale and PWMx_CTRL.scale, and select the clock needed by setting PWMx_CTRL.clk_sel.
3. Choose the output mode by setting PWMx_CTRL.output_mode, and set the duty polarity and inactive polarity by programming PWMx_CTRL.duty_pol and PWMx_CTRL.inactive_pol.
4. Set the PWMx_CTRL.rpt if the channel is desired to work in the one-shot mode;
5. Configure the channel to work in the one-shot mode or the continuous mode.
6. Enable the INT_EN.chx_int_en to enable the interrupt generation if the channel is desired to work in the one-shot mode;
7. If the channel is working in the one-shot mode, an interrupt is asserted after the end of operation, and the PWMx_CTRL.pwm_en is automatically cleared. Whatever mode the channel is working, write '0' to PWMx_CTRL.pwm_en bit to disable the PWM channel.

21.6.3 Low-power mode

Setting PWMx_CTRL.lp_en to '1' makes the channel enter the low-power mode. When then

PWM channel is inactive, the APB bus clock to the PWM channel is gated in order to reduce the power consumption. It is recommended to disable the channel before entering the low-power mode, and quit the low-power mode before enabling the channel.

21.6.4 Other notes

When the channel is active to produce waveforms, it is free to programming the PWMx_PERIOD_HPC and PWMx_DUTY_LPC register. The change will not take effect immediately until the current period ends.

An active channel can be changed to another operation mode without disable the PWM channel. However, during the transition of the operation mode there may be some irregular output waveforms.

If the PWM operational frequency is desired to changed, it is recommended to disable the channel first, and then make the channel enter the low-power mode to gate the clock. It is free to change clock setting. After clock setting is changed, quit the lower-power mode and enable the channel to take the change into effect.

Chapter 22 I2C Interface

22.1 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. This I2C bus controller supports master mode acting as a bridge between AMBA protocol and generic I2C bus system.

22.1.1 Features

I2C Controller supports the following features:

- Item Compatible with I2C-bus
- AMBA APB slave interface
- Supports master mode of I2C bus
- Software programmable clock frequency and transfer rate up to 400Kbit/sec
- Supports 7 bits and 10 bits addressing modes
- Interrupt or polling driven multiple bytes data transfer
- Clock stretching and wait state generation
- I2C0 / I2C1 / I2C2 are in peripheral sub-system

22.2 Block Diagram

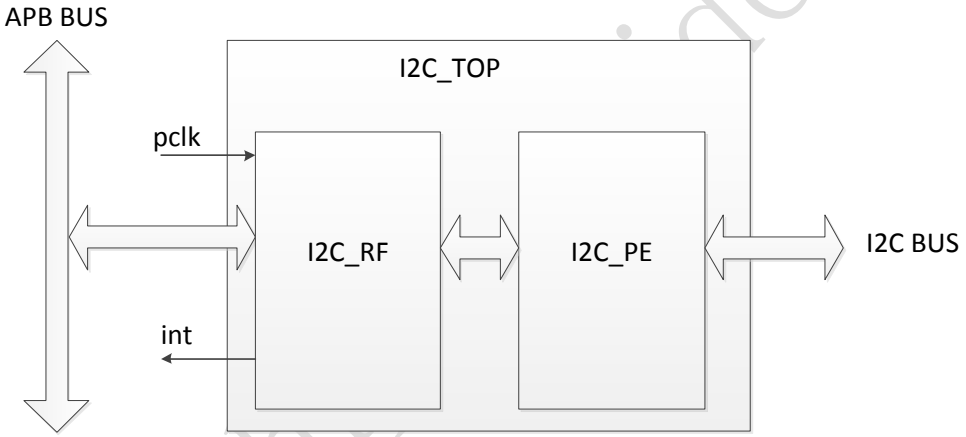


Fig 22-1 I2C architecture

I2C_RF

I2C_RF module 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_PE

I2C_PE module 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_TOP

I2C_TOP module is the top module of the I2C controller.

22.3 Function description

This chapter provides a description about the functions and behavior under various conditions. The I2C controller supports only Master function. It 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 2 parts and described separately: initialization and master mode programming.

22.3.1 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 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 as the system clock so the APB clock will determine the I2C bus clock. The correct register setting is subject to the system requirement.

22.3.2 Master Mode Programming

1. SCL Clock: When the I2C controller is programmed in Master mode, the SCL frequency is determine by I2C_CLKDIV register. The SCL frequency is calculated by the following formula：
SCL Divisor = 8*((CLKDIVL+1) + (CLKDIVH+1))
SCL = PCLK/ SCLK Divisor

2. Data Receiver Register Access
When the i2c controller received MRXCNT bytes data， CPU can get the data through register RXDATA0 ~ RXDATA7. The controller can receive up to 32 byte data in one transaction.
When MRXCNT register is written, the I2C controller will start to drive SCL to receive data.

3. Transmit Transmitter Register
Data to transmit are written to TXDATA0~7 by CPU. The controller can transmit up to 32 byte data in one transaction. The lower byte will be transmitted first.
When MTXCNT register is written, the I2C controller will start to transmit data.

4. Start Command
Write 1 to I2C_CON[3], the controller will send I2C start command.

5. Stop Command
Write 1 to I2C_CON[4], the controller will send I2C stop command

6. I2C Operation mode
There are four i2c operation modes.
When I2C_CON[2:1] is 2'b00, the controller transmit all valid data in TXDATA0~TXDATA7 byte by byte. The controller will transmit lower byte first.
When I2C_CON[2:1] is 2'b01,the controller will transmit device address in MRXADDR first (Write/Read bit = 0) and then transmit device register address in MRXRADDR. After that, the controller will assert restart signal and resend MRXADDR (Write/Read bit = 1). At last, the controller enter receive mode.
When I2C_CON[2:1] is 2'b10, the controller is in receive mode, it will triggered clock to read MRXCNT byte data.
When I2C_CON[2:1] is 2'b11,the controller will transmit device address in MRXADDR first (Write/Read bit = 1) and then transmit device register address in MRXRADDR . After that, the controller will assert restart signal and resend MRXADDR (Write/Read bit = 1). At last, the controller enter receive mode.

7. Read/Write Command
When I2C_OPMODE(I2C_CON[2:1]) is 2'b01 or 2'b11, the Read/Write command bit is decided by controller itself.
In RX only mode (I2C_CON[2:1] is 2'b10), the Read/Write command bit is decided by MRXADDR[0].
In TX only mode (I2C_CON[[2:1] is 2'b00), the Read/Write command bit is decided by TXDATA[0].

8. Master Interrupt Condition
There are 7 interrupt bits in I2C_ISR register related to master mode.
Byte transmitted finish interrupt (Bit 0): The bit is asserted when Master complete transmitting a byte.
Byte received finish interrupt (Bit 1): The bit is asserted when Master complete receiving a byte.
MTXCNT bytes data transmitted finish interrupt (Bit 2): The bit is asserted when Master complete transmitting MTXCNT bytes.
MRXCNT bytes data received finish interrupt (Bit 3): The bit is asserted when Master complete receiving MRXCNT bytes.
Start interrupt(Bit 4): The bit is asserted when Master finish asserting start command to I2C bus.
Stop interrupt (Bit 5): The bit is asserted when Master finish asserting stop command to I2C

bus.
NAK received interrupt (Bit 6): The bit is asserted when Master received a NAK handshake.

9. Last byte acknowledge control

If I2C_CON[5] is 1, the I2C controller will transmit NAK handshake to slave when the last byte received in RX only mode.
If I2C_CON[5] is 0, the I2C controller will transmit ACK handshake to slave when the last byte received in RX only mode.

10. How to handle nak handshake received

If I2C_CON[6] is 1, the I2C controller will stop all transactions when NAK handshake received. And the software should take responsibility to handle the problem.
If I2C_CON[6] is 0, the I2C controller will ignore all NAK handshake received.

11. 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.

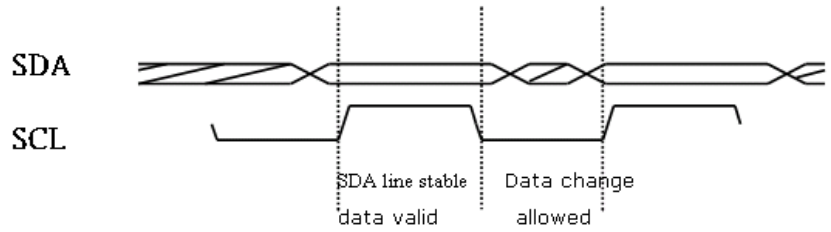


Fig 22-2 I2C DATA Validity

(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.

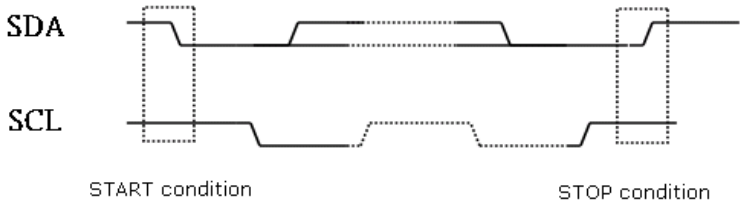


Fig 22-3 I2C Start and stop conditions

- **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".

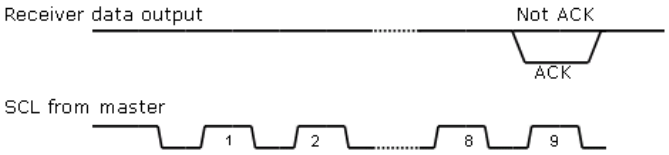


Fig 22-4 I2C Acknowledge

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

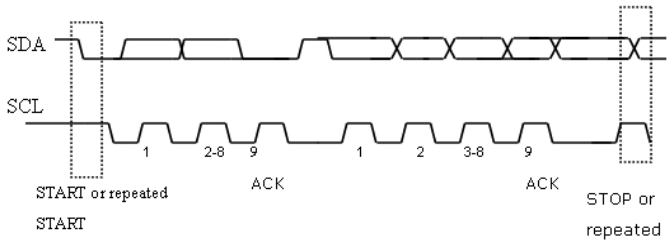


Fig 22-5 I2C byte transfer

22.4 Register Description

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

22.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
I2C_CON	0x0000	W	0x00000000	control register
I2C_CLKDIV	0x0004	W	0x00060006	Clock divisor register
I2C_MRXADDR	0x0008	W	0x00000000	the slave address accessed for master receive mode
I2C_MRXRADDR	0x000c	W	0x00000000	the slave register address accessed for master receive mode
I2C_MTXCNT	0x0010	W	0x00000000	master transmit count
I2C_MRXCNT	0x0014	W	0x00000000	master receive count
I2C_IEN	0x0018	W	0x00000000	interrupt enable register
I2C_IPD	0x001c	W	0x00000000	interrupt pending register
I2C_FCNT	0x0020	W	0x00000000	finished count
I2C_TXDATA0	0x0100	W	0x00000000	I2C transmit data register 0
I2C_TXDATA1	0x0104	W	0x00000000	I2C transmit data register 1
I2C_TXDATA2	0x0108	W	0x00000000	I2C transmit data register 2
I2C_TXDATA3	0x010c	W	0x00000000	I2C transmit data register 3
I2C_TXDATA4	0x0110	W	0x00000000	I2C transmit data register 4
I2C_TXDATA5	0x0114	W	0x00000000	I2C transmit data register 5
I2C_TXDATA6	0x0118	W	0x00000000	I2C transmit data register 6
I2C_TXDATA7	0x011c	W	0x00000000	I2C transmit data register 7
I2C_RXDATA0	0x0200	W	0x00000000	I2C receive data register 0
I2C_RXDATA1	0x0204	W	0x00000000	I2C receive data register 1
I2C_RXDATA2	0x0208	W	0x00000000	I2C receive data register 2
I2C_RXDATA3	0x020c	W	0x00000000	I2C receive data register 3
I2C_RXDATA4	0x0210	W	0x00000000	I2C receive data register 4
I2C_RXDATA5	0x0214	W	0x00000000	I2C receive data register 5
I2C_RXDATA6	0x0218	W	0x00000000	I2C receive data register 6
I2C_RXDATA7	0x021c	W	0x00000000	I2C receive data register 7

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

22.4.2 Detail Register Description

I2C_CON

Address: Operational Base + offset (0x0000)

control register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved

Bit	Attr	Reset Value	Description
6	RW	0x0	act2nak operation when NAK handshake is received 1'b0: ignored 1'b1: stop transaction
5	RW	0x0	ack last byte acknowledge control last byte acknowledge control in master receive mode . 1'b0: ACK 1'b1: NAK
4	W1C	0x0	stop stop enable when this bit is written to 1，I2C will generate stop signal. It cleared itself when stop operation ends.
3	W1C	0x0	start start enable when this bit is written to 1，I2C will generate start signal. It cleared itself when start operation ends.
2:1	RW	0x0	i2c_mode 2'b00: transmit only 2'b01: transmit address (device + register address) --> restart -->transmit address –>receive only 2'b10:receive only 2'b11: transmit address (device + register address, write/read bit is 1) --> restart -->transmit address (device address) -->receive data
0	RW	0x0	i2c_en i2c module enable 1: i2c is enabled. 0: i2c is disabled.

I2C_CLKDIV
Address: Operational Base + offset (0x0004)
Clock divisor register

Bit	Attr	Reset Value	Description
31:16	RW	0x0006	CLKDIVH SCL high level clock count T（SCL_HIGH）= T(PCLK) *(CLKDIVH+1) * 8
15:0	RW	0x0006	CLKDIVL SCL low level clock count T（SCL_LOW）= T(PCLK) *(CLKDIVL+1) * 8

I2C_MRXADDR
Address: Operational Base + offset (0x0008)
the slave address accessed for master receive mode

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26	RW	0x0	addhvld address high byte valid
25	RW	0x0	addmvld address middle byte valid
24	RW	0x0	addlvld address low byte valid
23:0	RW	0x000000	saddr master address register the lowest bit indicate write or read

I2C_MRXRADDR
Address: Operational Base + offset (0x000c)
the slave register address accessed for master receive mode

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26	RW	0x0	sraddhvld address high byte valid
25	RW	0x0	sraddmvld address middle byte valid
24	RW	0x0	sraddlvld address low byte valid
23:0	RW	0x000000	sraddr slave register address accessed

I2C_MTXCNT
Address: Operational Base + offset (0x0010)
master transmit count

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RW	0x00	mtxcnt master transmit count

I2C_MRXCNT
Address: Operational Base + offset (0x0014)
Masterreceive count

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved
5:0	RW	0x00	mrxcnt master receive count

I2C_IEN
Address: Operational Base + offset (0x0018)
interrupt enable register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	nakrcvien NAK handshake received interrupt enable
5	RW	0x0	stopien stop operation finished interrupt enable
4	RW	0x0	startien start operation finished interrupt enable
3	RW	0x0	mbrfien MRXCNT data received finished interrupt enable
2	RW	0x0	mbtfien MTXCNT data transmit finished interrupt enable
1	RW	0x0	brfien byte receive finished interrupt enable
0	RW	0x0	btfien byte transmit finished interrupt enable

I2C_IPD
Address: Operational Base + offset (0x001c)
interrupt pending register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	nakrcvipd NAK handshake received interrupt pending bit
5	RW	0x0	stopipd stop operation finished interrupt pending bit
4	RW	0x0	startipd start operation finished interrupt pending bit
3	RW	0x0	mbrfipd MRXCNT data received finished interrupt pending bit
2	RW	0x0	mbtfipd MTXCNT data transmit finished interrupt pending bit
1	RW	0x0	brfipd byte receive finished interrupt pending bit
0	RW	0x0	btfipd byte transmit finished interrupt pending bit

I2C_FCNT
Address: Operational Base + offset (0x0020)
finished count

Bit	Attr	Reset Value	Description
31:6	RO	0x0	reserved

Bit	Attr	Reset Value	Description
5:0	RW	0x00	fcnt finished count the count of data which has been transmitted or received for debug purpose

I2C_TXDATA0
Address: Operational Base + offset (0x0100)
I2C transmit data register 0

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata0

I2C_TXDATA1
Address: Operational Base + offset (0x0104)
I2C transmit data register 1

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata1

I2C_TXDATA2
Address: Operational Base + offset (0x0108)
I2C transmit data register 2

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata2

I2C_TXDATA3
Address: Operational Base + offset (0x010c)
I2C transmit data register 3

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata3

I2C_TXDATA4
Address: Operational Base + offset (0x0110)
I2C transmit data register 4

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata4

I2C_TXDATA5
Address: Operational Base + offset (0x0114)
I2C transmit data register 5

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata5

I2C_TXDATA6
Address: Operational Base + offset (0x0118)

I2C transmit data register 6

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata6

I2C_TXDATA7

Address: Operational Base + offset (0x011c)

I2C transmit data register 7

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	txdata7

I2C_RXDATA0

Address: Operational Base + offset (0x0200)

I2C receive data register 0

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata0

I2C_RXDATA1

Address: Operational Base + offset (0x0204)

I2C receive data register 1

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata1

I2C_RXDATA2

Address: Operational Base + offset (0x0208)

I2C receive data register 2

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata2

I2C_RXDATA3

Address: Operational Base + offset (0x020c)

I2C receive data register 3

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata3

I2C_RXDATA4

Address: Operational Base + offset (0x0210)

I2C receive data register 4

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata4

I2C_RXDATA5

Address: Operational Base + offset (0x0214)

I2C receive data register 5

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata5

I2C_RXDATA6
Address: Operational Base + offset (0x0218)
I2C receive data register 6

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata6

I2C_RXDATA7
Address: Operational Base + offset (0x021c)
I2C receive data register 7

Bit	Attr	Reset Value	Description
31:0	RO	0x00000000	rxdata7

22.5 Interface description

I2C0 has 3 IOMUX, which is controlled by GRF_IOMUX_CON[1:0].

When GRF_IOMUX_CON[1:0] is 2'b00, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C0			
i2c0_sda	I/O	IO_I2C0Asda_SPI0Brx_EBCborder0_GPIOP2b3	GRF_GPIO2B_IOMUX[7:6]=2'b01
i2c0_scl	I/O	IO_I2C0Ascl_SPI0Btx_EBCsdce5_GPIOP2b2	GRF_GPIO2B_IOMUX[5:4]=2'b01

When GRF_IOMUX_CON[1:0] is 2'b01, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C0			
i2c0_sda	I/O	IO_UART1Atx_I2C0Bsda_SPI1Bclk_GPIOP2c0	GRF_GPIO2C_IOMUX[1:0]=2'b10
i2c0_scl	I/O	IO_UART1Arx_I2C0Bscl_SPI1Btx_GPIOP2c1	GRF_GPIO2C_IOMUX[3:2]=2'b10

When GRF_IOMUX_CON[1:0] is 2'b10, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C0			
i2c0_sda	I/O	IO_EMMCd2_SFCd3_I2C0Csda_GPIO0a5	GRF_GPIO0A_IOMUX[11:10]=2'b11
i2c0_scl	I/O	IO_EMMCd3_SFCd2_I2C0Cscl_GPIO0a6	GRF_GPIO0A_IOMUX[13:12]=2'b11

I2C1 has 3 IOMUX, which is controlled by GRF_IOMUX_CON[3:2].

When GRF_IOMUX_CON[3:2] is 2'b00, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C1			
i2c1_sda	I/O	IO_I2C1Asda_SPI0Bcs_EBCsdce3_GPIOP2b0	GRF_GPIO2B_IOMUX[1:0]=2'b01

i2c1_scl	I/O	IO_I2C1Ascl_SPI0Bclk_EBCborder1_GPIOP2b1	GRF_GPIO2B_IOMUX[3:2]=2'b01

When GRF_IOMUX_CON[3:2] is 2'b01, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C1			
i2c1_sda	I/O	IO_SDMMCd3_I2C1Bsda_UART5rx_GPIO1b2	GRF_GPIO1B_IOMUX[5:4]=2'b10
i2c1_scl	I/O	IO_SDMMCd2_I2C1Bscl_UART5tx_GPIO1b1	GRF_GPIO1B_IOMUX[3:2]=2'b10

When GRF_IOMUX_CON[3:2] is 2'b10, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C1			
i2c1_sda	I/O	IO_UART0Arx_JTG1tms_I2C1Csda_GPIOP2b4	GRF_GPIO2B_IOMUX[9:8]=2'b11
i2c1_scl	I/O	IO_UART0Atx_JTG1tck_I2C1Cscl_GPIOP2b5	GRF_GPIO2B_IOMUX[11:10]=2'b11

I2C2 has 3 IOMUX, which is controlled by GRF_IOMUX_CON[5:4].

When GRF_IOMUX_CON[5:4] is 2'b00, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C2			
i2c2_sda	I/O	IO_PWM4out_I2C2Asda_EBCgdpwr0_GPIOP2a0	GRF_GPIO2A_IOMUX[1:0]=2'b10
i2c2_scl	I/O	IO_PWM3out_I2C2Ascl_EBCsdce0_GPIOP2a1	GRF_GPIO2A_IOMUX[3:2]=2'b10

When GRF_IOMUX_CON[5:4] is 2'b01, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C2			
i2c2_sda	I/O	IO_LCDrs_I2C2Bsda_EBCsdoe_GPIO0d1	GRF_GPIO0D_IOMUX[3:2]=2'b10
i2c2_scl	I/O	IO_LCDwrn_I2C2Bscl_EBCsdle_GPIO0d0	GRF_GPIO0D_IOMUX[1:0]=2'b10

When GRF_IOMUX_CON[5:4] is 2'b10, the IOMUX is as follow.

Module pin	Direction	Pad name	IOMUX
I2C2			
i2c2_sda	I/O	IO_I2C2Csda_JTG0trst_PMUst0_GPIOP2a6	GRF_GPIO2A_IOMUX[13:12]=2'b01
i2c2_scl	I/O	IO_I2C2Cscl_JTG0tdo_PMUst1_GPIOP2a5	GRF_GPIO2A_IOMUX[11:10]=2'b01

22.6 Application Notes

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 3 sections, transmit only mode, receive only mode, and mix mode. Users are strongly advised

to following.

- Transmit only mode (I2C_CON[1:0]=2'b00)

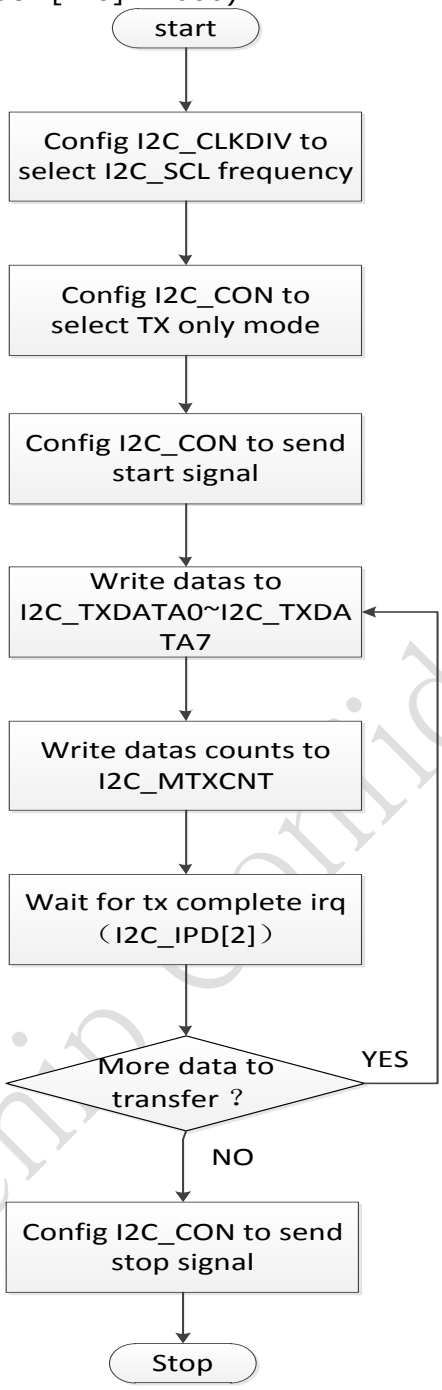


Fig 22-6 I2C Flow chat for transmit only mode

- Receive only mode (I2C_CON[1:0]=2'b10)

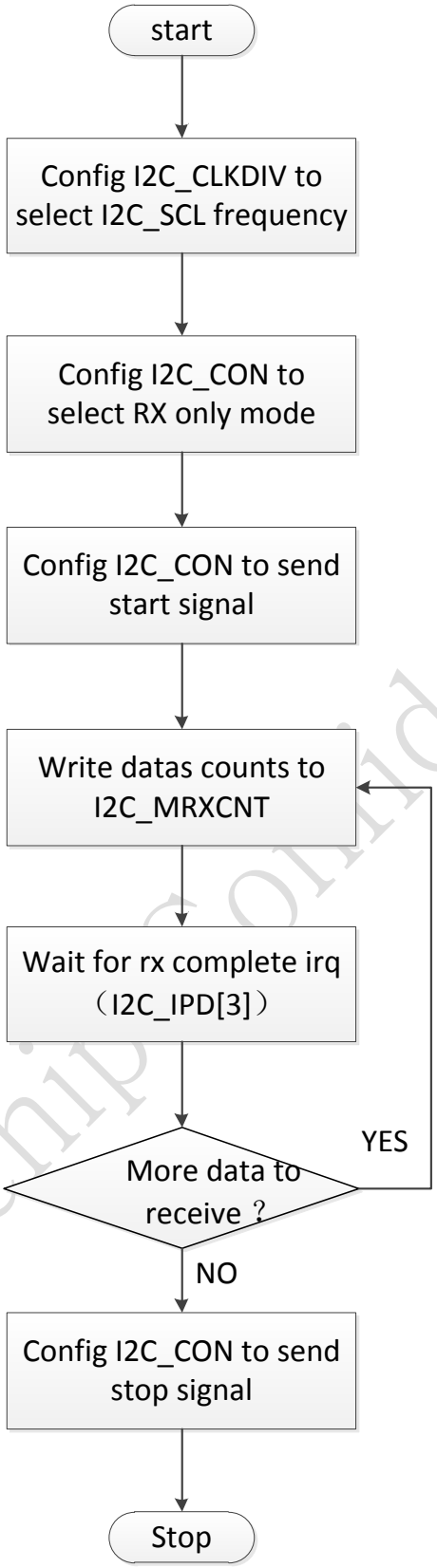


Fig 22-7 I2C Flow chart for receive only mode

- Mix mode (I2C_CON[1:0]=2'b01 or I2C_CON[1:0]=2'b11)

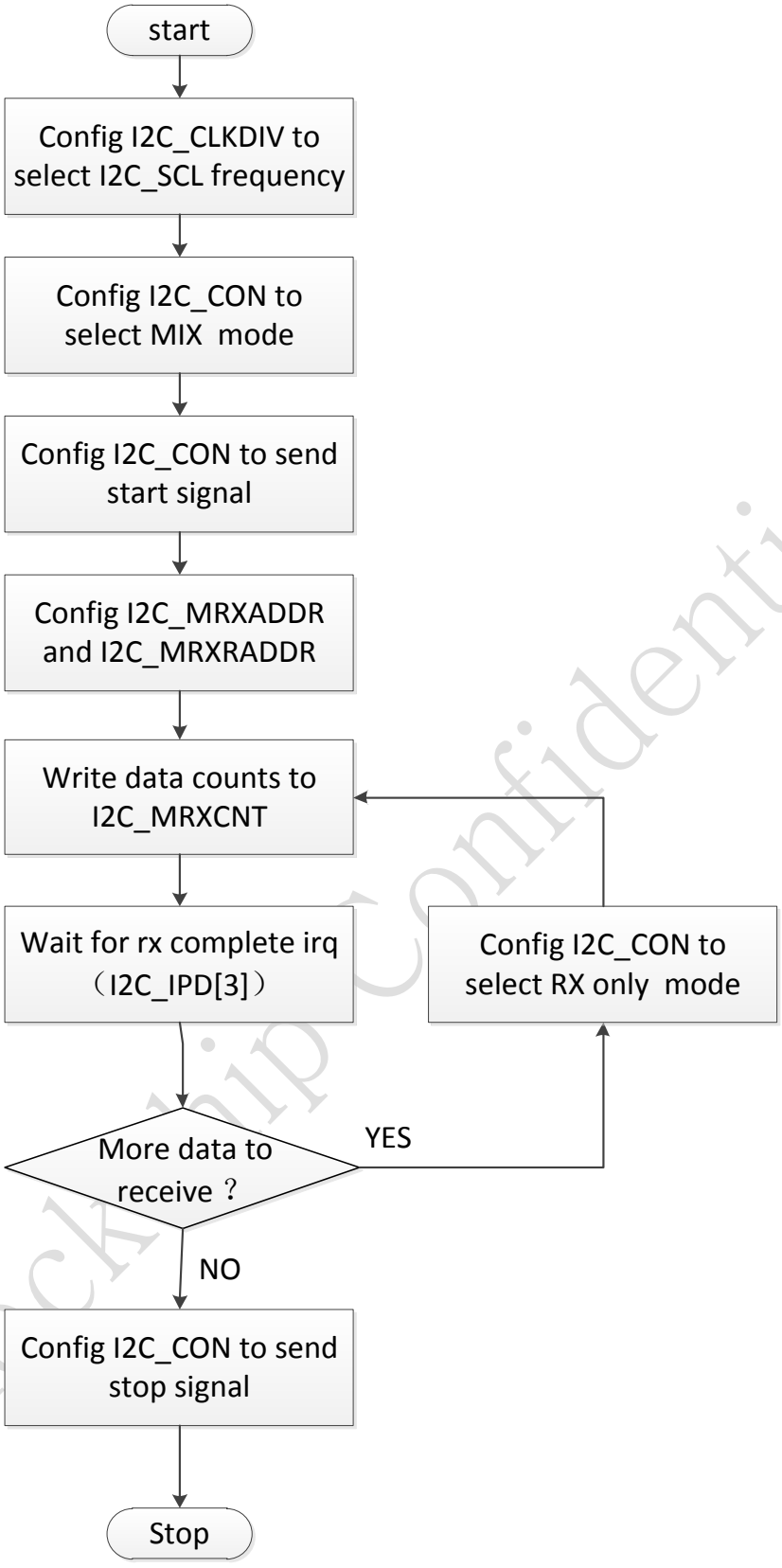


Fig 22-8 I2C Flow chart for mix mode

Chapter 23 UART

23.1 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.

23.1.1 Features

UART Controller supports the following features:

- AMBA APB interface Support interrupt interface to interrupt controller
- UART0/UART1/UART2/UART3/UART4/UART5 contains two 32Bytes FIFOs for data receive and transmit.
- Programmable serial data baud rate as calculated by the following: baud rate = (serial clock frequency)/(16×divisor)
- UART1 / UART2 support auto flow-control
- UART0 / UART1 / UART2 / UART3 / UART4 / UART5 are in peripheral subsystem

23.2 Block Diagram

This section provides a description about the functions and behavior under various conditions. The UART Controller comprises with:

- AMBA APB interface
- FIFO controllers
- Register block
- Modem synchronization block and baud clock generation block
- Serial receiver and serial transmitter

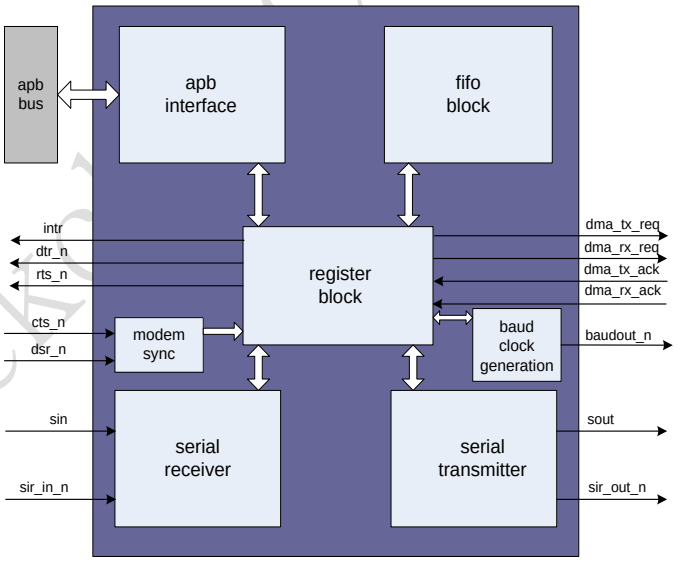


Fig 23-1 UART Architecture

APB INTERFACE

The host processor accesses data, control, and status information on the UART through the APB interface. The UART 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
Generate 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 serial UART format.
Serial Receiver
Converts the serial data character (as specified by the control register) received in the UART format to parallel form. Parity error detection, framing error detection and line break detection is carried out in this block.

23.3 Function description

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.

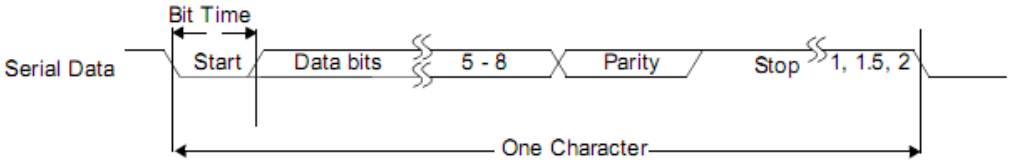


Fig 23-2 UART Serial protocol

Baud Clock
The baud rate is 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.

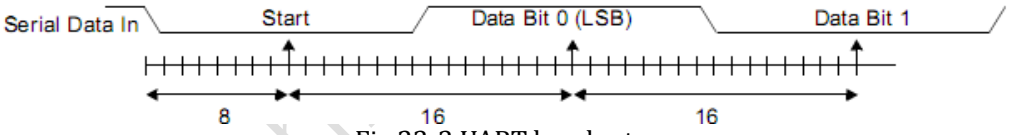


Fig 23-3 UART baud rate

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 of UART0/UART1/UART2/UART3/UART4/UART5 is 32bytes.The FIFO mode of all the UART is 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 Interrupt mode)
- Modem Status

DMA Support
The UART supports DMA signaling 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:
- When the Transmitter Holding Register is empty in non-FIFO mode.
- When the transmitter FIFO is empty in FIFO mode with Programmable THRE interrupt mode disabled.

- 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:

- When there is a single character available in the Receive Buffer Register in non-FIFO mode.
- 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.

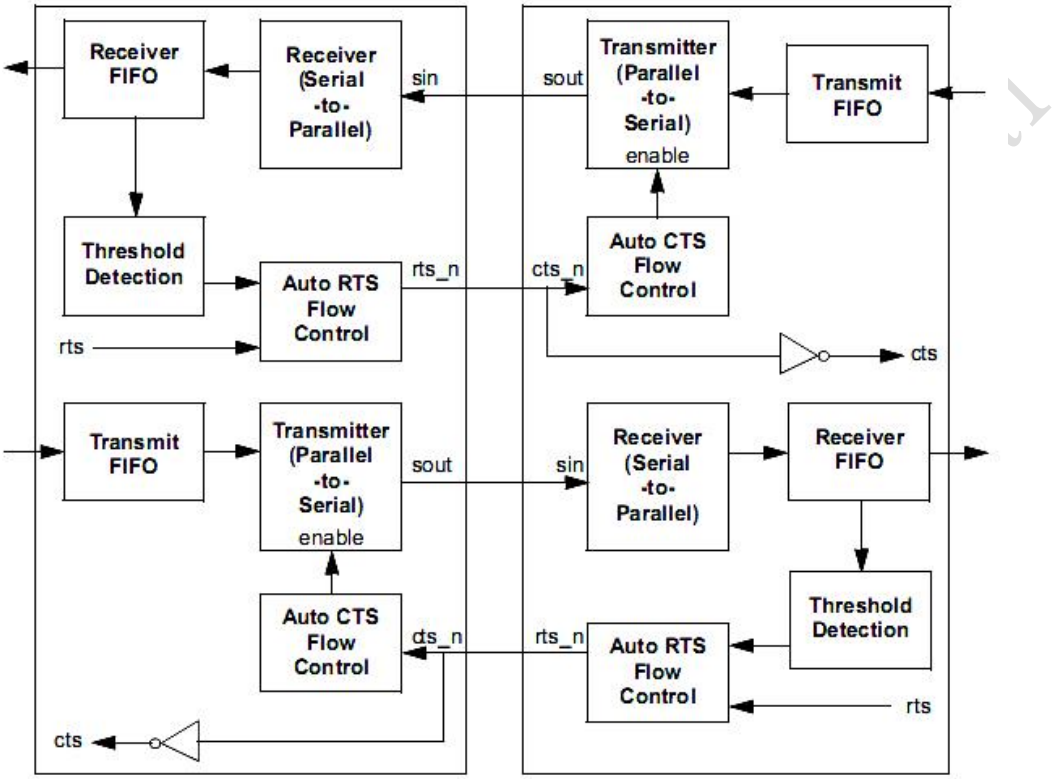


Fig 23-4 UART Auto flow control block diagram

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)

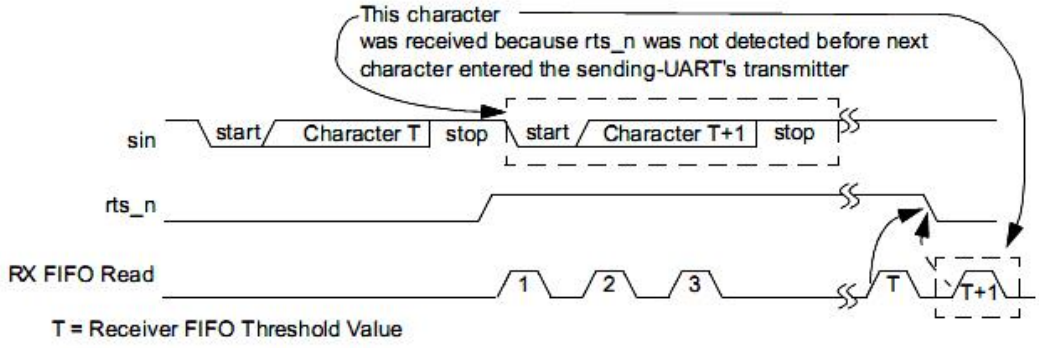


Fig 23-5 UART AUTO RTS TIMING

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)

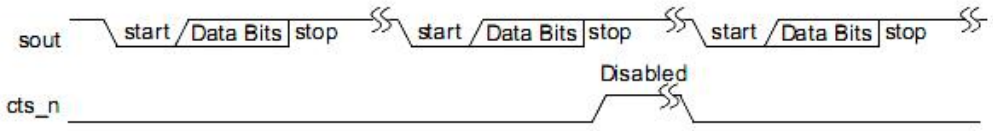


Fig 23-6 UART AUTO CTS TIMING

23.4 Register Description

This section describes the control/status registers of the design. There are 6 UARTs in the chip, and each one has its own base address.

23.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
UART_RBR	0x0000	W	0x00000000	Receive Buffer Register
UART_THR	0x0000	W	0x00000000	Transmit Holding Register
UART_DLL	0x0000	W	0x00000000	Divisor Latch (Low)
UART_DLH	0x0004	W	0x00000000	Divisor Latch (High)
UART_IER	0x0004	W	0x00000000	Interrupt Enable Register
UART_IIR	0x0008	W	0x00000000	Interrupt Identification Register
UART_FCR	0x0008	W	0x00000000	FIFO Control Register
UART_LCR	0x000c	W	0x00000000	Line Control Register
UART_MCR	0x0010	W	0x00000000	Modem Control Register
UART_LSR	0x0014	W	0x00000000	Line Status Register
UART_MSR	0x0018	W	0x00000000	Modem Status Register
UART_SCR	0x001c	W	0x00000000	Scratchpad Register
UART_SRBR	0x0030	W	0x00000000	Shadow Receive Buffer Register
UART_STHR	0x006c	W	0x00000000	Shadow Transmit Holding Register
UART_FAR	0x0070	W	0x00000000	FIFO Access Register
UART_TFR	0x0074	W	0x00000000	Transmit FIFO Read
UART_RFW	0x0078	W	0x00000000	Receive FIFO Write
UART_USR	0x007c	W	0x00000000	UART Status Register
UART_TFL	0x0080	W	0x00000000	Transmit FIFO Level
UART_RFL	0x0084	W	0x00000000	Receive FIFO Level
UART_SRR	0x0088	W	0x00000000	Software Reset Register
UART_SRTS	0x008c	W	0x00000000	Shadow Request to Send
UART_SBCR	0x0090	W	0x00000000	Shadow Break Control Register
UART_SDMAM	0x0094	W	0x00000000	Shadow DMA Mode
UART_SFE	0x0098	W	0x00000000	Shadow FIFO Enable
UART_SRT	0x009c	W	0x00000000	Shadow RCVR Trigger
UART_STET	0x00a0	W	0x00000000	Shadow TX Empty Trigger
UART_HTX	0x00a4	W	0x00000000	Halt TX
UART_DMASA	0x00a8	W	0x00000000	DMA Software Acknowledge
UART_CPR	0x00f4	W	0x00000000	Component Parameter Register
UART_UCV	0x00f8	W	0x0330372a	UART Component Version
UART_CTR	0x00fc	W	0x44570110	Component Type Register

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

23.4.2 Detail Register Description

UART_RBR

Address: Operational Base + offset (0x0000)

Receive Buffer Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	data_input 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 (0x0000)

Transmit Holding Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:0	RW	0x00	data_output 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 (0x0000)
Divisor Latch (Low)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:0	RW	0x00	baud_rate_divisor_L 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 (0x0004)
Divisor Latch (High)

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	baud_rate_divisor_H 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 (0x0004)
Interrupt Enable Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	prog_thre_int_en Programmable THRE Interrupt Mode Enable This is used to enable/disable the generation of THRE Interrupt. 0 = disabled 1 = enabled
6:4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3	RW	0x0	modem_status_int_en 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	receive_line_status_int_en 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	trans_hold_empty_int_en Enable Transmit Holding Register Empty Interrupt.
0	RW	0x0	receive_data_available_int_en 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 (0x0008)
Interrupt Identification Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	RO	0x0	fifos_en FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00 = disabled 11 = enabled
5:4	RO	0x0	reserved

Bit	Attr	Reset Value	Description
3:0	RO	0x0	int_id 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 (0x0008)
FIFO Control Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:6	WO	0x0	rcvr_trigger 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 1/4 full 10 = FIFO 1/2 full 11 = FIFO 2 less than ful
5:4	WO	0x0	tx_empty_trigger 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 1/4 full 11 = FIFO 1/2 full

Bit	Attr	Reset Value	Description
3	WO	0x0	dma_mode DMA Mode This determines the DMA signaling 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	WO	0x0	xmit_fifo_reset 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	WO	0x0	rcvr_fifo_reset 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	WO	0x0	fifo_en FIFO Enable. 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 (0x000c)

Line Control Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7	RW	0x0	div_lat_access 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_ctrl 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	RO	0x0	reserved
4	RW	0x0	even_parity_sel 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_en 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

Bit	Attr	Reset Value	Description
2	RW	0x0	stop_bits_num 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_sel 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 (0x0010)
Modem Control Register

Bit	Attr	Reset Value	Description
31:7	RO	0x0	reserved
6	RW	0x0	sir_mode_en SIR Mode Enable. SIR Mode Enable. This is used to enable/disable the IrDA SIR Mode. 0 = IrDA SIR Mode disabled 1 = IrDA SIR Mode enabled

Bit	Attr	Reset Value	Description
5	RW	0x0	auto_flow_ctrl_en Auto Flow Control Enable. 0 = Auto Flow Control Mode disabled 1 = Auto Flow Control Mode enabled
4	RW	0x0	loopback LoopBack Bit. This is used to put the UART into a diagnostic mode for test purposes.
3	RW	0x0	out2 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 OUT1
1	RW	0x0	req_to_send 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 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 (0x0014)

Line Status Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7	RO	0x0	receiver_fifo_error 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	RO	0x0	trans_empty Transmitter Empty bit. 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	RO	0x0	trans_hold_reg_empty 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	RO	0x0	break_int Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.
3	RO	0x0	framing_error 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.

Bit	Attr	Reset Value	Description
2	RO	0x0	parity_eror 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	RO	0x0	overrun_error 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	RO	0x0	data_ready 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 (0x0018)
Modem Status Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RO	0x0	data_carrior_detect Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n.
6	RO	0x0	ring_indicator Ring Indicator. This is used to indicate the current state of the modem control line ri_n.
5	RO	0x0	data_set_ready Data Set Ready. This is used to indicate the current state of the modem control line dsr_n.
4	RO	0x0	clear_to_send Clear to Send. This is used to indicate the current state of the modem control line cts_n.
3	RO	0x0	delta_data_carrier_detect 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.

Bit	Attr	Reset Value	Description
2	RO	0x0	trailing_edge_ring_indicator Trailing Edge of Ring Indicator. 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	RO	0x0	delta_data_set_ready 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	RO	0x0	delta_clear_to_send 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 (0x001c)
Scratchpad Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RW	0x00	temp_store_space This register is for programmers to use as a temporary storage space.

UART_SRBR
Address: Operational Base + offset (0x0030)
Shadow Receive Buffer Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved

Bit	Attr	Reset Value	Description
7:0	RO	0x00	shadow_rbr 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 (0x006c)
Shadow Transmit Holding Register

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	shadow_thr This is a shadow register for the THR.

UART_FAR
Address: Operational Base + offset (0x0070)
FIFO Access Register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved

Bit	Attr	Reset Value	Description
0	RW	0x0	fifo_access_test_en 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 (0x0074)
Transmit FIFO Read

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	trans_fifo_read 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.

UART_RFW
Address: Operational Base + offset (0x0078)
Receive FIFO Write

Bit	Attr	Reset Value	Description
31:10	RO	0x0	reserved
9	WO	0x0	receive_fifo_framing_error Receive FIFO Framing Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).
8	WO	0x0	receive_fifo_parity_error Receive FIFO Parity Error. These bits are only valid when FIFO access mode is enabled (FAR[0] is set to one).

Bit	Attr	Reset Value	Description
7:0	WO	0x00	receive_fifo_write 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 (0x007c)
UART Status Register

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4	RO	0x0	receive_fifo_full 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	RO	0x0	receive_fifo_not_empty 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	RO	0x0	trasn_fifo_empty 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

Bit	Attr	Reset Value	Description
1	RO	0x0	trans_fifo_not_full 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	RO	0x0	uart_busy UART Busy. 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 (0x0080)
Transmit FIFO Level

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RW	0x00	trans_fifo_level Transmit FIFO Level. This is indicates the number of data entries in the transmit FIFO.

UART_RFL
Address: Operational Base + offset (0x0084)
Receive FIFO Level

Bit	Attr	Reset Value	Description
31:5	RO	0x0	reserved
4:0	RO	0x00	receive_fifo_level Receive FIFO Level. This is indicates the number of data entries in the receive FIFO.

UART_SRR
Address: Operational Base + offset (0x0088)
Software Reset Register

Bit	Attr	Reset Value	Description
31:3	RO	0x0	reserved
2	WO	0x0	xmit_fifo_reset XMIT FIFO Reset. This is a shadow register for the XMIT FIFO Reset bit (FCR[2]).

Bit	Attr	Reset Value	Description
1	WO	0x0	rcvr_fifo_reset RCVR FIFO Reset. This is a shadow register for the RCVR FIFO Reset bit (FCR[1]).
0	WO	0x0	uart_reset 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 (0x008c)
Shadow Request to Send

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	shadow_req_to_send 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 (0x0090)
Shadow Break Control Register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	shadow_break_ctrl 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 (0x0094)
Shadow DMA Mode

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	shadow_dma_mode Shadow DMA Mode. This is a shadow register for the DMA mode bit (FCR[3]).

UART_SFE
Address: Operational Base + offset (0x0098)
Shadow FIFO Enable

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	shadow_fifo_en Shadow FIFO Enable. Shadow FIFO Enable. This is a shadow register for the FIFO enable bit (FCR[0]).

UART_SRT
Address: Operational Base + offset (0x009c)
Shadow RCVR Trigger

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	shadow_rcvr_trigger Shadow RCVR Trigger. This is a shadow register for the RCVR trigger bits (FCR[7:6]).

UART_STET
Address: Operational Base + offset (0x00a0)
Shadow TX Empty Trigger

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	shadow_tx_empty_trigger Shadow TX Empty Trigger. This is a shadow register for the TX empty trigger bits (FCR[5:4]).

UART_HTX
Address: Operational Base + offset (0x00a4)
Halt TX

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	halt_tx_en 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 (0x00a8)
DMA Software Acknowledge

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	WO	0x0	dma_software_ack This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition.

UART_CPR
Address: Operational Base + offset (0x00f4)
Component Parameter Register

Bit	Attr	Reset Value	Description
31:24	RO	0x0	reserved
23:16	RO	0x00	FIFO_MODE 0x00 = 0 0x01 = 16 0x02 = 32 to 0x80 = 2048 0x81- 0xff = reserved
15:14	RO	0x0	reserved
13	RO	0x0	DMA_EXTRA 0 = FALSE 1 = TRUE
12	RO	0x0	UART_ADD_ENCODED_PARAMS 0 = FALSE 1 = TRUE
11	RO	0x0	SHADOW 0 = FALSE 1 = TRUE
10	RO	0x0	FIFO_STAT 0 = FALSE 1 = TRUE
9	RO	0x0	FIFO_ACCESS 0 = FALSE 1 = TRUE
8	RO	0x0	NEW_FEAT 0 = FALSE 1 = TRUE
7	RO	0x0	SIR_LP_MODE 0 = FALSE 1 = TRUE
6	RO	0x0	SIR_MODE 0 = FALSE 1 = TRUE

Bit	Attr	Reset Value	Description
5	RO	0x0	THRE_MODE 0 = FALSE 1 = TRUE
4	RO	0x0	AFCE_MODE 0 = FALSE 1 = TRUE
3:2	RO	0x0	reserved
1:0	RO	0x0	APB_DATA_WIDTH 00 = 8 bits 01 = 16 bits 10 = 32 bits 11 = reserved

UART_UCV
Address: Operational Base + offset (0x00f8)
UART Component Version

Bit	Attr	Reset Value	Description
31:0	RO	0x0330372a	ver ASCII value for each number in the version

UART_CTR
Address: Operational Base + offset (0x00fc)
Component Type Register

Bit	Attr	Reset Value	Description
31:0	RO	0x44570110	peripheral_id This register contains the peripherals identification code.

23.5 Interface description

UART0 have 2 IOMUX, which is controlled by GRF_UOC_CON0[2:0].
When GRF_UOC_CON0[2:0] is 3’b000. The IOMUX is as follow.

Table 23-1UART0 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART0 Interface			
uart0_sin	I	IO_UART0Arx_JTG1tms_I2C1Csda_GPIOP2b4	GRF_GPIO2B_IOMUX[9:8]=2’b01
uart0_sout	O	IO_UART0Atx_JTG1tck_I2C1Cscl_GPIOP2b5	GRF_GPIO2B_IOMUX[11:10]=2’b01

When GRF_UOC_CON0[2:0] is 3’b111. The IOMUX is as follow.

Table 23-2UART0 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART0 Interface			
uart0_sin	I	IO_USB0PP	GRF_UOC_CON0[2:0]=3’b111
uart0_sout	O	IO_USB0PN	GRF_UOC_CON0[2:0]=3’b111

UART1 have 2 IOMUX, which is controlled by GRF_IOMUX_CON[9].
When GRF_IOMUX_CON[9] is 1'b0. The IOMUX is as follow.

Table 23-3UART1 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART1 Interface			
Uart1_sin	I	IO_UART1Arx_I2C0Bscl_SPI1Btx_GPIOP2c1	GRF_GPIO2C_IOMUX[3:2]=2'b01
Uart1_sout	O	IO_UART1Atx_I2C0Bsda_SPI1Bclk_GPIOP2c0	GRF_GPIO2C_IOMUX[1:0]=2'b01
Uart1_cts_n	I	IO_UART1Actx_JTG0tck_SPI1Brx_GPIOP2b7	GRF_GPIO2B_IOMUX[15:14]=2'b01
Uart1_rts_n	O	IO_UART1Arts_JTG0tms_SPI1Bcs_GPIOP2b6	GRF_GPIO2B_IOMUX[13:12]=2'b01

When GRF_IOMUX_CON[9] is 1'b1. The IOMUX is as follow.

Table 23-4UART1 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART1 Interface			
Uart1_sin	I	IO_I2S1Alrck_UART1Brx_EBCsdshr_GPIO1a1	GRF_GPIO1A_IOMUX[3:2]=2'b10
Uart1_sout	O	IO_I2S1Asclk_UART1Btx_EBCsdce2_GPIO1a2	GRF_GPIO1A_IOMUX[5:4]=2'b10
Uart1_cts_n	I	IO_I2S1Asdo_UART1Bcts_EBCsdce1_GPIO1a3	GRF_GPIO1A_IOMUX[7:6]=2'b10
Uart1_rts_n	O	IO_I2S1Asdi_UART1Brts_EBCsdce4_GPIO1a4	GRF_GPIO1A_IOMUX[9:8]=2'b10

UART2 have 3 IOMUX, which is controlled by GRF_IOMUX_CON[11:10].
When GRF_IOMUX_CON[11:10] is 2'b00. The IOMUX is as follow.

Table 23-5UART2 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART2 Interface			
Uart2_sin	I	IO_UART2Arx_I2S0sclk_EBCgdclk_GPIO0b5	GRF_GPIO0B_IOMUX[11:10]=2'b01
Uart2_sout	O	IO_UART2Atx_I2S0lrck_EBCgdsp_GPIO0b6	GRF_GPIO0B_IOMUX[13:12]=2'b01
Uart2_cts_n	I	IO_UART2Acts_I2S0sdo_EBCsdclk_GPIO0b4	GRF_GPIO0B_IOMUX[9:8]=2'b01
Uart2_rts_n	O	IO_UART2Arts_I2S0sdi_EBCvcom_GPIO0b3	GRF_GPIO0B_IOMUX[7:6]=2'b01

When GRF_IOMUX_CON[11:10] is 2'b01. The IOMUX is as follow.

Table 23-6UART2 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART2 Interface			
Uart2_sin	I	IO_LCDd4_UART2Brx_EBCsddo4_GPIO0c4	GRF_GPIO0C_IOMUX[9:8]=2'b10

Uart2_sout	O	IO_LCDd5_UART2Btx_EBCsddo5_GPIO0c5	GRF_GPIO0C_IOMUX[11:10]=2'b10
Uart2_cts_n	I	IO_LCDd7_UART2Bcts_EBCsddo7_GPIO0c7	GRF_GPIO0C_IOMUX[15:14]=2'b10
Uart2_rts_n	O	IO_LCDd6_UART2Brts_EBCsddo6_GPIO0c6	GRF_GPIO0C_IOMUX[13:12]=2'b10

When GRF_IOMUX_CON[11:10] is 2'b10. The IOMUX is as follow.

Table 23-7UART2 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART2 Interface			
Uart2_sin	I	IO_EMMCcmd_I2S1Bsclk_UART2Crx_GPIO0a2	GRF_GPIO0A_IOMUX[5:4]=2'b11
Uart2_sout	O	IO_EMMCclk_I2S1Blrck_UART2Ctx_GPIO0a1	GRF_GPIO0A_IOMUX[3:2]=2'b11
Uart2_cts_n	I	IO_EMMCd0_I2S1Bsdo_UART2Ccts_GPIO0a3	GRF_GPIO0A_IOMUX[7:6]=2'b11
Uart2_rts_n	O	IO_EMMCd1_I2S1Bsdi_UART2Crts_GPIO0a4	GRF_GPIO0A_IOMUX[9:8]=2'b11

Table 23-8UART3 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART3 Interface			
Uart3_sin	I	IO_SDMMCclk_SPI1Aclk_UART3rx_GPIO1a6	GRF_GPIO1A_IOMUX[13:12]=2'b11
Uart3_sout	O	IO_SDMMCcmd_SPI1Acs_UART3tx_GPIO1a5	GRF_GPIO1A_IOMUX[11:10]=2'b11

Table 23-9UART4 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART4 Interface			
Uart4_sin	I	IO_SDMMCd1_SPI1Atx_UART4rx_GPIO1b0	GRF_GPIO1B_IOMUX[1:0]=2'b11
Uart4_sout	O	IO_SDMMCd0_SPI1Arx_UART4tx_GPIO1a7	GRF_GPIO1A_IOMUX[15:14]=2'b11

Table 23-10UART5 Interface Description

Module pin	Dir.	Pad name	IOMUX
UART5 Interface			
Uart5_sin	I	IO_SDMMCd3_I2C1Bsda_UART5rx_GPIO1b2	GRF_GPIO1B_IOMUX[5:4]=2'b11
Uart5_sout	O	IO_SDMMCd2_I2C1Bscl_UART5tx_GPIO1b1	GRF_GPIO1B_IOMUX[3:2]=2'b11

23.6 Application Notes

23.6.1 None FIFO Mode Transfer Flow

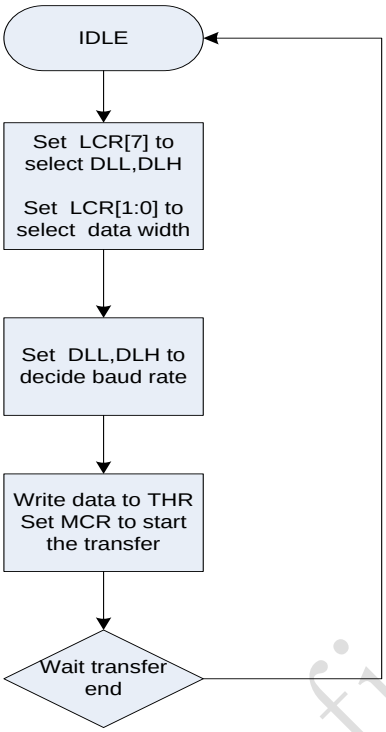


Fig 23-7 UART none fifo mode

23.6.2 FIFO Mode Transfer Flow

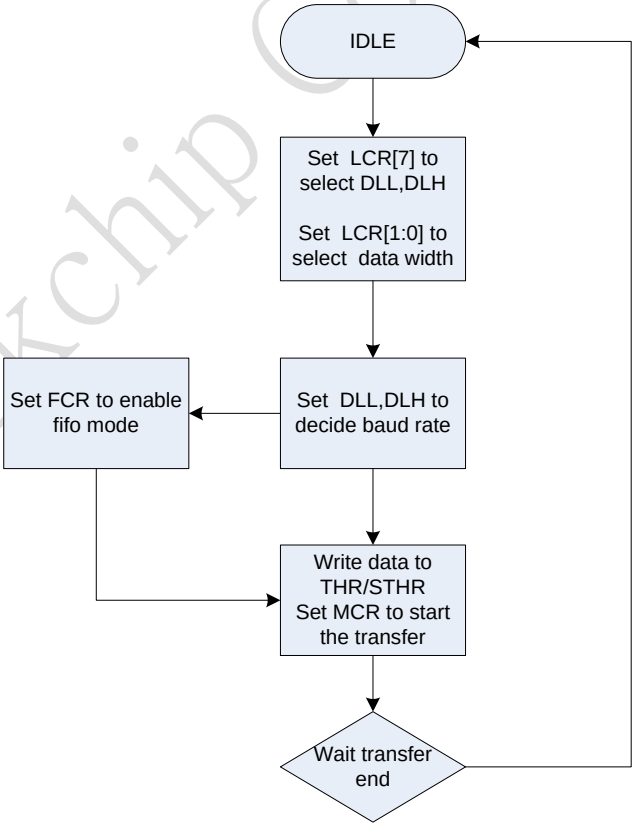


Fig 23-8 UART fifo mode

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 32-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.

23.6.3 Baud Rate Calculation

UART clock generation

The following figures shows the UART clock generation.

UART1 source clocks can be selected from xin24m or PLL_MUX_CLK or usbphy480m. UART0/ UART2 / UART3/UART4/UART5 source clocks can be selected from xin24m or PLL_MUX_CLK. UART clocks can be generated by 1 to 64 division of its source clock.

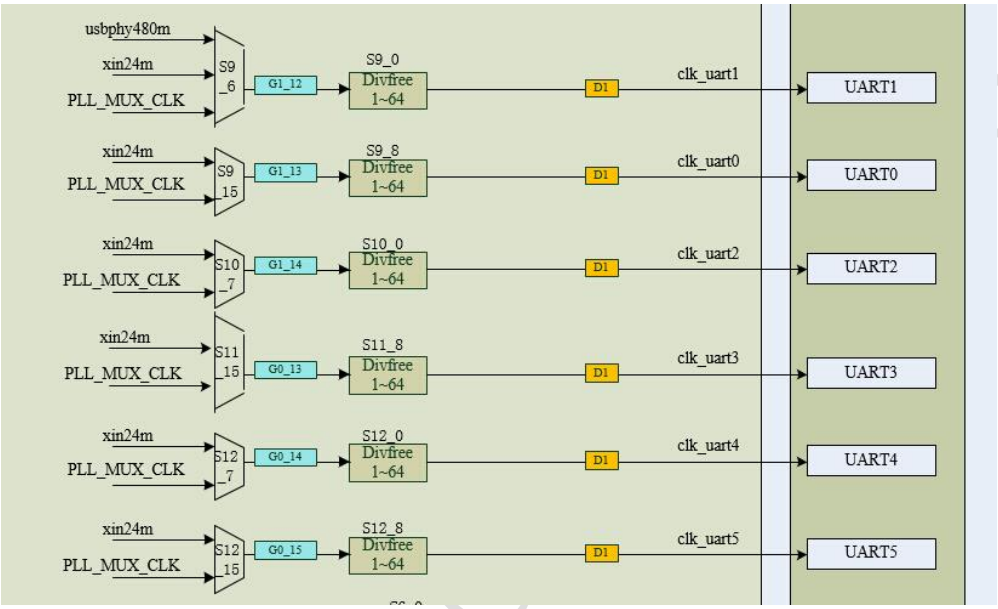


Fig 23-9 UART clock generation

Chapter 24 EBC

24.1 Overview

EBC is the TCON module for Electronic Paper Display (EPD).
EBC supports the following features:

System interface

- AHB slave for register configuration
- AHB master for frame data transfer (DMA)
- Interrupt output

EPD interface

- E-ink EPD compatible
- Up to 2048x2048 resolution
- Up to 16 level gray scale
- Up to 128 frames every scanning
- LUT updateable (8KB)
- Direct mode and LUT mode
- All-update mode and Diff-update mode
- Single-phase and multi-phase mode
- Support window display
- Source driver interface
- Gate driver interface

24.2 Block Diagram

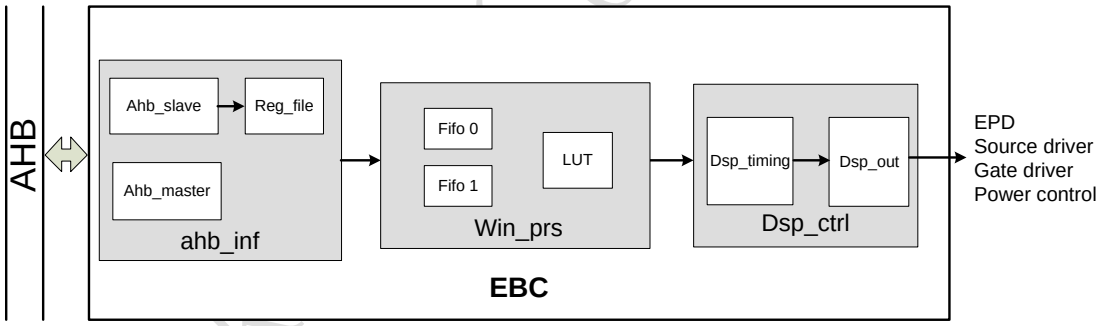


Fig 24-1 EBC Block Diagram

24.3 Function description

24.3.1 Data Format

Table 24-1EBC Input Data Format

DSP_LUT MODE	WIN_FMT [1:0]	Data format	Bit Map
0	xx	S-data(2bpp)	{S3[7:0], S2[7:0], S1[7:0], S0[7:0]}
1	00	Y-data(4bpp)	{Y7[3:0], Y6[3:0], Y5[3:0], Y4[3:0], Y3[3:0], Y2[3:0], Y1[3:0], Y0[3:0]}
1	01	Y-data(8bpp)	{Y3[7:0], Y2[7:0], Y1[7:0], Y0[7:0]}
1	10	RGB888	{8'bx, R[7:0], G[7:0], B[7:0]}
1	11	RGB565	{ R1[4:0], G1[5:0], B1[4:0], R0[4:0], G0[5:0], B0[4:0]}

24.3.2 Waveform generation mode

1. Direct mode

In direct scanning mode, the source data is just read by internal DMA and sent to the display output directly.

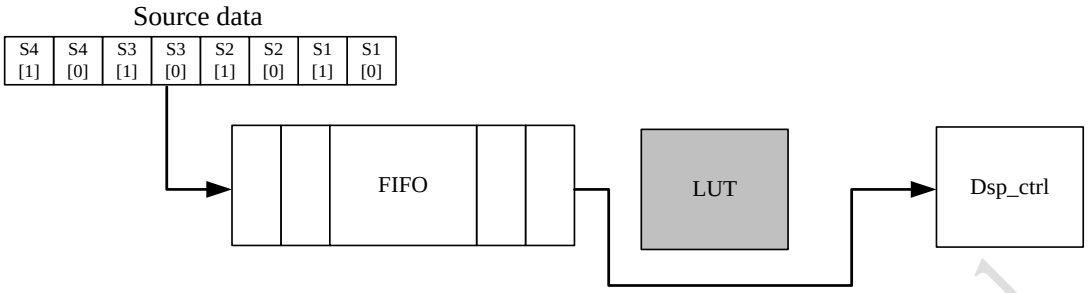


Fig 24-2 EBC Block Diagram

2. LUT mode

In LUT scanning mode, internal DMA read the original pixel data into the FIFO, then the pixel data is sent to the look-up table to be translated the EPD source data.

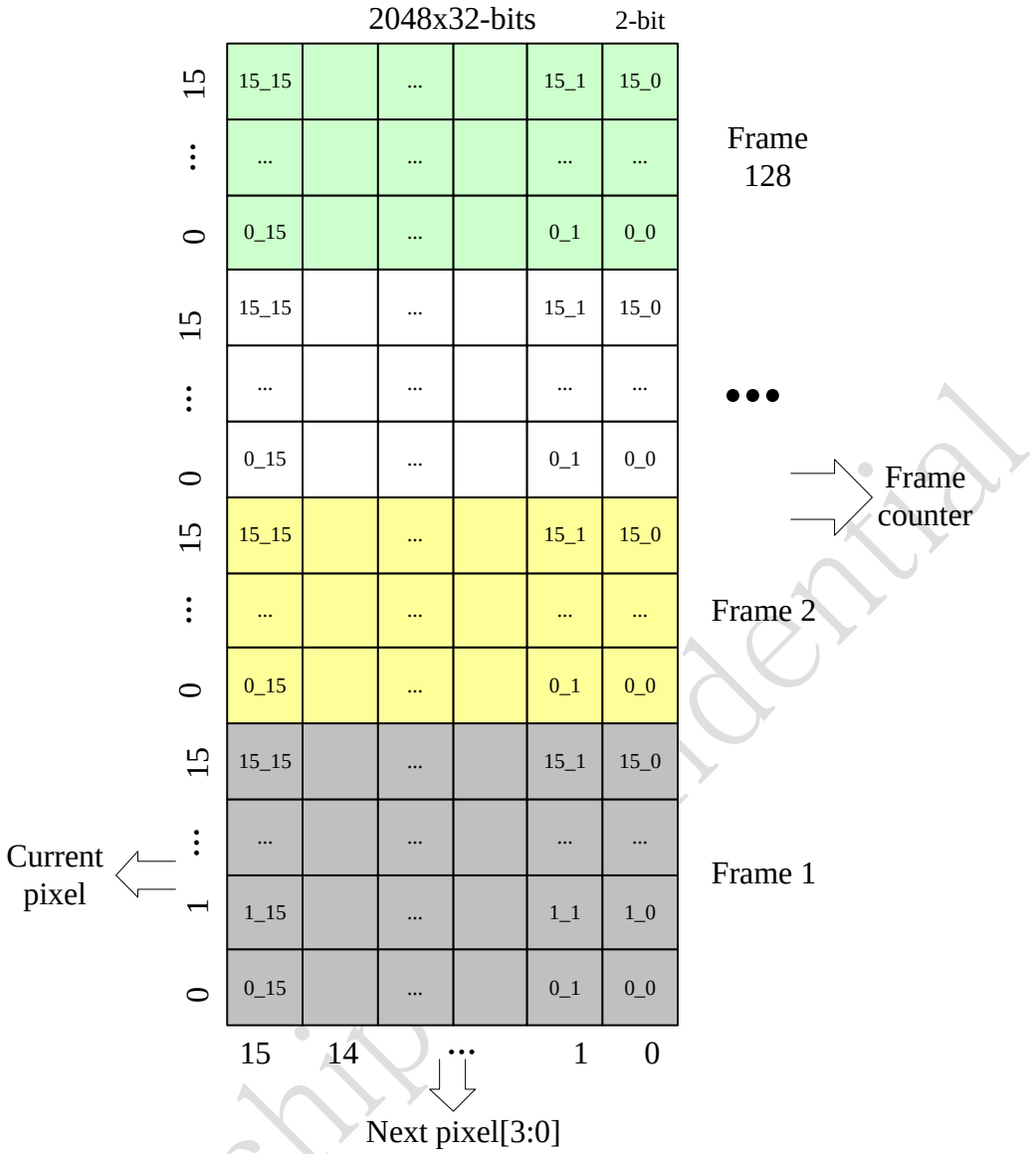


Fig 24-3 EBC LUT structure

24.3.3 Window display mode

Window display is supported in EBC, dsp_win_width/dsp_win_height and dsp_win_st should be set to define the display window. The source value of other area is background value, which is configurable.

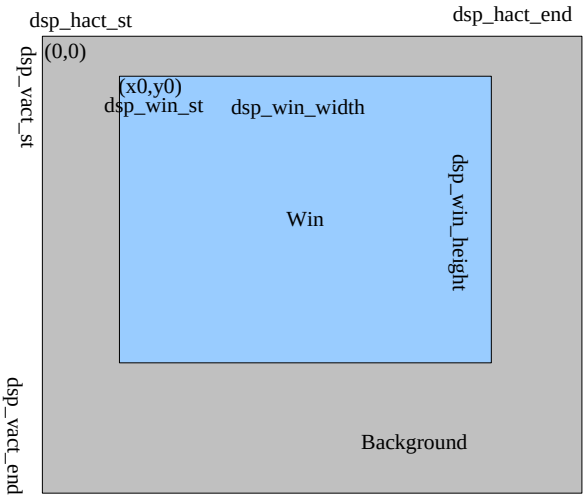


Fig 24-4 EBC window display

24.3.4 Frame control

Single frame mode(always used in direct mode)

In single frame mode (direct mode), every display frame is controlled by the "frame_start" command. The next "frame_start" command should be after the end of the last frame.

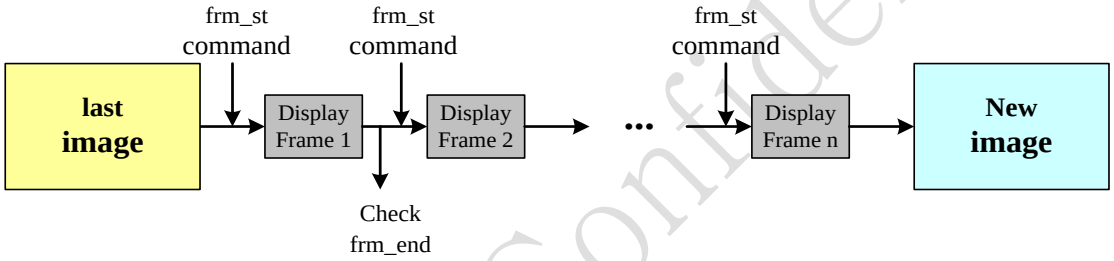


Fig 24-5 EBC single frame display

Multi-frame mode

In multi-frame mode, the number of the display frames should be set before the "frame_start" command. Then the frame scanning is done automatically until the end of the last frame.

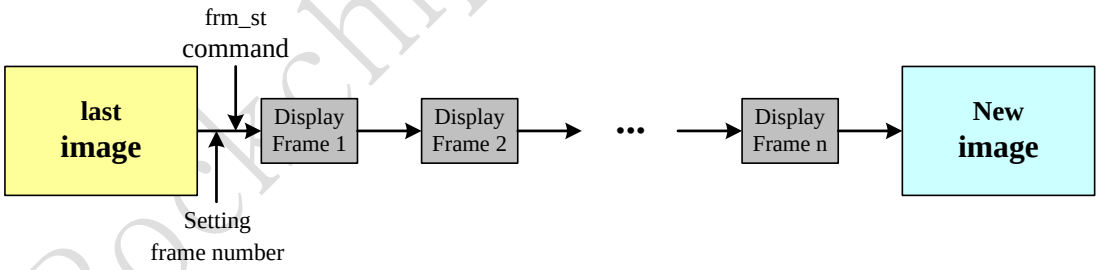


Fig 24-6 EBC multi-frame display

24.3.5 Image translation phase

Single phase mode

If the number of image translation frames is less than 64. Single phase is necessary for image translation

In this mode, the "vcom_en" is just turn on during the display period.

Multi-phase mode

If the number of image translation frames is larger than 64, multi-phase should be done for image translation because the max display frame is 64.

In this mode, the "vcom_en" should not be turn off between the idle period of different display phase. To avoid this, you can set VCOM_MODE = "1" before the first start display start command. To turn off the VCOM_EN, "0" should be write in before the last start display start command.

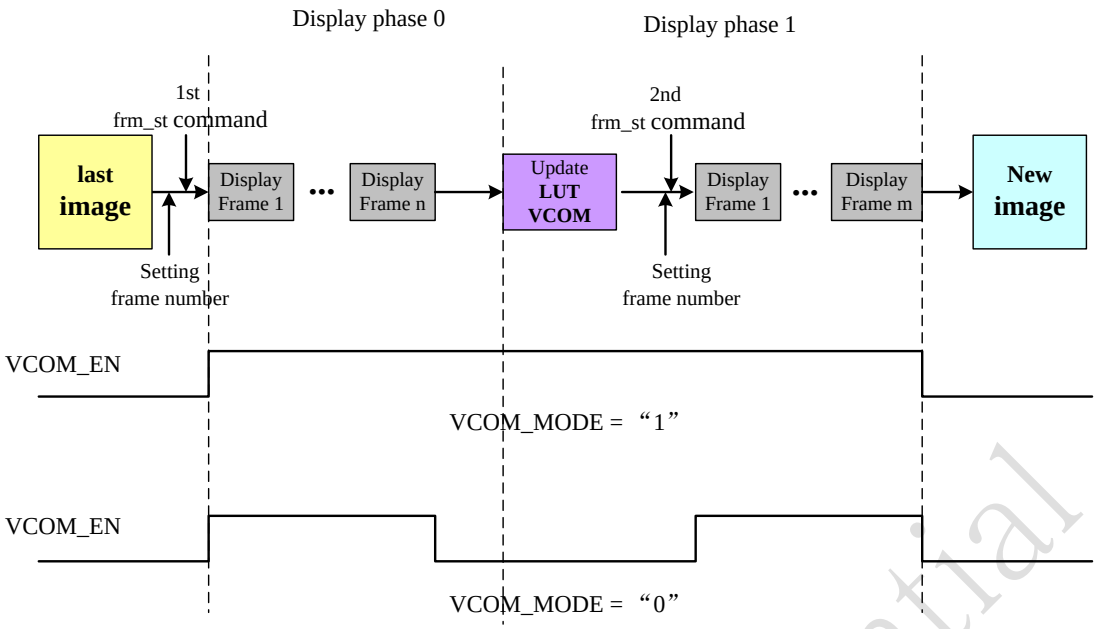


Fig 24-7 EBC display phase

24.3.6 IO description

IO MUX

IO MUX can be switched to EBC outputs when configured as following table:

Table 24-2 IO MUX configuration

Function	IO	IO MUX Configuration
sdclk	IO_UART2Acts_I2S0sdo_EBCsdclk_GPIO0b4	gpio0b4_sel_0 = 1, gpio0b4_sel_1 = 1
sdle	IO_LCDwrn_I2C2Bscl_EBCsdle_GPIO0d0	gpio0d0_sel_0 = 1, gpio0d0_sel_1 = 1
sdoe	IO_LCDrs_I2C2Bsda_EBCsdoe_GPIO0d1	gpio0d1_sel_0 = 1, gpio0d1_sel_1 = 1
sdce[0]	IO_PWM3out_I2C2Ascl_EBCsdce0_GPIOP2a1	gpiop2a1_sel_0 = 1, gpiop2a1_sel_1 = 1
sdce[1]	IO_I2S1Asdo_UART1Bcts_EBCsdce1_GPIO1a3	gpio1a3_sel_0 = 1, gpio1a3_sel_1 = 1
sdce[2]	IO_I2S1Asclk_UART1Btx_EBCsdce2_GPIO1a2	gpio1a2_sel_0 = 1, gpio1a2_sel_1 = 1
sdce[3]	IO_I2C1Asda_SPI0Bcs_EBCsdce3_GPIOP2b0	gpiop2b0_sel_0 = 1, gpiop2b0_sel_1 = 1
sdce[4]	IO_I2S1Asdi_UART1Brts_EBCsdce4_GPIO1a4	gpio1a4_sel_0 = 1, gpio1a4_sel_1 = 1
sdce[5]	IO_I2C0Ascl_SPI0Btx_EBCsdce5_GPIOP2b2	gpiop2b2_sel_0 = 1, gpiop2b2_sel_1 = 1
sddo[x] x = 0 to 7	IO_LCDd0_SPI0Atx_EBCsddo0_GPIO0c0 IO_LCDd1_SPI0Arx_EBCsddo1_GPIO0c1 IO_LCDd2_SPI0Aclk_EBCsddo2_GPIO0c2 IO_LCDd3_SPI0Acs_EBCsddo3_GPIO0c3 IO_LCDd4_UART2Brx_EBCsddo4_GPIO0c4 IO_LCDd5_UART2Btx_EBCsddo5_GPIO0c5 IO_LCDd6_UART2Brts_EBCsddo6_GPIO0c6 IO_LCDd7_UART2Bcts_EBCsddo7_GPIO0c	gpio0cx_sel_0 = 1, gpio0cx_sel_1 = 1 (x = 0 to 7)
sdshr	IO_I2S1Alrck_UART1Brx_EBCsdshr_GPIO1a1	gpio1a1_sel_0 = 1, gpio1a1_sel_1 = 1
gdclk	IO_UART2Arx_I2S0sclk_EBCgdclk_GPIO0b5	gpio0b5_sel_0 = 1, gpio0b5_sel_1 = 1
gdsp	IO_UART2Atx_I2S0lrck_EBCgdsp_GPIO0b6	gpio0b6_sel_0 = 1,

		gpio0b6_sel_1 = 1
gdoe	IO_LCDcsn_I2S0clk_EBCgdoe_GPIO0b7	gpio0b7_sel_0 = 1, gpio0b7_sel_1 = 1
gdrl	IO_I2S1Aclk_EBCgdrl_GPIO1a0	gpio1a0_sel_0 = 0, gpio1a0_sel_1 = 1
border[0]	IO_I2C0Asda_SPI0Brx_EBCborder0_GPIOP2b3	gpiop2b3_sel_0 = 1, gpiop2b3_sel_1 = 1
border[1]	IO_I2C1Ascl_SPI0Bclk_EBCborder1_GPIOP2b1	gpiop2b1_sel_0 = 1, gpiop2b1_sel_1 = 1
vcom	IO_UART2Arts_I2S0sdi_EBCvcom_GPIO0b3	gpio0b3_sel_0 = 1, gpio0b3_sel_1 = 1
gdpwr[0]	IO_PWM4out_I2C2Asda_EBCgdpwr0_GPIOP2a0	gpiop2a0_sel_0 = 1, gpiop2a0_sel_1 = 1
gdpwr[1]	IO_PWM1out_CLKobs_EBCgdpwr1_GPIOP2a3	gpiop2a3_sel_0 = 1, gpiop2a3_sel_1 = 1
gdpwr[2]	IO_PWM2out_EBCgdpwr2_PMUst3_GPIOP2a2	gpiop2a2_sel_0 = 0, gpiop2a2_sel_1 = 1

Note: For each IO MUX selection, the corresponding write_enable register also should be asserted. Register description and address can be found in Chapter 4 GRF.

Source driver interface

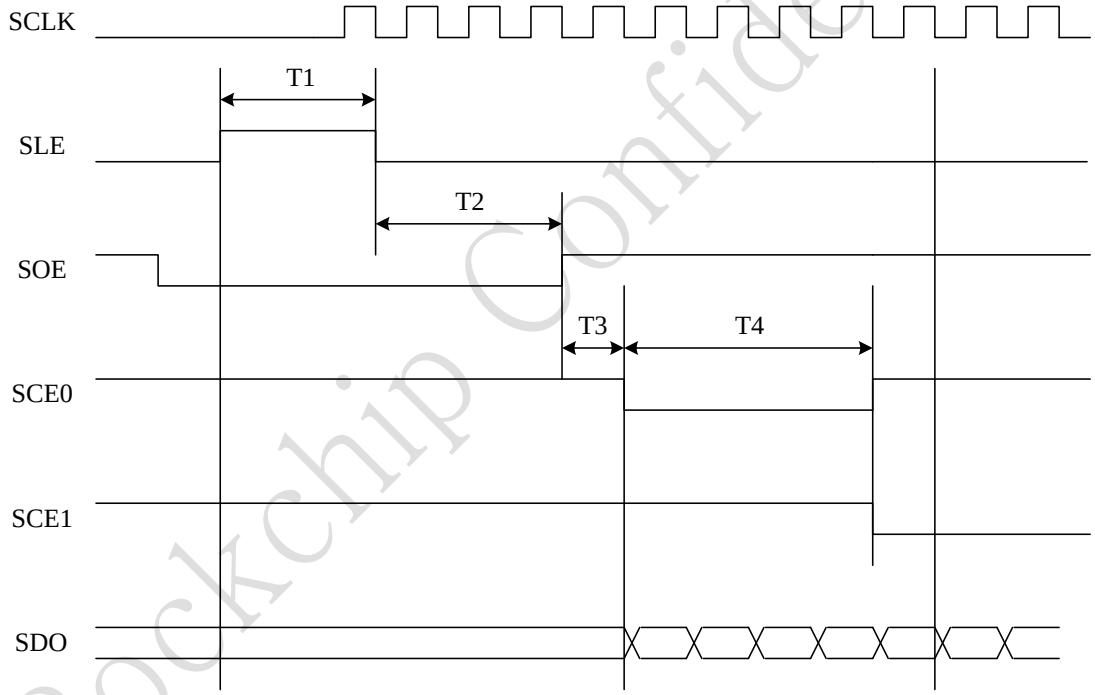


Fig 24-8 EBC source driver timing 1

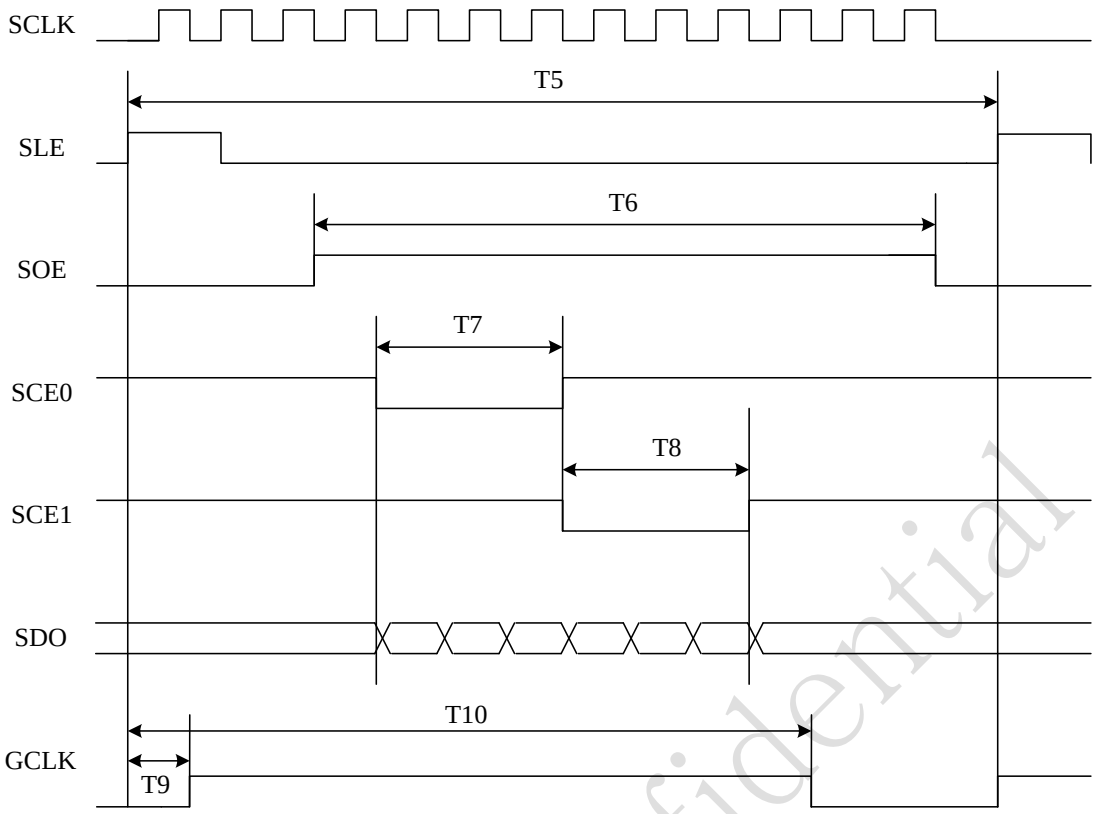


Fig 24-9 EBC source driver timing 2

Table 24-3EBC source driver setting

Timing	Value （6-inch 800x600）	Register setting
DCLK	33.25MHz	
SCLK	4 DCLK(8.3125 MHz)	(DSP_CLK_DIV+1)
T1	10 SCLK	DSP_HS_END
T2	3 SCLK	(DSP_HACT_ST – DSP_HS_END)
T3	1 SCLK	Fixed
T1+T2+T3	14 SCLK	DSP_WIN_XST
T4	100 SCLK	DSP_SCE_WIDTH
T5	315 SCLK	DSP_HTOTAL
T6	300 SCLK	(DSP_HACT_END – DSP_HACT_ST)
T7	100 SCLK	DSP_SCE_WIDTH
T8	100 SCLK	DSP_SCE_WIDTH
T9	0 SCLK	DSP_GCLK_ST
T10	215 SCLK	DSP_GCLK_END

Gate driver interface

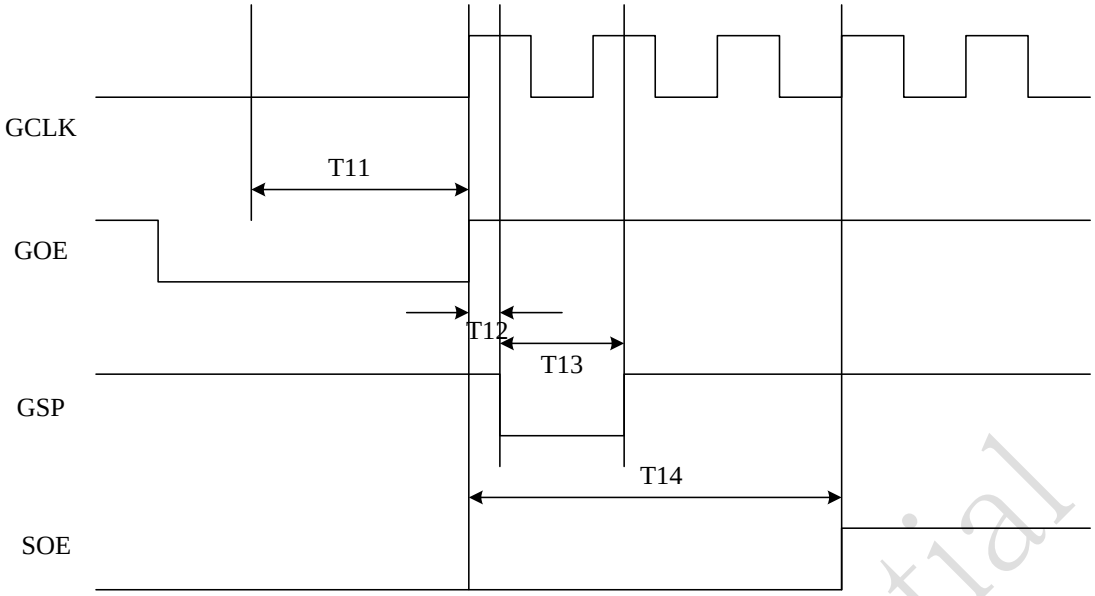


Fig 24-10 EBC gate driver timing 1

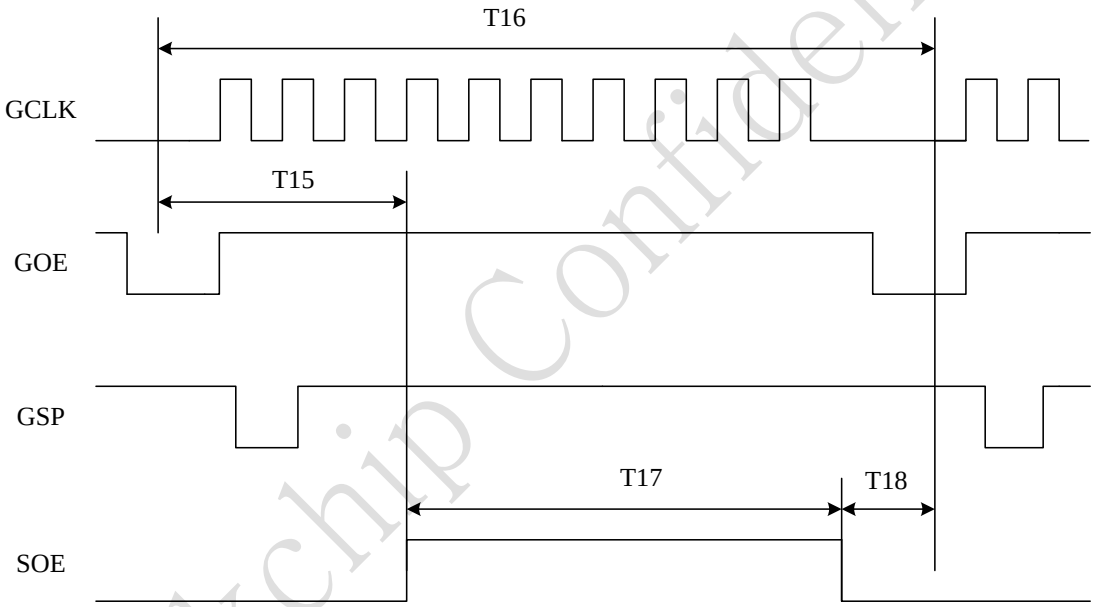


Fig 24-11 EBC gate driver timing 2

Table 24-4EBC gate driver setting

Timing	Value （6-inch 800x600）	Register setting
GCLK	315 SCLK	DSP_HTOTAL
T11	4 GCLK	DSP_VS_END
T12	107 SCLK	(DSP_GD_END/2)
T13	1 GCLK	Fixed
T14	4 GCLK	（DSP_VACT_ST - DSP_VS_END）
T15	8 GCLK	DSP_VACT_ST
T16	619 GCLK	DSP_VTOTAL
T17	601 GCLK	（DSP_VACT_END - DSP_VACT_ST）
T18	11 GCLK	（DSP_VTOTAL - DSP_VACT_END）

24.4 Register Description

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

24.4.1 Registers Summary

Name	Offset	Size	Reset Value	Description
EBC_DSP_ST	0x0000	W	0x00000000	frame start register
EBC_EPD_CTRL	0x0004	W	0x00006400	EPD control register
EBC_DSP_CTRL	0x0008	W	0x00030000	Display control register
EBC_DSP_HTIMING0	0x000c	W	0x013b000a	Display horizontal timing setting0
EBC_DSP_HTIMING1	0x0010	W	0x0139000d	Display horizontal timing setting1
EBC_DSP_VTIMING0	0x0014	W	0x026b0004	Display vertical timing setting0
EBC_DSP_VTIMING1	0x0018	W	0x02610008	Display vertical timing setting0
EBC_DSP_ACT_INFO	0x001c	W	0x025800c8	Display active width/height
EBC_WIN_CTRL	0x0020	W	0x00000000	Window control register
EBC_WIN_MST0	0x0024	W	0x00000000	Old window layer memory start address
EBC_WIN_MST1	0x0028	W	0x00000000	New window layer memory start address
EBC_WIN_VIR	0x002c	W	0x00000320	Window layer virtual width
EBC_WIN_ACT	0x0030	W	0x02580320	Window active width/height
EBC_WIN_DSP	0x0034	W	0x02580320	Window display width/height
EBC_WIN_DSP_ST	0x0038	W	0x00000000	Window display start position
EBC_INT_CTRL	0x003c	W	0x00000038	Interrupt control register
EBC_VCOM0	0x0040	W	0x00000000	VCOM0
EBC_VCOM1	0x0044	W	0x00000000	VCOM1
EBC_VCOM2	0x0048	W	0x00000000	VCOM2
EBC_VCOM3	0x004c	W	0x00000000	VCOM3
EBC_CONFIG_DONE	0x0050	W	0x00000000	CONFIG finish register
EBC_LUT_ADDR_MAP	0x1000	W	0x00000000	LUT address map

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

24.4.2 Detailed Register Description

Operational Base Address of EBC is 0x60040000.

EBC_DSP_ST

Address: Operational Base + offset (0x0000)

frame start register

Bit	Attr	Reset Value	Description
31:9	RO	0x0	reserved
8:2	RW	0x00	frm_total_num Frame total number(n) Frame_num = n+1. Up to 128
1	RO	0x0	reserved
0	RW	0x0	frm_st Frame start bit Writing'1' to trigger one frame display. Read this bit for Hold status.

EBC_EPD_CTRL
Address: Operational Base + offset (0x0004)
EPD control register

Bit	Attr	Reset Value	Description
31:27	RW	0x00	dsp_gd_st DSP_GD_ST GCLK rising edge point(SCLK), which count from the falling edge of hsync
26:16	RW	0x0d7	dsp_gd_end DSP_GD_END GCLK falling edge point(SCLK), which count from the falling edge of hsync
15:8	RW	0x64	dsp_sce_width DSP_SCE_WIDTH Source driver length
7:5	RO	0x0	reserved
4:2	RW	0x0	pwr_en Power enable[3:0]
1	RW	0x0	gate_scan_dir Gate scanning direction 1: button to top 0: top to button
0	RW	0x0	source_scan_dir Source scanning direction 1: right to left 0: left to right

EBC_DSP_CTRL
Address: Operational Base + offset (0x0008)
Display control egister

Bit	Attr	Reset Value	Description
31:30	RW	0x0	dsp_swap Display swap 11: SDO[7:0] = P0,P1,P2,P3 10: SDO[7:0] = P2,P3,P0,P1 01: SDO[7:0] = P1,P0,P3,P2 00: SDO[7:0] = P3,P2,P1,P0
29	RW	0x0	diff_update_mode_en Diff update Mode enable 1: Diff update Mode 0: Normal update Mode
28	RW	0x0	dsp_mode Display Mode 1: LUT mode 0: Direct mode

Bit	Attr	Reset Value	Description
27	RW	0x0	vcom_mode VCOM mode 1: Multi phase scanning mode 0: one phase scanning mode
26	RW	0x0	goe_pol_inv DSP OUTPUT GOE polarity invert 1: invert; 0: default;
25	RW	0x0	gsp_pol_inv DSP OUTPUT GSP polarity invert 1: invert; 0: default;
24	RW	0x0	gclk_pol_inv DSP OUTPUT GCLK polarity invert 1: invert; 0: default;
23	RW	0x0	sce_pol_inv DSP OUTPUT SCE polarity invert 1: invert; 0: default;
22	RW	0x0	soe_pol_inv DSP OUTPUT SOE polarity invert 1: invert; 0: default;
21	RW	0x0	sle_pol_inv DSP OUTPUT SLE polarity invert 1: invert; 0: default;
20	RW	0x0	sclk_pol_inv DSP OUTPUT SCLK polarity invert 1: invert; 0: default;
19:16	RW	0x3	sclk_div SCLK divide rate SCLK =（DSP_CLK_DIV+1）* DCLK
15:0	RW	0x0000	sdo_default_val SDO default value

EBC_DSP_HTIMING0

Address: Operational Base + offset (0x000c)

Display horizontal timing setting0

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x13b	dsp_h_period Panel display scanning horizontal period.

Bit	Attr	Reset Value	Description
15:8	RO	0x0	reserved
7:0	RW	0x0a	dsp_hsync_width Panel display scanning hsync(SLE) pulse width.

EBC_DSP_HTIMING1
Address: Operational Base + offset (0x0010)
Display horizontal timing setting1

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x139	dsp_hact_end Panel display scanning horizontal active end point
15:8	RO	0x0	reserved
7:0	RW	0x0d	dsp_hact_st Panel display scanning horizontal active start point

EBC_DSP_VTIMING0
Address: Operational Base + offset (0x0014)
Display vertical timing setting0

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x26b	dsp_v_period Panel display scanning vertical period.
15:8	RO	0x0	reserved
7:0	RW	0x04	dsp_vsync_width Panel display scanning vsync(GOE) pulse width.

EBC_DSP_VTIMING1
Address: Operational Base + offset (0x0018)
Display vertical timing setting0

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x261	dsp_vact_end Panel display scanning vertical active end point
15:8	RO	0x0	reserved
7:0	RW	0x08	dsp_vact_st Panel display scanning vertical active start point

EBC_DSP_ACT_INFO

Address: Operational Base + offset (0x001c)
Display active width/height

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x258	dsp_height Display height
15:11	RO	0x0	reserved
10:0	RW	0x0c8	dsp_width Display width

EBC_WIN_CTRL
Address: Operational Base + offset (0x0020)
Window control register

Bit	Attr	Reset Value	Description
31:18	RO	0x0	reserved
17	RW	0x0	win_en Win enable
16:12	RW	0x00	ahb_incr_num Ahb master Incrnum
11:9	RW	0x7	ahb_trans_type Ahb master burst type
8:2	RW	0x70	win_fifo_almost_full_level Win fifo almost full level
1	RO	0x0	reserved
0	RW	0x0	win_fmt win_fmt [1:0] win source data format: 11: RGB888(24bpp) 10: RGB565(16bpp) 01: Y-data(8bpp) 00: Y-data(4bpp)

EBC_WIN_MST0
Address: Operational Base + offset (0x0024)
Old window layer memory start address

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	win_mst0 Start address of current image data in memory

EBC_WIN_MST1
Address: Operational Base + offset (0x0028)
New window layer memory start address

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	win_mst1 Start address of next image data in memory

EBC_WIN_VIR
Address: Operational Base + offset (0x002c)
Window layer virtual width

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:0	RW	0x0320	win_vir_width Win virtual width

EBC_WIN_ACT
Address: Operational Base + offset (0x0030)
Window active width/height

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x258	win_act_height Win active height
15:11	RO	0x0	reserved
10:0	RW	0x320	win_act_width Win active width

EBC_WIN_DSP
Address: Operational Base + offset (0x0034)
Window display width/height

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x258	win_dsp_height Win display height
15:11	RO	0x0	reserved
10:0	RW	0x320	win_dsp_width Win display width

EBC_WIN_DSP_ST
Address: Operational Base + offset (0x0038)
Window display start position

Bit	Attr	Reset Value	Description
31:27	RO	0x0	reserved
26:16	RW	0x008	win_dsp_y_st Win display y start point
15:11	RO	0x0	reserved
10:0	RW	0x00d	win_dsp_x_st Win display x start point

EBC_INT_CTRL
Address: Operational Base + offset (0x003c)
Interrupt control register

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:9	RW	0x00	frm_flag_num Frame number of the scanning flag interrupt The display frame number when the flag interrupt occur, the range is (0~ DSP_FRM_TOTAL-1).
8	W1C	0x0	frm_flag_int_clr Frame flag Interrupt clear After be set to 1, this bit will clear by itself 1 cycle later.
7	W1C	0x0	dsp_end_int_clr Display end interrupt clear After be set to 1, this bit will clear by itself 1 cycle later.
6	W1C	0x0	frm_end_int_clr Frame end interrupt clear After be set to 1, this bit will clear by itself 1 cycle later.
5	RW	0x1	frm_flag_int_msk Frame flag Interrupt Mask 0: unmask; 1: mask;
4	RW	0x1	dsp_end_int_msk Display end interrupt mask 0: unmask; 1: mask;
3	RW	0x1	frm_end_int_msk Frame end interrupt mask 0: unmask; 1: mask;
2	RO	0x0	frm_flag_int_sta Frame flag Interrupt status
1	RO	0x0	dsp_end_int_sta Display end interrupt status
0	RO	0x0	frm_end_int_sta Frame end interrupt status

EBC_VCOM0

Address: Operational Base + offset (0x0040)

VCOM0

Bit	Attr	Reset Value	Description

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	vcom0 VCOM [1:0] values in 0-16 frame. The VCOM [1:0] indicate the COM voltage in one frame. The VCOM can be different in different frame. So we can set the VCOM [1:0] sequence using display frames before start the display.

EBC_VCOM1
Address: Operational Base + offset (0x0044)
VCOM1

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	vcom1 VCOM [1:0] values in 16-32 frame. The VCOM [1:0] indicate the COM voltage in one frame. The VCOM can be different in different frame. So we can set the VCOM [1:0] sequence using display frames before start the display.

EBC_VCOM2
Address: Operational Base + offset (0x0048)
VCOM2

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	vcom2 VCOM [1:0] values in 32-48 frame. The VCOM [1:0] indicate the COM voltage in one frame. The VCOM can be different in different frame. So we can set the VCOM [1:0] sequence using display frames before start the display.

EBC_VCOM3
Address: Operational Base + offset (0x004c)
VCOM3

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	vcom3 VCOM [1:0] values in 48-64 frame. The VCOM [1:0] indicate the COM voltage in one frame. The VCOM can be different in different frame. So we can set the VCOM [1:0] sequence using display frames before start the display.

EBC_CONFIG_DONE

Address: Operational Base + offset (0x0050)
CONFIG finish register

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	W1C	0x0	config_done config finish In the first setting of the register, the new value was saved into the mirror register. When all the register config finish, writing this register to enable the copyright of the mirror register to real register. Then register would be updated at the start of every frame.

EBC_LUT_ADDR_MAP
Address: Operational Base + offset (0x1000)
LUT address map

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	lut_addr_map LUT memory address map (8k Byte)

Chapter 25 Serial Flash Controller(SFC)

25.1 Overview

The serial flash controller (SFC) is used to control the data transfer between the chip system and the serial nor/nand flash device.

25.1.1 Features

The SFC supports the following features:

- Support AHB slave interface to configure register and read/write serial flash
- Support AHB master interface to transfer data from/to SPI flash device
- Support AHB burst with incr4x32bits, or incr x32bits
- Support two independent clock domain: AHB clock and SPI clock
- Support x1,x2,x4 data bits mode
- Support up to 4 chip select
- Support interrupt output, interrupt maskable
- Support Spansion, MXIC, Gigadevice…vendor's nor flash memory.

25.2 Block Diagram

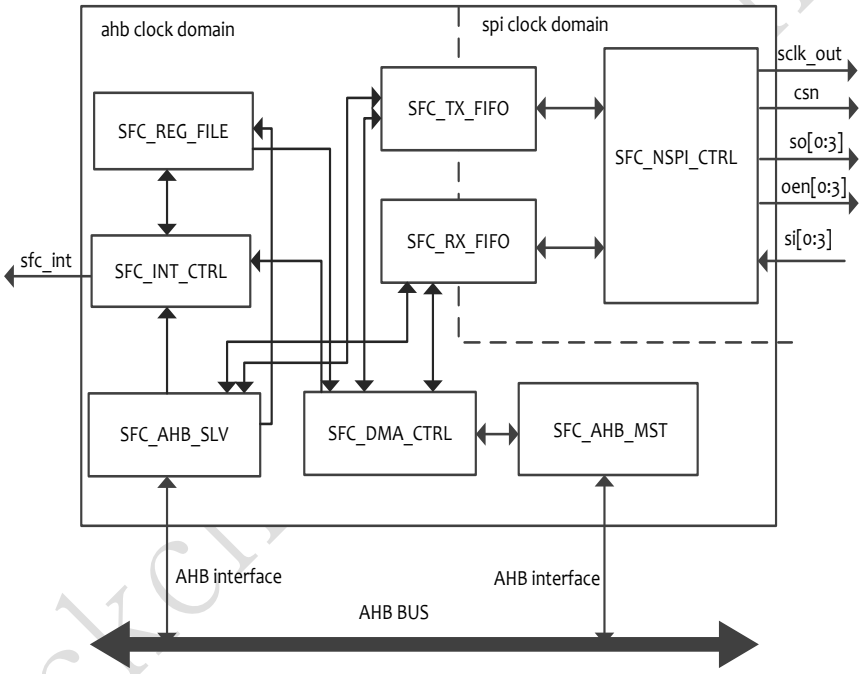


Fig 25-1 SFC architecture

25.3 Function description

25.3.1 SFC AHB slave

The AHB slave is used to configure the register, and also write to/read from the serial nor/nand flash device.

The SFC_CTRL register is a global control register, when the controller is in busy state(SFC_SR), SFC_CTRL cannot be set. The field sclk_idle_level_cycles of this register is used to configure the idle level cycles of sclk before read the first bit of the read command. Like the following picture shows: the red line of the sclk is the idle cycles, during these cycles, the chip pad is switched to output. When sclk_idle_level_cycles=0, it means there will be not such idle level.

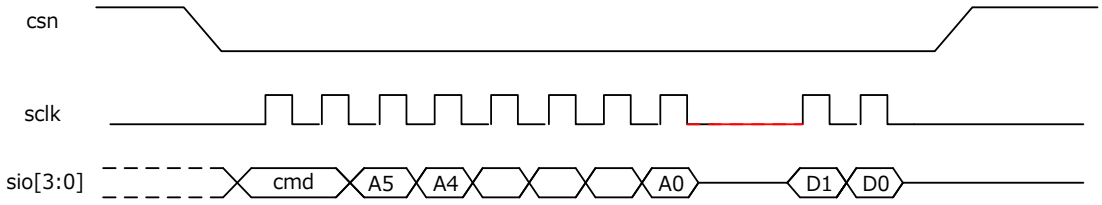


Fig 25-2 idle cycles

When the field spi mode is set, the transfer waveform will like following, and switch to mode3.

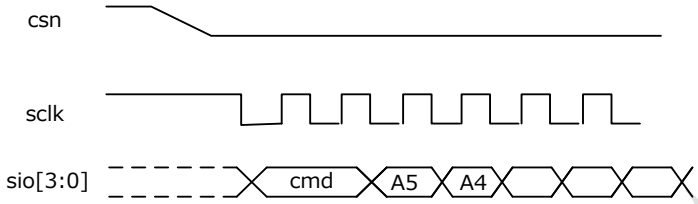


Fig 25-3 spi mode

25.4 Register Description

25.4.1 Register Summary

Name	Offset	Size	Reset Value	Description
SFC_CTRL	0x0000	W	0x00000000	Control Register 0
SFC_IMR	0x0004	W	0x00000000	Interrupt Mask
SFC_ICLR	0x0008	W	0x00000000	Interrupt Clear
SFC_FTLR	0x000c	W	0x00000000	FIFO Threshold Level
SFC_RCVR	0x0010	W	0x00000000	SFC Recover
SFC_AX	0x0014	W	0x00000000	SFC AX Value
SFC_ABIT	0x0018	W	0x00000000	Flash Address bits
SFC_ISR	0x001c	W	0x00000000	Interrupt Status
SFC_FSR	0x0020	W	0x00000002	FIFO Status
SFC_SR	0x0024	W	0x00000000	SFC Status
SFC_DMATR	0x0080	W	0x00000000	DMA Trigger
SFC_DMAADDR	0x0084	W	0x00000000	DMA Address
SFC_CMD	0x0100	W	0x00000000	SFC CMD
SFC_ADDR	0x0104	W	0x00000000	SFC Address
SFC_DATA	0x0108	W	0x00000000	SFC DATA

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

25.4.2 Detail Register Description

SFC_CTRL

Address: Operational Base + offset (0x0000)
Control Register 0

Bit	Attr	Reset Value	Description
31:14	RO	0x0	reserved

Bit	Attr	Reset Value	Description
13:12	RW	0x0	DATB Data bits width Data bits width 00: 1bit 01: 2bits 10: 4bits
11:10	RW	0x0	ADRB Address bits width Address bits width 00: 1bit 01: 2bits 10: 4bits
9:8	RW	0x0	CMDB Command bits width Command bits width 00: 1bit 01: 2bits 10: 4bits 11:reserved
7:4	RW	0x0	IDLE_CYCLE Sclk Idle Level Cycles 4'b0: idle hold is disable 4'b1: hold the sclk_out in idle for two cycles when switch to shift in ....
3:2	RO	0x0	reserved
1	RW	0x0	SHIFTPHASE Shift Phase Select 0: shift in the data at posedgesclk_out 1: shift in the data at negedgesclk_out
0	RW	0x0	SPIM SPI MODE Select SPI MODE Select 0: mode 0 1: mode 3

SFC_IMR
Address: Operational Base + offset (0x0004)
Interrupt Mask

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	DMAM DMA finish interrupt mask 0: dma_intr interrupt is not masked 1: dma_intr interrupt is masked

Bit	Attr	Reset Value	Description
6	RW	0x0	NSPIM SPI Error interrupt mask 0: nspi_intr interrupt is not masked 1: nspi_intr interrupt is masked
5	RW	0x0	AHBM AHB Error interrupt mask 0: ahb_intr interrupt is not masked 1: ahb_intr interrupt is masked
4	RW	0x0	TRANSM Transfer finish Interrupt Mask 0: transf_intr interrupt is not masked 1: transf_intr interrupt is masked
3	RW	0x0	TXEM Transmmit FIFO Empty Interrupt Mask 0: txe_intr interrupt is not masked 1: txe_intr interrupt is masked
2	RW	0x0	TXOM Transmmit FIFO Overflow Interrupt Mask 0: txo_intr interrupt is not masked 1: txo_intr interrupt is masked
1	RW	0x0	RXUM Receive FIFO Underflow Interrupt Mask 0: rxu_intr interrupt is not masked 1: rxu_intr interrupt is masked
0	RW	0x0	RXFM Receive FIFO Full Interrupt Mask 0: rxf_intr interrupt is not masked 1: rxf_intr interrupt is masked

SFC_ICLR

Address: Operational Base + offset (0x0008)

Interrupt Clear

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	W1C	0x0	DMAC DMA finish Interrupt Clear Write and clear
6	W1C	0x0	NSPIC SPI Error Interrupt Clear Write and clear
5	W1C	0x0	AHBC AHB Error Interrupt Clear Write and clear

Bit	Attr	Reset Value	Description
4	W1C	0x0	TRANSC Transfer Finish Interrupt Clear Write and clear
3	W1C	0x0	TXEC Transmmit FIFO Empty Interrupt Clear Write and clear
2	W1C	0x0	TXOC Transmmit FIFO Overflow Interrupt Clear Write and clear
1	W1C	0x0	RXUC Receive FIFO Underflow Interrupt Clear Write and clear
0	W1C	0x0	RXFC Receive FIFO Full Interrupt Clear Write and clear

SFC_FTLR
Address: Operational Base + offset (0x000c)
FIFO Threshold Level

Bit	Attr	Reset Value	Description
31:16	RO	0x0	reserved
15:8	RW	0x00	RXFTLR Receive FIFO Threshold Level When the number of receive FIFO entries is bigger than or equal to this value, the receive FIFO full interrupt is triggered.
7:0	RW	0x00	TXFTLR Transmit FIFO Threshold Level When the number of transmit FIFO entries is less than or equal to this value, the transmit FIFO empty interrupt is triggered.

SFC_RCVR
Address: Operational Base + offset (0x0010)
SFC Recover

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RO	0x0	RCVR SFC Recover SFC Recover Write 1 to recover the SFC State Machine, FIFO state and other logic state.

SFC_AX
Address: Operational Base + offset (0x0014)

SFC AX Value

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7:0	RO	0x00	AX AX The AX Value when doing the continuous read(enhance mode).

SFC_ABIT

Address: Operational Base + offset (0x0018)

Flash Address bits

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	RW	0x0	ABIT Flash Address bits Flash Address bits

SFC_ISR

Address: Operational Base + offset (0x001c)

Interrupt Status

Bit	Attr	Reset Value	Description
31:8	RO	0x0	reserved
7	RW	0x0	DMAS DMA Finish Interrupt Status 0: not active 1: active
6	RW	0x0	NSPIS SPI Error Interrupt Status 0: not active 1: active
5	RW	0x0	AHBS AHB Error Interrupt Status 0: not active 1: active
4	RW	0x0	TRANSS Transfer finish Interrupt Status 0: not active 1: active
3	RW	0x0	TXES Transmmit FIFO Empty Interrupt Status 0: not active 1: active
2	RW	0x0	TXOS Transmmit FIFO Overflow Interrupt Status 0: not active 1: active

Bit	Attr	Reset Value	Description
1	RW	0x0	RXUS Receive FIFO Underflow Interrupt Status 0: not active 1: active
0	RW	0x0	RXFS Receive FIFO Full Interrupt Status 0: not active 1: active

SFC_FSR
Address: Operational Base + offset (0x0020)
FIFO Status

Bit	Attr	Reset Value	Description
31:21	RO	0x0	reserved
20:16	RW	0x00	RXWLVL RX FIFO Water Level 0: fifo is empty 1: 1 entry is taken ...
15:13	RO	0x0	reserved
12:8	RO	0x00	TXWLVL TX FIFO Water Level 0: fifo is full 1: left 1 entry ...
7:4	RO	0x0	reserved
3	RO	0x0	RXFS Receive FIFO Full Status 0: rxfifo is not full 1: rxfifo is full
2	RO	0x0	RXES Receive FIFO Empty Status 0: rxfifo is not empty 1: rxfifo is empty
1	RO	0x1	TXES Transmit FIFO Empty Status 0: txfifo is not empty 1: txfifo is empty
0	RO	0x0	TXFS Transmit FIFO Full Status 0: txfifo is not full 1: txfifo is full

SFC_SR
Address: Operational Base + offset (0x0024)

SFC Status

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	WO	0x0	SR SFC Status 0: SFC is idle 1: SFC is busy When busy, don't set the control register.

SFC_DMATR

Address: Operational Base + offset (0x0080)

DMA Trigger

Bit	Attr	Reset Value	Description
31:1	RO	0x0	reserved
0	W1C	0x0	DMATR DMA Trigger Write 1 to start the dma transfer.

SFC_DMAADDR

Address: Operational Base + offset (0x0084)

DMA Address

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	DMAADDR DMA Address DMA Address

SFC_CMD

Address: Operational Base + offset (0x0100)

SFC CMD

Bit	Attr	Reset Value	Description
31:30	RW	0x0	CS Flash chip select 00: chip select 0 01: chip select 1 10: chip select 2 11: chip select 3
29:16	RW	0x0000	TRB Transfer Bytes number Total Data Bytes number that will write to /read from the flash.
15:14	RW	0x0	ADDRB Address bits number select 00: 0bits 01: 24bits 10: 32bits 11: From the ABIT register

Bit	Attr	Reset Value	Description
13	RW	0x0	CONT Continuous read mode 0: disable continuous read mode 1: enable continuous read mode
12	RW	0x0	WR Flash Write or Read 0:read 1:write
11:8	WO	0x0	DUMM Dummy Bits Number Dummy Bits Number
7:0	WO	0x00	CMD Flash CMD Flash Command

SFC_ADDR
Address: Operational Base + offset (0x0104)
SFC Address

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	ADDR SFC Address Flash's address

SFC_DATA
Address: Operational Base + offset (0x0108)
SFC DATA

Bit	Attr	Reset Value	Description
31:0	RW	0x00000000	ADDR SFC DATA Flash's Data

25.5 Interface Description

Table 25-1 IOMUX Settings for SFC

Module Pin	Direction	Pad Name	IOMUX Setting
sfc_clk	O	IO_EMMCd6_SFCclk_JTG1tdo_GPIO0b1	GRF_GPIO0B_IOMUX[3:2]=2'b10
sfc_csn	O	IO_EMMCd7_SFCcs_JTG1trst_GPIO0b2	GRF_GPIO0B_IOMUX[5:4]=2'b10
sfc_sio0	I/O	IO_EMMCd5_SFCd0_JTG1tdi_GPIO0b0	GRF_GPIO0B_IOMUX[1:0]=2'b10
sfc_sio1	I/O	IO_EMMCd4_SFCd1_GPIO0a7	GRF_GPIO0A_IOMUX[15:14]=2'b10
sfc_sio2	I/O	IO_EMMCd3_SFCd2_I2C0Cscl_GPIO0a6	GRF_GPIO0A_IOMUX[13:12]=2'

			b10
sfc_sio3	I/O	IO_EMMCd2_SFCd3_I2C0Csda_GPIO0a5	GRF_GPIO0A_IOMUX[11:10]=2'b10

*Notes: Direction: **I**– Input, **O**– Output, **I/O**– Input/Output*

25.6 Application Notes

25.6.1 AHB Slave write flash flow

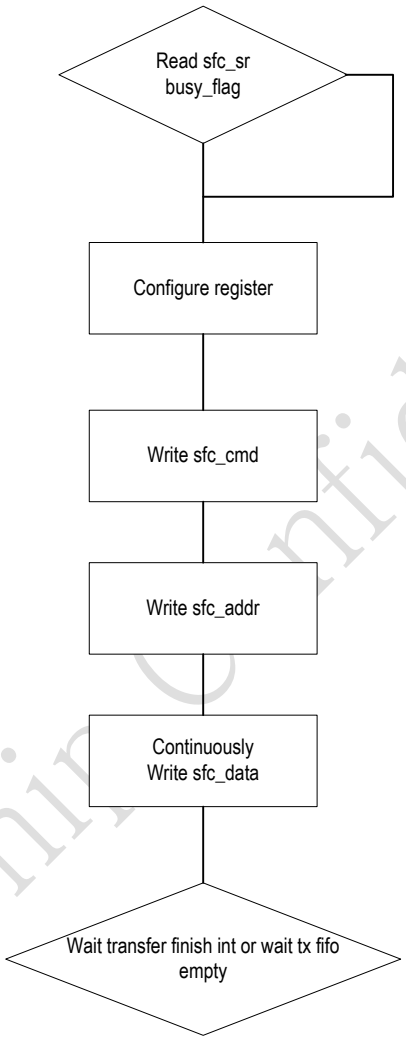


Fig 25-4 slave mode write

25.6.2 AHB Slave read flash flow

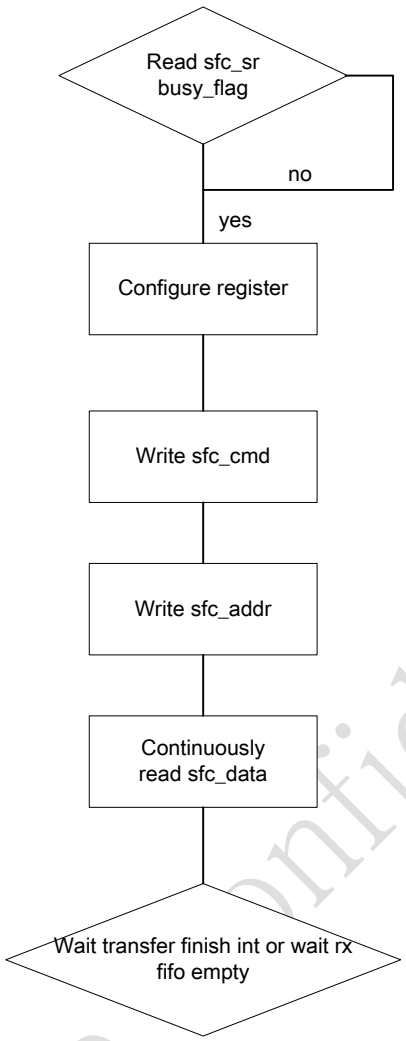


Fig 25-5 slave mode read

25.6.3 AHB dma transfer flow

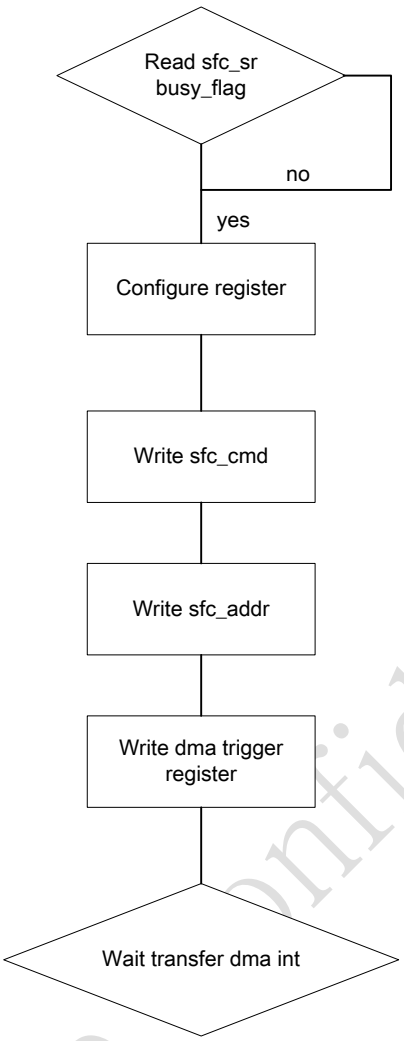


Fig 25-6 master mode flow

25.6.4 Software notice

The sfcsclk need to be kept under 200MHZ. It’s better to soft reset the sfc before data transfer.

Chapter 26 HIFI Accelerator(HIFIACC)

26.1 Overview

HIFIACC is a design of fixed point accelerator, including functions of FFT, MAC and LPC. LPC is used on the process of audio decode, for special APE, ALAC and FLAC decode.

26.1.1 Features

- FFT/IFFT
  - Support order 64, 128, 256, 512, 1024 FFT&IFFT
  - Support CPU&DMA memory address read/write
- MAC
  - Support each order number of 1 to 1024
  - Support 32x32 bit, and result is 64 bit
- FLAC
  - Order number of 1 to 32
  - CPU read/write, and DMA burst2/4/8/16
- ALAC
  - Order number of 1 to 32
  - CPU read/write, and DMA burst 2/4/8/16
- APE
  - Support compressed rate 2000/3000/4000/5000, under version 3990
  - Support compressed rate 2000/3000/4000, under version 3970
  - CPU read/write, and DMA burst 2/4/8/16
  - Support 192k * 24bit Sample rate, under clock 300MHZ

26.2 Block Diagram

The HIFIACC architecture is shown in the following figure.

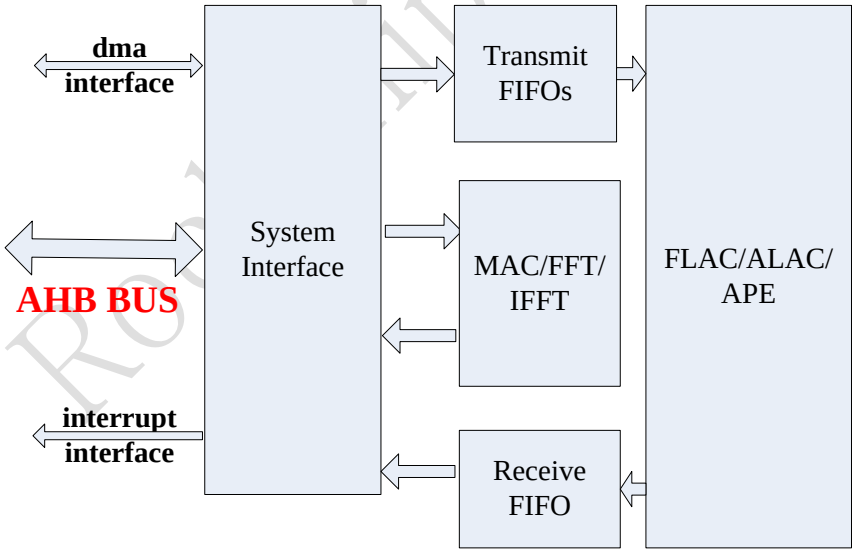


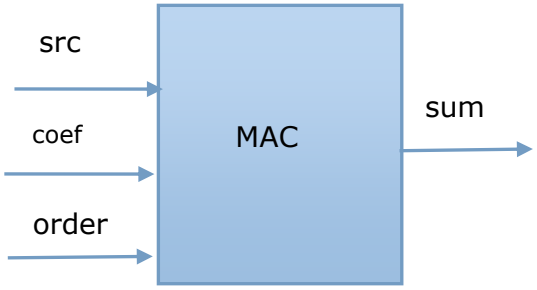
Fig 26-1 HIFIACC Block Diagram

1. FFT/IFFT through AHB interface and memory (address share with MAC), calculation result placed in the same address, can be accessed directly.
2. MAC 32x32 bit Multiplier A&B through AHB interface, share the same memory address with FFT/IFFT, 64 bit result write to register, can be accessed directly by CPU.
3. ALAC/FLAC/APE through AHB interface and depth of 32x32 bit FIFO.

26.3 Function Description

26.3.1 MAC

Mac data format and calculate



$$*sum = \sum_{k=1}^{count} coef_k src(k)$$

26.3.2 FFT/IFFT

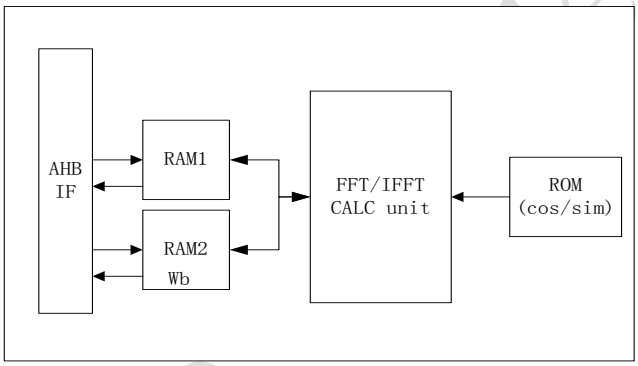


Fig 26-2 FFT/IFFT data flow and structure

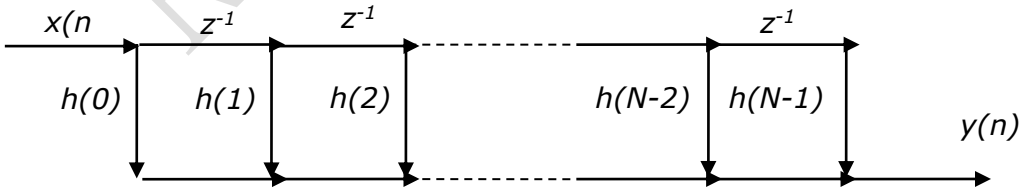
26.3.3 FLAC

FLAC is a coef fixed FIR. Coef h(n-1) to h(0) can be write/read directly by CPU&DMA.

$$y(n) = \sum_{m=0}^{N-1} h(m)x(n-m) = h(0)x(n) + h(1)x(n-1) + \cdots + h(N-1)x(n-(N-1))$$

(H(i) = h(N-1-i), 0 <= i<= N-1)

FLAC data format and structure:



26.3.4 ALAC

ALAC is a coef variable FIR. Initial Coef h(n-1) to h(0) can be write/read directly by CPU&DMA.

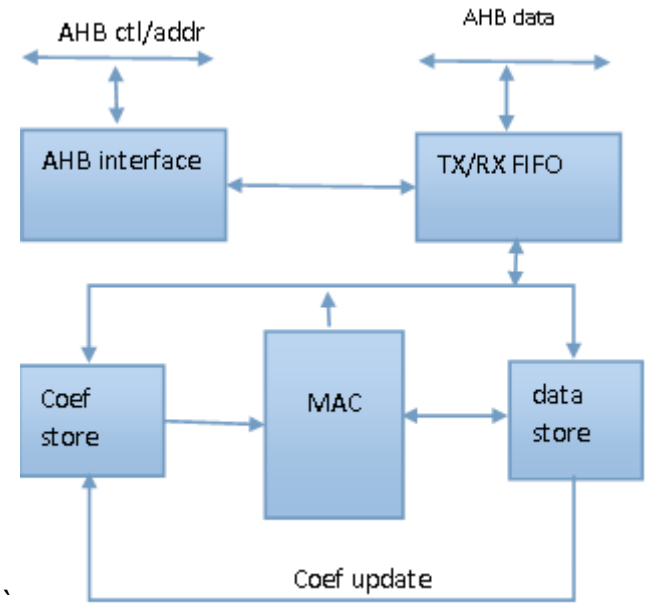


Fig 26-3 ALAC structure

Chapter 27 SYNTH

27.1 Overview

The Synth module is a hardware accelerator of MP3 decoder. It is designed for high performance and power efficient MP3 decoder. The Mp3 decoding is done by the Imdct36, Synth and CPU.

27.1.1 Features

- MP3 Subband Synthesis calculation cell
- AHB slave interface
- One 32x32-bits and SRAM0
- One 1024x20-bits memory-map SRAM1
- Interrupt output

27.2 Block Diagram

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

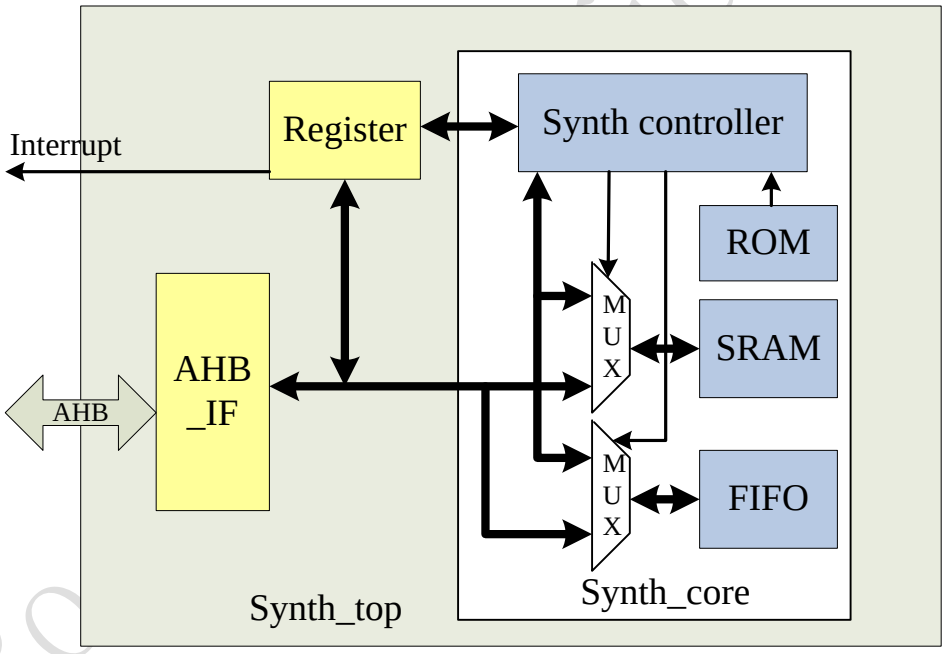


Fig 27-1 Synth Block Diagram

AHB Interface

The AHB Interface implements the AHB slave operation. It's bus width is 32 bits.

SRAM0 & SRAM1

There are two SRAM blocks in the Synth module.

The SRAM0 is a 32x32-bit internal sram. It is used as the input raw data buffer in the first stage of Synth operation, and used as result buffer in the last stage of Synth operation.

The SRAM1 is a 1024x20-bits internal sram used as a fifo buffer in Synth module.

The two SRAMs are memory-maped and can be access through AHB bus. The memory map of Synth is indicated the following figure.

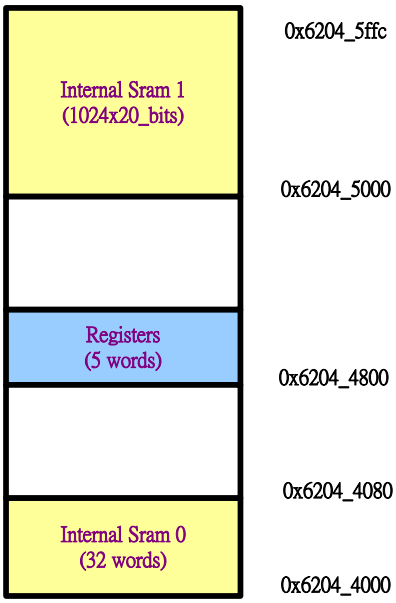


Fig 27-2 Synth Memory map

ROM

There is a 1024x20-bits ROM for coefficients used in Synth calculation.

Synth core

This block process the input data(in SRAM0) and history data(in SRAM1), calculate the subband synthesis result and write it back into SRAM0. The process depends on the channel & phase which are set in register before.

Chapter 28 Imdct36

28.1 Overview

The Imdct36 module is a hardware accelerator of MP3 decoder. It is designed for high performance and power efficient MP3 decoder. The MP3 decoding is done by the Imdct36, Synth and CPU.

28.1.1 Features

- Imdct36 calculation cell, support standard and pre-right-shift result output mode
- AHB slave interface
- 54x32-bit memory-map SRAM
- Interrupt output

28.2 Block Diagram

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

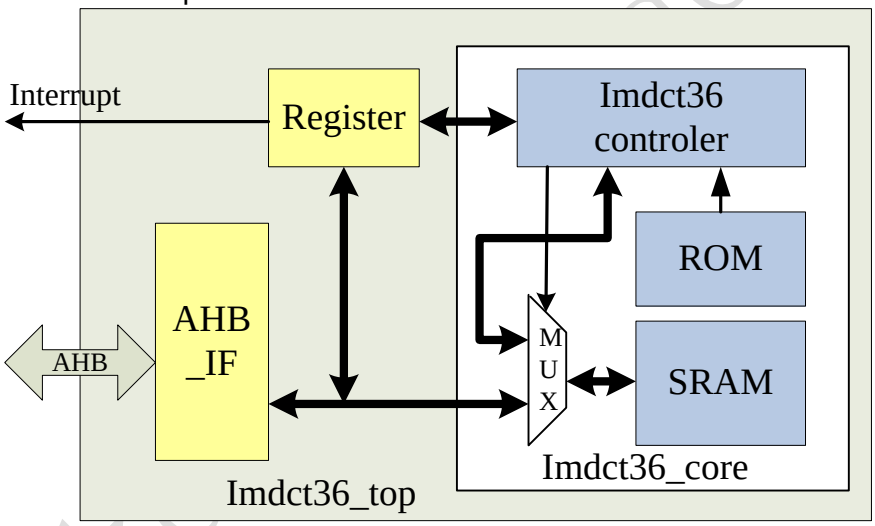


Fig 28-1 Imdct36 Block Diagram

AHB Interface

The AHB Interface implements the AHB slave operation. It's bus width is 32 bits.

SRAM

There is a 54x32-bit Internal SRAM in Imdct36 block. The first 18 words are used as the input raw data buffer, and the after 36 words are used to store the calculation results.

This SRAM is memory-maped and can be access through AHB bus. The Imdct36 memory map is indicated in the following figure.

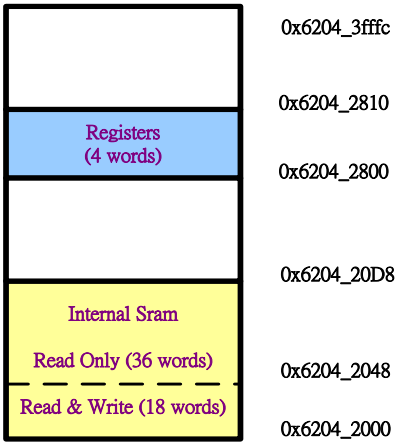


Fig 28-2 Imdct36 Memory map

ROM

There is a 32x20-bit ROM for coefficients used in Imdct36 calculation.

Imdct36 core

This block process the input data (in SRAM low 18-word) calculate the imdct36 result and write it back into SRAM high 36-word. In pre-right-shift output mode, the result has been right shifted 2 bits before write back.