BL616/BL618

Reference Manual

Version: 0.98

www.bouffalolab.com

Contents

List of Figures

List of Tables

1 System and Memory

1.1 System Architecture

BL616/BL618 series chips adopt RISC-V 32-bit CPU, equipped with 16KB D-cache and 32KB I-Cache, CPU frequency up to 320MHz, suitable for high-performance applications such as Internet of Things, embedded and artificial intelligence.

The main system consists of the following parts:

- Bus interface: AXI bus and AHB bus
  - CPU accesses memory through AXI bus
  - The CPU accesses peripherals through the AHB bus
  - With low power consumption, high performance and other characteristics
- memory
  - 480KB SRAM for data storage
  - 128KB ROM
  - Support embedded high-speed PSRAM, expand RAM capacity(Only for BL618)
- Peripherals
  - A total of 30 peripheral modules, including advanced peripherals such as USB/SDH/SDU/EMAC(Only for BL618)/CAM(Only for BL618)/Display(Only for BL618)

The Wi-Fi 6/BLE/Zigbee wireless subsystem is integrated on-chip, which can realize a variety of wireless connections and data transmission, and provide a variety of connection and transmission experiences.

The system architecture of BL616/BL618 is shown in the following figure:

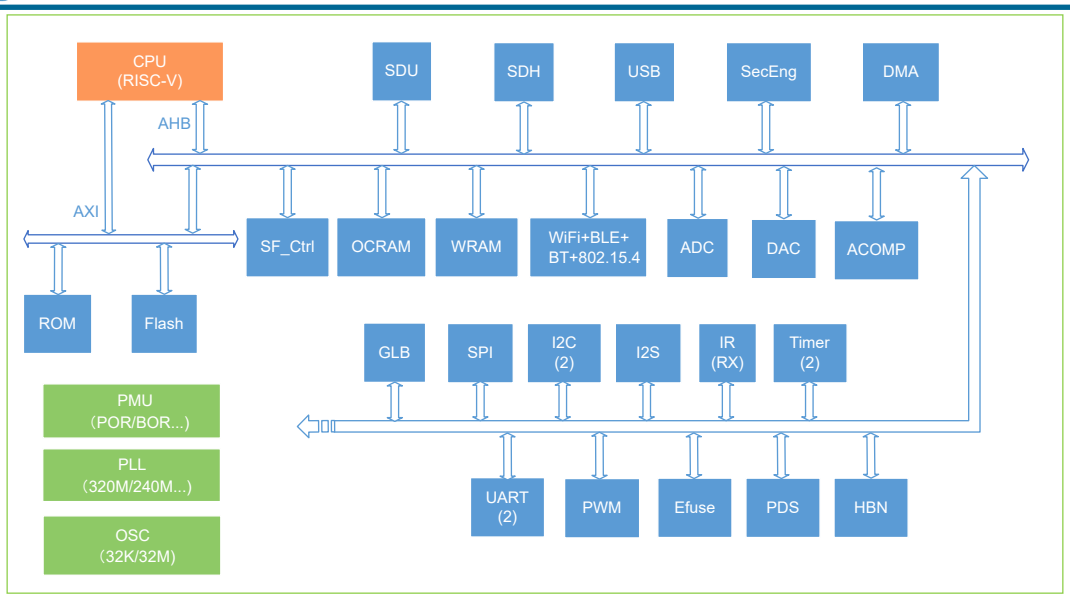


Fig. 1.1: BL616 System Architecture

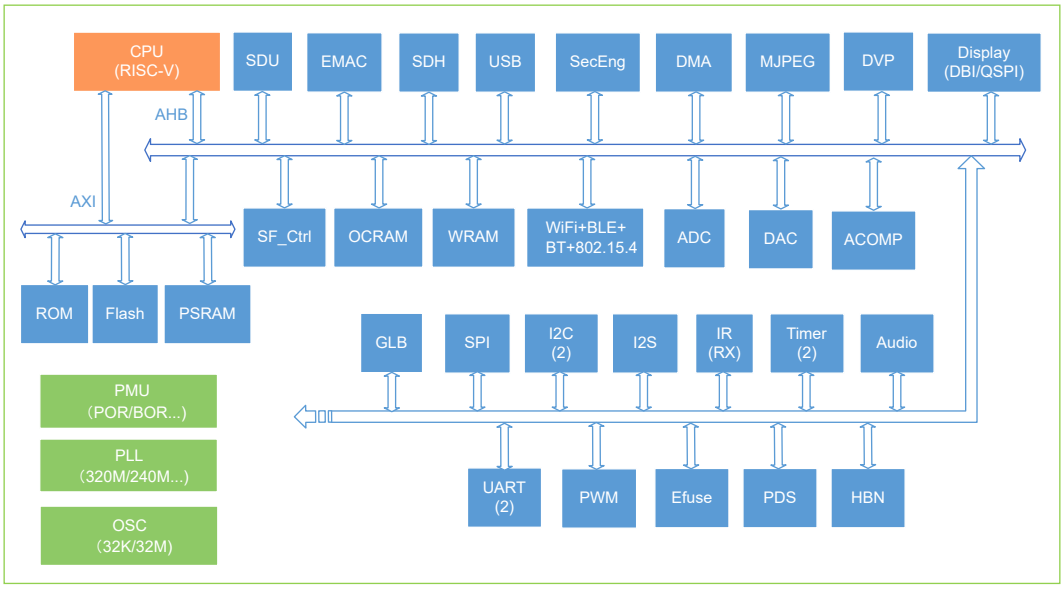


Fig. 1.2: BL618 System Architecture

1.2 CPU

1.2.1 Features

BL616/BL618 has a built-in 32-bit RISC-V CPU, uses a 5-stage pipeline: fetch, decode, execute, memory access, and write back. This high-performance embedded microprocessor has the following features:

- 32-bit RISC processor
- Supports RISC-V RV32IMAFCP instruction set

- Supports RISC-V 32-bit/16-bit mixed instruction set
- Supports RISC-V machine mode and user mode
- Thirty-two 32-bit integer general purpose registers (GPR) and thirty-two 32-bit/64-bit floating-point GPRs
- Integer (5-stage)/floating-point (7-stage), single-issue, sequentially executed pipeline
- Supports AXI 4.0 main device interface and AHB 5.0 peripheral interface
- 32KB instruction cache, two-way set associative structure
- 16KB data cache, two-way set associative structure
- Unaligned memory access
- Double-cycle hardware multiplier and Radix-4 hardware divider
- Supports BHT (8K) and BTB
- Supports extended enhanced instruction set
- Supports MCU feature extension technique, including interrupt processing acceleration technique and MCU extension feature
- Compatible with RISC-V CLIC interrupt standard and interrupt nesting; 96 external interrupt sources, 4 bits for configuring interrupt priority
- Compatible with RISC-V PMP, 8 configurable areas
- Supports hardware performance monitor (HPM) units
- Supports RISC-V Debug Specification

1.2.2 Extended Functions

- Bit operation instruction, arithmetic operation instruction, and enhanced memory access instruction
- CACHE operation instruction and synchronization instruction, making programming easy for software programmers
- Interrupt speculative push technique and vector interrupt tail-biting technique, accelerated interrupt response, with interrupt response delay of 20 processor clock cycles
- Common application requirements in MCU fields like NMI, low-power mode for deep/light sleep, low-power wake-up events, and locking
- Supports unaligned memory access, which can be enabled/disabled by software, making programming and debugging easy for software programmers

1.2.3 Implemented Standards

The processor is compatible with the RISC-V standard and the specific versions are:

- The RISC-V Instruction Set Manual, Volume I: RISC-V User-Level ISA, Version 2.2
- The RISC-V Instruction Set Manual, Volume II: RISC-V Privileged Architecture, Version1.10
- RISC-V Core-Local Interrupt Controller (CLIC) Version 0.8
- RISC-V External Debug Support Version 0.13.2
- RISC-V “P” Extension Proposal Version 0.9

1.3 BOOT process

This section will introduce the steps involved in the startup process of BL616/618, from power-on to the execution of the main function.

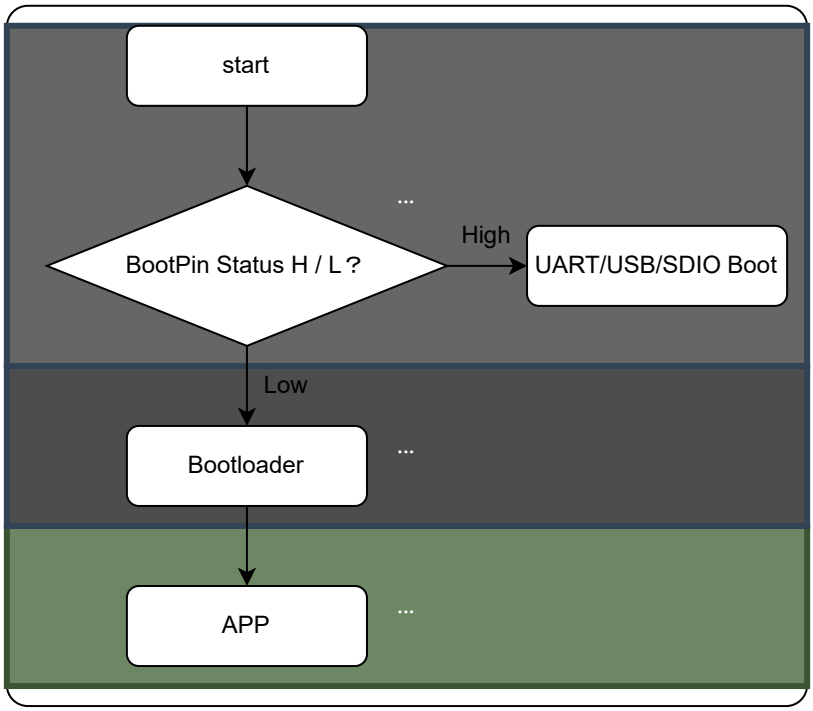


Fig. 1.3: Startup Flow Diagram

The chip boot process generally falls into two categories and can be divided into the following 2 to 3 steps:

The two scenarios are:

- Boot from Flash
- Boot from UART/USB/SDIO

Taking booting from flash as an example, the following describes the complete boot process of the chip in detail, which

is divided into three steps:

1. The primary bootloader is embedded in the internal ROM of the BL616/8. It reads the Firmware header (FW Header) from the flash's 0x0 offset address, completes chip-related configuration and checks based on the Header parameters, and after configuration, starts the secondary bootloader from the 0x2000 address.
2. The secondary bootloader reads the partition table from the flash and boots the latest main program firmware according to the partition table.
3. The application starts running, sets chip-related clocks, pins, and other configurations as needed by the user, and finally starts the RTOS scheduler.

1.3.1 Primary Bootloader

After the chip is reset or re-powered, the CPU will immediately start running, performing a series of initialization operations to restore the chip to an initial state. The initialization reset code is fixed in the mask ROM of the BL616/8 and cannot be changed.

During the boot process, the primary bootloader determines the boot mode of the BL616/8 based on the level state of the HBN_RSV0 register and GPIO2, where the HBN_RSV0 register holds the flag for chip wake-up, with the specific execution process as follows:

1. Reset from deep sleep mode: Check the HBN_RSV0 register to determine if a wake-up from deep sleep mode is requested. If so, wake up from deep sleep mode and enter the fast boot process. After verifying the flag in the HBN_RSV0 register, the program will use the boot parameters stored at the OCRAM starting address, and immediately jump to that address to run. If the flag in the verified HBN_RSV0 register is not valid, or if there is a boot anomaly, then it will continue to boot as in a normal power-on reset process.
2. Power-on reset, software PWRON reset: Check the GPIO2 pin level to determine if booting from a Flash image. If the GPIO2 pin level is 1 (High), it enters the UART/USB/SDIO boot mode, which is mainly for Flash writing or downloading images to RAM for execution (wireless pass-through scenario). If the GPIO2 pin level is 0 (Low), it enters the Flash boot mode, and after entering this mode, the secondary bootloader is generally run.
3. Secure boot: According to the configuration in efuse, start the secure boot process, which checks the integrity and legality of the secondary bootloader's signature, and decides whether to enable flash decryption.

1.3.2 Secondary Bootloader

The image stored from the flash's 0x0 offset address (including FW header) is the secondary bootloader. The source code of the secondary bootloader can be found in the boot2_isp directory of the released SDK. The secondary bootloader can easily plan flash partition information through the partition table, and it supports secure boot, encrypted boot, signature verification, and Over-The-Air (OTA) update functions for user applications.

After the primary bootloader starts, it reads the FW Header from the beginning of the image, completes related chip configuration and checks according to the FW Header, such as checking for encrypted boot, etc.; after configuration, it jumps to run the secondary bootloader.

After the secondary bootloader starts, it reads the partition table information from the flash's 0xE000 offset address by default (configurable). The default design uses A/B dual partition tables, and the application firmware entity location will also be in two separate A/B partitions, which facilitates OTA. The secondary bootloader will determine which partition to boot based on the active_index and age information in the partition table. See the partition table instructions for specific reference.

For the partition that will be booted, the secondary bootloader will read the FW Header from the flash, check the integrity and legality of the application's signature, and decide whether to enable flash decryption. Finally, it jumps to execute the application.

1.3.3 Application Boot Stage

The application boot includes all processes from the start of the application's execution to before the user's APP *main* function. Mainly divided into two stages:

- Power, clock, and basic C language runtime environment initialization.
- Run the main function and call *main*.

1.3.3.1 Hardware and System Runtime Environment Initialization

The application entry starts executing from the *_start* within the *start.S* file. This initialization function establishes a basic C Runtime (CRT) and configures the internal hardware of the chip.

- Reconfigures the CPU exception vector for the application.
- Initializes the Floating Point Unit (FPU).
- Sets the stack pointer.
- Initializes internal memory (data and bss sections).
- Restores the CPU interrupt status register to its default state and enables interrupts.
- Enables the cache.
- If memory protection is configured, memory protection is initialized.

After these operations are completed, the user's *main* function is called.

1.3.3.2 Running the Main Task

Once the necessary runtime environment is initialized, it transitions to the user program. Typically, the first segment of the user program calls the *board_init* function from *board.c*, completing the initialization of project-specific ports, potentially including UART, GPIO, and other peripheral ports.

After the basic hardware port initialization is complete, users can choose to run a bare-metal system or the FreeRTOS operating system, depending on their needs. If wireless functionality is used, FreeRTOS must be enabled, and the user will need to create tasks to run the different features. If running bare-metal, typically the user's functionality runs within a while loop.

1.3.4 Booting from UART/USB/SDIO

Booting from UART/USB/SDIO, commonly referred to as booting from interface mode (or programming mode), is a branch scenario after the primary bootloader starts. This mode is mainly used for Flash programming or downloading images to RAM for execution (in wireless pass-through scenarios).

The system supports boot from Flash/UART/USB/SDIO, as described below:

Table 1.1: Boot mode

Boot Pin	Level	Description
GPIO2	1	Boot from UART(GPIO21/22)/USB/SDIO, this mode is mainly used for flash programming or downloading image to RAM for execution (wireless transparent transmission scenario)
	0	Boot the application image from Flash

1.4 Address Mapping

Table 1.2: Memory address map

Module	Size	Base Address	
		Cache	Non-cache
OCRAM	320KB	0x62FC0000	0x22FC0000
WRAM	160KB	0x63010000	0x23010000

OCRAM and WRAM can be accessed either through the AHB bus or through AXI. When the CPU uses the 0x62FC0000 address to access the OCRAM, it will go through the internal cache and access the OCRAM through AXI to AHB. When the CPU uses the 0x22FC0000 address to access the OCRAM, it will directly access the OCRAM through the AHB bus.

Table 1.3: Address mapping

Module	Target	Base Address	Size	Description
FLASH	Flash	0xA0000000	128MB	Application address space
PSRAM	pSRAM	0xA8000000	128MB	pSRAM memory address space (optional, depends on the specific chip model,Only for BL618)
RAM	HBN RAM	0x20010000	4KB	HBN RAM,mainly used for data saving in ultra-low power mode
Peripheral	USB	0x20072000	4KB	USB High Speed OTG Control Register
	EMAC	0x20070000	4KB	EMAC Control Register(Only for BL618)
	SDH	0x20060000	4KB	SDH Control Register
	MJPEG	0x20059000	4KB	MJPEG Control Register(Only for BL618)
	CAM	0x20057000	4KB	Camera interface Control Register(Only for BL618)
	Efuse	0x20056000	4KB	Efuse storage Control Register
	AUDIO DAC	0x20055000	4KB	Audio DAC Control Register
	PSRAM_Ctrl	0x20052000	4KB	PSRAM Control Register(Only for BL618)
	HBN	0x2000F000	4KB	Hibernate register
	PDS	0x2000E000	4KB	Power-down sleep register
	SDU	0x2000D000	4KB	SDU Control Register
	DMA	0x2000C000	4KB	DMA Control Register
	SF_Ctrl	0x2000B000	4KB	Serial Flash Control Register
	Audio ADC	0x2000A000	256B	Audio ADC Control Register
	I2S	0x2000AB00	256B	I2S Control Register
	I2C1	0x2000A900	256B	I2C1 Control Register
	Display	0x2000A800	256B	Display Control Register(Only for BL618)
	IRR	0x2000A600	256B	IR Receiver Control Register
	TIMER	0x2000A500	256B	TIMER Control Register
	PWM	0x2000A400	256B	PWM Control Register
	I2C0	0x2000A300	256B	I2C0 Control Register
	SPI	0x2000A200	256B	SPI Control Register
	UART1	0x2000A100	256B	UART1 Control Register
	UART0	0x2000A000	256B	UART0 Control Register
	TZ	0x20005000	4KB	TrustZone Control Register
	SEC_ENG	0x20004000	4KB	Security Engine Control Register
	GPIP	0x20002000	1KB	General Purpose DAC/ADC/ACOMP Interface Control Register
	GLB	0x20000000	4KB	Global control register
ROM	ROM	0x90000000	128KB	Bootrom address space

1.5 Interrupt Source

A total of 64 interrupt sources are included, and the interrupt sources and corresponding interrupt numbers are shown in the following table:

Table 1.4: Interrupt assignment

Interrupt source		Interrupt Number	Description
BMX	BUS Error	IRQ_NUM_BASE+0	BUS Error Respones Interrupt
	BUS Timeout	IRQ_NUM_BASE+1	BUS Respones Timeout Interrupt
Display	Display	IRQ_NUM_BASE+2	Display All Interrupt
SDU	SDU Software Reset	IRQ_NUM_BASE+3	SDU Reset Triggered by Host
Audio ADC	Audio ADC	IRQ_NUM_BASE+4	Audio ADC Interrupt
RF	RF Interrupt0	IRQ_NUM_BASE+5	RF Interrupt0
	RF Interrupt1	IRQ_NUM_BASE+6	RF Interrupt1
SDU	SDU Side Interrupt	IRQ_NUM_BASE+7	SDU Side All Interrupt
Wi-Fi	TBTT SLEEP	IRQ_NUM_BASE+8	Wi-Fi TBTT SLEEP Interrupt
SecEng	Group0	IRQ_NUM_BASE+9	Group0 SHA/AES/TRNG/PKA/GMAC Interrupt
	Group1	IRQ_NUM_BASE+10	Group1 SHA/AES/TRNG/PKA/GMAC Interrupt
	Group0 CDET	IRQ_NUM_BASE+11	Group0 CDET Interrupt
	Group1 CDET	IRQ_NUM_BASE+12	Group1 CDET Interrupt
SF Ctrl	Group0	IRQ_NUM_BASE+13	SF_Ctrl Group0 Interrupt
	Group1	IRQ_NUM_BASE+14	SF_Ctrl Group1 Interrupt
DMA	DMA0_ALL	IRQ_NUM_BASE+15	DMA0 ALL Interrupt
CAM_-OUT0	CAM_OUT0	IRQ_NUM_BASE+16	CAM_OUT0 Interrupt
SDH	SDH All Interrupt	IRQ_NUM_BASE+17	SDH All Interrupt
CAM_-OUT1	CAM_OUT1	IRQ_NUM_BASE+18	CAM_OUT1 Interrupt
Wi-Fi	TBTT WAKEUP	IRQ_NUM_BASE+19	Wi-Fi TBTT WAKEUP Interrupt
IR	IRRX	IRQ_NUM_BASE+20	IR RX Interrupt
USB	USB	IRQ_NUM_BASE+21	USB Interrupt
Audio DAC	Audio DAC	IRQ_NUM_BASE+22	Audio DAC Interrupt
MJPEG	Encoder	IRQ_NUM_BASE+23	MJPEG Encoder All Interrupt
EMAC	EMAC	IRQ_NUM_BASE+24	EMAC Interrupt
GPADC	GPADC_DMA	IRQ_NUM_BASE+25	GPADC_DMA Interrupt
Efuse	Efuse	IRQ_NUM_BASE+26	Efuse Interrupt
SPI	SPI	IRQ_NUM_BASE+27	SPI Interrupt
UART	UART0	IRQ_NUM_BASE+28	UART0 Interrupt

Table 1.4: Interrupt assignment (continued)

Interrupt source		Interrupt Number	Description
	UART1	IRQ_NUM_BASE+29	UART1 Interrupt
GPIO	GPIO_DMA	IRQ_NUM_BASE+31	GPIO DMA Interrupt
I2C0	I2C0	IRQ_NUM_BASE+32	I2C0 Interrupt
PWM	PWM	IRQ_NUM_BASE+33	PWM Interrupt
TIMER0	TIMER0_CH0	IRQ_NUM_BASE+36	Timer0 Channel 0 Interrupt
	TIMER0_CH1	IRQ_NUM_BASE+37	Timer0 Channel 1 Interrupt
	TIMER0_WDT	IRQ_NUM_BASE+38	Timer0 Watch Dog Interrupt
I2C1	I2C1	IRQ_NUM_BASE+39	I2C1 Interrupt
I2S	I2S	IRQ_NUM_BASE+40	I2S Interrupt
	Reserved	IRQ_NUM_BASE+41	Reserved
	Reserved	IRQ_NUM_BASE+42	Reserved
XTAL	Xtal Ready	IRQ_NUM_BASE+43	Xtal Ready Interrupt
GPIO	GPIO_INT0	IRQ_NUM_BASE+44	GPIO Interrupt
DM	DM	IRQ_NUM_BASE+45	DM Interrupt
BT	BT	IRQ_NUM_BASE+46	BT Interrupt
MAC154	ENH Ack	IRQ_NUM_BASE+47	MAC154 ENH Ack Interrupt
	Others	IRQ_NUM_BASE+48	MAC154 Other Interrupt
	AES	IRQ_NUM_BASE+49	MAC154 AES Interrupt
PDS	PDS	IRQ_NUM_BASE+50	PDS Interrupt
HBN	HBN OUT0	IRQ_NUM_BASE+51	HBN Out 0 Interrupt
	HBN OUT1	IRQ_NUM_BASE+52	HBN Out 1 Interrupt
BOR	BOR	IRQ_NUM_BASE+53	Brown Out Reset Interrupt
Wi-Fi	Wi-Fi	IRQ_NUM_BASE+54	Wi-Fi Interrupt
BZ Phy	BZ Phy	IRQ_NUM_BASE+55	BZ Phy Interrupt
BLE	BLE	IRQ_NUM_BASE+56	BLE Interrupt
WiFi	MAC TR Timer	IRQ_NUM_BASE+57	MAC TX&RX Timer Interrupt
	MAC TR MISC	IRQ_NUM_BASE+58	MAC TX&RX Misc Interrupt
	MAC RX Trigger	IRQ_NUM_BASE+59	MAC RX Trigger Interrupt
	MAC TX Trigger	IRQ_NUM_BASE+60	MAC TX Trigger Interrupt
	MAC General	IRQ_NUM_BASE+61	MAC General Interrupt
	MAC Prot	IRQ_NUM_BASE+62	MAC Prot Interrupt
	IPC	IRQ_NUM_BASE+63	MAC IPC Interrupt

Note: IRQ_NUM_BASE is 16 and the interrupt number 015 is RISC-V reserved interrupt.

1.6 Peripheral Overview

Table 1.5: Peripheral List

Peripheral	Number	Note
GPIO	19/35	QFN40 corresponds to 19 GPIOs, QFN56 corresponds to 35 GPIOs
UART	2	Support RTS/CTS
SPI	1	Support Master/Slave mode
I2C	2	Support Master mode
I2S	1	Support Left-Justified/Right-Justified/Normal I2S/DSP and other data formats
PWM	4	Support adjustable output polarity, dual threshold value setting
Timer	2	Support FreeRun mode and PreLoad mode
DMA	4	Support LLI linked list function
IR	1	Support receiving, the protocol includes NEC and RC-5, and also supports receiving data by pulse width counting
Audio PWM	1	Support audio playback
Audio ADC	1	Support recording
EMAC	1	Supports 100Mbps
CAM	1	Support image rectangle crop
MJPEG	1	Supports arbitrary quantization tables
DBI	1	Support Type B/Type C 3-wire/Type C 4-wire, and also integrates QSPI mode
SDH	1	Support high-speed SD card
SDU	1	Support CCCR (function0) and function1, support SDU soft reset, function1 has 16 port receive buffers
SEC_ENG	1	Support AES/SHA/GMAC/TRNG
USB	1	USB2.0

2 Reset and Clock

2.1 Overview

The clock sources in the chip include XTAL, PLL, and RC. They are sent to each module along with frequency division configuration.

2.2 Reset Management

Chip reset:

- CPU reset: Only CPU is reset. The program will roll back, and peripherals will not be reset
- System reset: All peripherals and CPUs will be reset, but the registers in the AON field will not be reset
- Power-on reset: The whole system including the registers in the AON field will be reset

The application can select a proper reset mode as required.

Table 2.1: Software reset function table

BL616	CHIP_EN	Watch Dog/ PDS MISC/ Software Power On (swrst_cfg2[0])	Software Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS SOC	CPU Reset (swrst_cfg2[1])/ PDS NP
CPU	✓	✓			✓
BUS	✓	✓		✓	
GLB	✓	✓	swrst_s1[0]		
MIX	✓	✓	swrst_s1[1]		
GPIP	✓	✓	swrst_s1[2]	✓	
SEC_ENG	✓	✓	swrst_s1[4]	✓	
TZ	✓	✓		✓	
Efuse	✓	✓			
DMA	✓	✓	swrst_s1[12]	✓	
SDU/USB	✓	✓	swrst_s1[13]		
MM_MISC	✓	✓	swrst_s1_ext[1]	✓	
pSRAM_ctrl	✓	✓	swrst_s1_ext[2]	✓	
USB	✓	✓	swrst_s1_ext[3]	✓	
AUDIO DAC	✓	✓	swrst_s1_ext[5]	✓	
SDH	✓	✓	swrst_s1_ext[6]	✓	
EMAC	✓	✓	swrst_s1_ext[7]	✓	

Table 2.1: Software reset function table(continued)

BL616	CHIP_EN	Watch Dog/ PDS MISC/ Software Power On (swrst_cfg2[0])	Software Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS SOC	CPU Reset (swrst_cfg2[1])/ PDS NP
PDS	✓		swrst_s1[14]		
HBN	✓				
AON	✓				
UART0	✓	✓	swrst_s1a[0]	✓	
UART1	✓	✓	swrst_s1a[1]	✓	
SPI	✓	✓	swrst_s1a[2]	✓	
I2C0	✓	✓	swrst_s1a[3]	✓	
PWM	✓	✓	swrst_s1a[4]	✓	
TIMER	✓	✓	swrst_s1a[5]	✓	
IRR	✓	✓	swrst_s1a[6]	✓	
CKS	✓	✓	swrst_s1a[7]	✓	
DBI	✓	✓	swrst_s1a[8]	✓	
I2C1	✓	✓	swrst_s1a[9]	✓	
I2S	✓	✓	swrst_s1a[11]	✓	
AUDIO ADC	✓	✓	swrst_s1a[12]	✓	
Wi-Fi	✓	✓	swrst_s2[0]		
BLE	✓	✓	swrst_s3[0][2]		

2.3 Clock Source

Types:

- XTAL: external crystal oscillator clock, with optional frequencies of 24, 26, 32, 38.4, and 40 MHz depending on system requirements
- XTAL32K: external crystal oscillator clock, with frequency of 32768 Hz
- RC32K: RC oscillator clock with frequency of 32768 Hz and calibration
- RC32M: RC oscillator clock with frequency of 32 MHz and calibration
- PLL: multiple PLL modules, which can generate several clocks with different frequencies to meet various application scenarios

The clock control unit distributes the clocks from the oscillator to the core and peripheral devices. You can choose the system clock source, dynamic frequency divider, and clock configuration, and use the 32 kHz clock in sleep to achieve low-power clock management.

Peripheral clocks include Flash, UART, I2C, SPI, PWM, IR-remote, GPADC, and GPDAC.

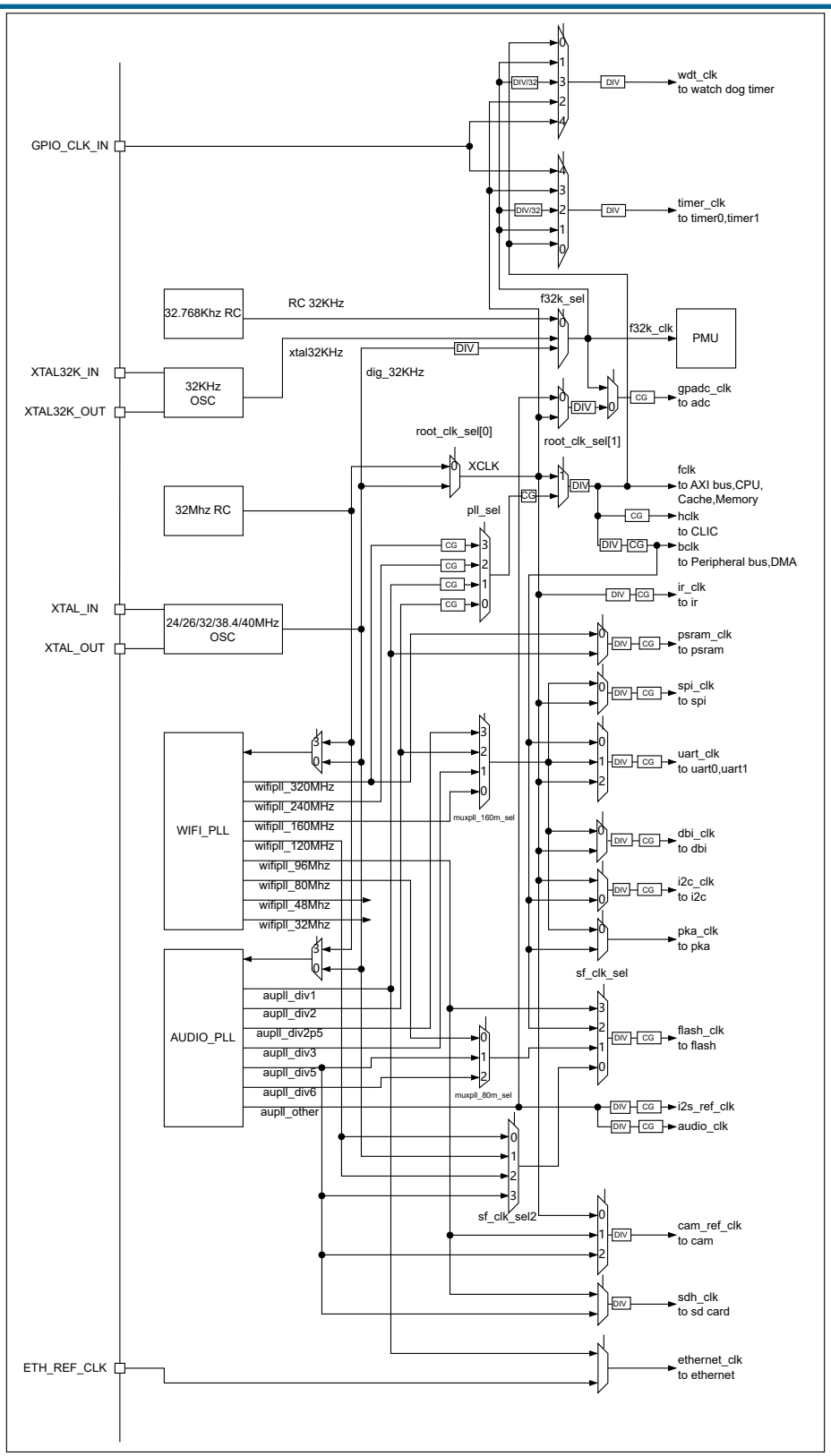


Fig. 2.1: Clock Architecture

3 GLB

3.1 Overview

Global Register (GLB), a general global setting module for chips, mainly includes clock, reset, bus, memory, power, and GPIO management.

3.2 Functional Description

3.2.1 Clock Management

It is mainly used to set the clocks of processors, buses, and peripherals, set the clock source and frequency division of the above modules, and achieve the clock gating of these modules, to save power for the system. For more details, see system clock sections.

3.2.2 Reset Management

It provides the separate reset function of each peripheral module and the chip reset function.

See *Reset Management* for details.

3.2.3 Bus Management

It provides bus arbitration and bus error settings, so that users can set whether to interrupt and provide the error bus address information when a bus error occurs, facilitating program debugging by users.

3.2.4 Memory Management

It manages the size of each memory area:

- EM management: A part of space of WRAM serves as EM, 32 KB by default. This function can assign the space of a specified size to EM, while the remaining one serves as WRAM.

It provides power management of each memory module in the low-power mode of the chip system, including two setting modes:

- Retention mode: The data in the memory can be saved, but it cannot be read or written before exiting the low-power mode
- Sleep mode: This mode is only used to reduce system power consumption, because it will cause memory data to lose

3.2.5 LDO Management

It provides on-chip LDO management:

- LDO management: The output voltage of the on-chip LDO can be adjusted to reduce power consumption.

4 GPIO

4.1 Overview

Users can connect General Purpose I/O Ports (GPIO) with external hardware devices to control these devices.

4.2 Features

- Up to 35 I/O pins
- Each I/O pin supports up to 25 functions
- Each I/O pin can be configured in pull-up, pull-down, or floating mode
- Each I/O pin can be configured as input, output or Hi-Z state mode
- The output mode of each I/O pin has 4 optional drive capabilities
- The input mode of each I/O pin can be set to enable/disable the Schmitt trigger
- Each I/O pin supports 9 external interrupt modes
- FIFO depth: 128 * 16-bit
- Data can be transferred through DMA from RAM to I/O pins for output

4.3 GPIO function list

The reg_gpio_xx_func_sel bit of the GPIO_CFGxx (where xx represents the GPIO pin number) register is used to set the multiplexing function of GPIO. The multiplexing function number is shown in the following table:

Table 4.1: GPIO function list

Number	Function
0	SDH

Table 4.1: GPIO function list (continued)

Number	Function
1	SPI0
2	FLASH
3	I2S0
4	PDM
5	I2C0
6	I2C1
7	UART0
8	EMAC
9	CAM
10	ANALOG
11	GPIO
12	SDIO
16	PWM0
17	JTAG
18	UART1
19	PWM1
20	SPI1
21	I2S1
22	DBI_B
23	DBI_C
24	QSPI
25	AUDAC
31	CLOCK_OUT

Note: If the GPIO is set as a peripheral function, just set the reg_gpio_xx_func_sel bit of the GPIO_CFGxx register to the corresponding number of the peripheral.

4.4 GPIO Input Settings

Set the register GPIO_CFGxx to configure the general interface to the input mode as described below (xx denotes the GPIO pin number):

- Set reg_gpio_xx_ie to 1 to enable the GPIO input mode
- Set reg_gpio_xx_func_sel to 11 to enter the SWGPIO mode
- In the SWGPIO mode, set to enable/disable the Schmitt trigger through reg_gpio_xx_smt for waveform shaping
- Set to enable/disable the internal pull-up and pull-down functions through reg_gpio_xx_pu and reg_gpio_xx_pd
- Set the type of external interrupt through reg_gpio_xx_int_mode_set , and then read the level value of I/O pin through reg_gpio_xx_i

4.5 GPIO Output Settings

The following four output modes of GPIO can be set through the register GPIO_CFGxx.

4.5.1 Normal Output Mode

- Set reg_gpio_xx_oe to 1 to enable the GPIO output mode
- Set reg_gpio_xx_func_sel to 11 to enter the SWGPIO mode
- Set reg_gpio_xx_mode to 0 to enable the normal output function of I/O
- Set to enable/disable the internal pull-up and pull-down functions through reg_gpio_xx_pu and reg_gpio_xx_pd , and then set the level of I/O pin through reg_gpio_xx_o

4.5.2 Set/Clear Output Mode

- Set reg_gpio_xx_oe to 1 to enable the GPIO output mode
- Set reg_gpio_xx_func_sel to 11 to enter the SWGPIO mode
- Set reg_gpio_xx_mode to 1 to enable the Set/Clear output function of I/O
- Set to enable/disable the internal pull-up and pull-down functions through reg_gpio_xx_pu and reg_gpio_xx_pd

In the Set/Clear output mode, you can set reg_gpio_xx_set to 1 to keep the I/O pin at the high level, or set reg_gpio_xx_clr to 1 to keep the I/O pin at the low level. If both reg_gpio_xx_set and reg_gpio_xx_clr are set to 1, the I/O pin is kept at the high level. If both of them are set to 0, the setting does not work.

4.5.3 Buffer Output Mode

- Set reg_gpio_xx_oe to 1 to enable the GPIO output mode
- Set reg_gpio_xx_func_sel to 11 to enter the SWGPIO mode
- Set reg_gpio_xx_mode to 2 to enable the buffer output function of I/O
- Set to enable/disable the internal pull-up and pull-down functions through reg_gpio_xx_pu and reg_gpio_xx_pd

In the buffer output mode, when the cr_gpio_tx_en bit in the register GPIO_CFG142 is set to 1, the data stored in the register GPIO_CFG144 is written into the corresponding I/O pins one by one in order, to set the level of pin. The size of the buffer area is 128 * 16 bits.

The level state output to the pin can be set freely. With XCLK as the clock source, the value set to cr_code_total_time in the register GPIO_CFG142 is one cycle:

Level state of logical ‘1’: For a high level set by cr_code1_high_time plus a low level set by cr_code_total_time cr_code1_high_time , when cr_invert_code1_high is 0, logical ‘1’ outputs high level first and then low level, and otherwise, low level first and then high level.

Level state of logical ‘0’: For a high level set by cr_code0_high_time plus a low level set by cr_code_total_time cr_code0_high_time , when cr_invert_code0_high is 0, logical ‘0’ outputs high level first and then low level, and otherwise, low level first and then high level.

Note: As the register of the buffer is 16-bit wide, every 16 pins form a group, and the pins with the lowest to highest serial numbers in a group are controlled by the corresponding bits in the buffer. The cr_gpio_dma_out_sel_latch bit in the register GPIO_CFG143 shall be set to 0. cr_gpio_dma_park_value is used to set the default level of I/O, namely 1 for high level and 0 for low level.

When cr_code_total_time = 10, cr_code0_high_time = 1, cr_code1_high_time = 5, cr_invert_code0_high = 0, cr_-invert_code1_high = 0, cr_gpio_dma_park_value =0, and cr_gpio_dma_out_sel_latch = 0, the waveform is shown as follows:

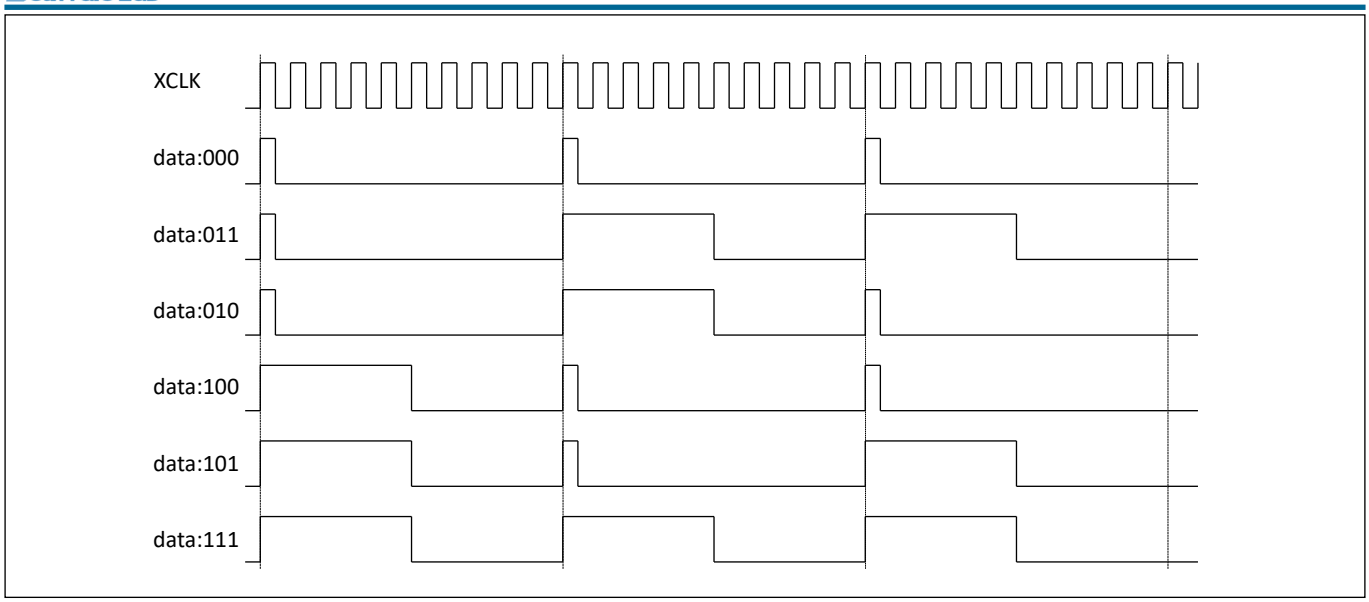


Fig. 4.1: General GPIO Output Waveform

When cr_code_total_time = 10, cr_code0_high_time = 1, cr_code1_high_time = 5, cr_invert_code0_high = 0, cr_-invert_code1_high = 1, cr_gpio_dma_park_value = 1, and cr_gpio_dma_out_sel_latch = 0, the waveform is shown as follows:

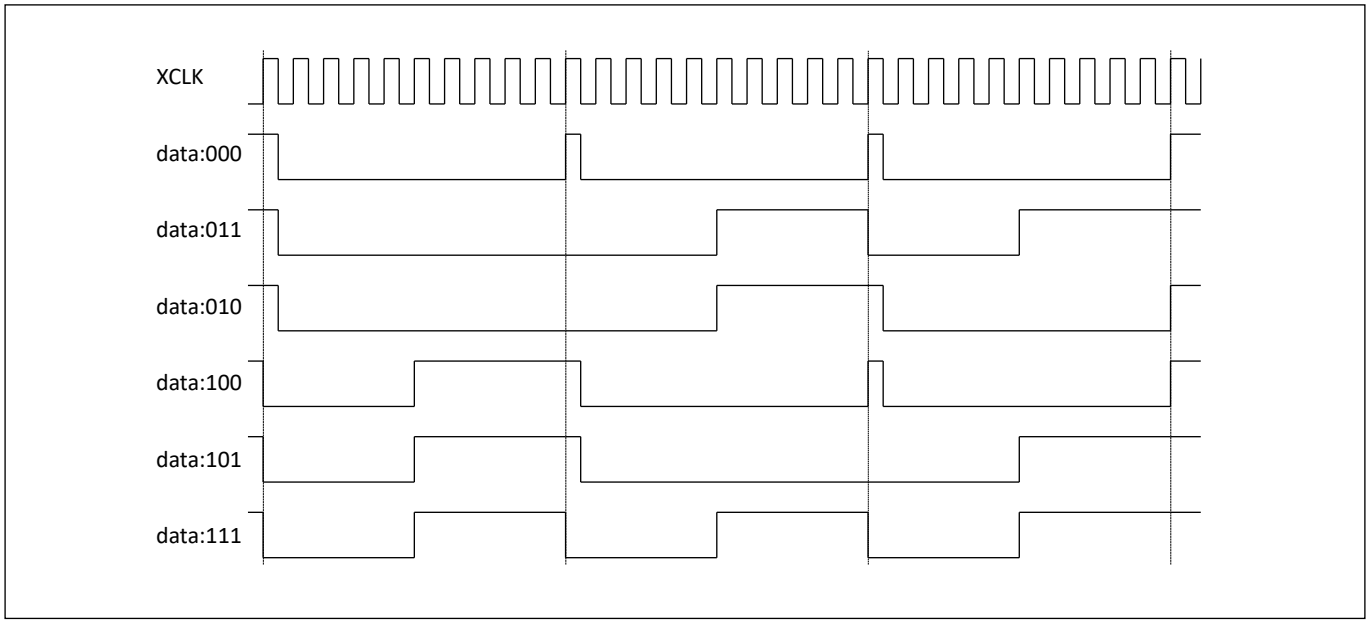


Fig. 4.2: Output Waveform of Inverted Level of Logical ‘1’ in Default High Level

When cr_code_total_time = 10, cr_code0_high_time = 1, cr_code1_high_time = 5, cr_invert_code0_high = 1, cr_-invert_code1_high = 0, cr_gpio_dma_park_value = 0, and cr_gpio_dma_out_sel_latch = 0, the waveform is shown as follows:

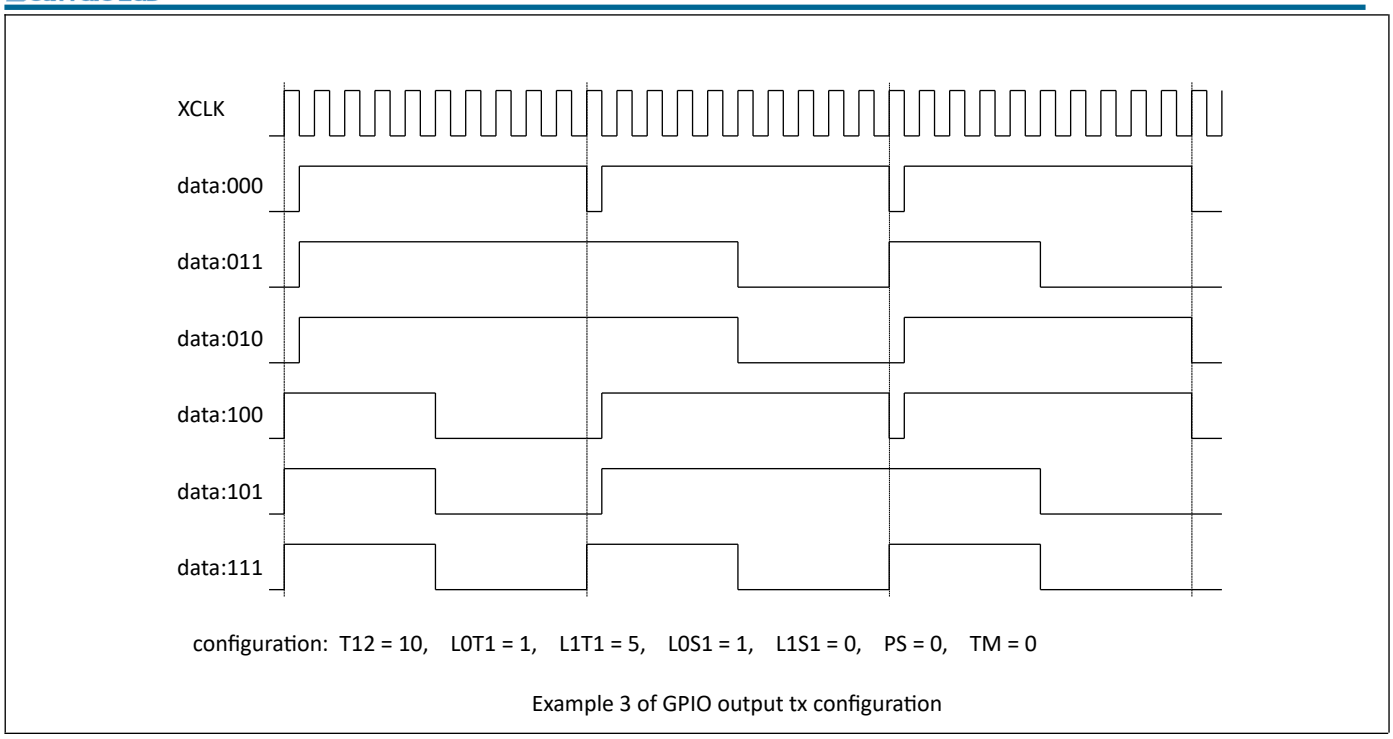


Fig. 4.3: Output Waveform of Inverted Level of Logical ‘0’ in Default Low Level

When cr_code_total_time = 10, cr_code0_high_time = 1, cr_code1_high_time = 5, cr_invert_code0_high = 1, cr_-invert_code1_high = 1, cr_gpio_dma_park_value = 1, and cr_gpio_dma_out_sel_latch = 0, the waveform is shown as follows:

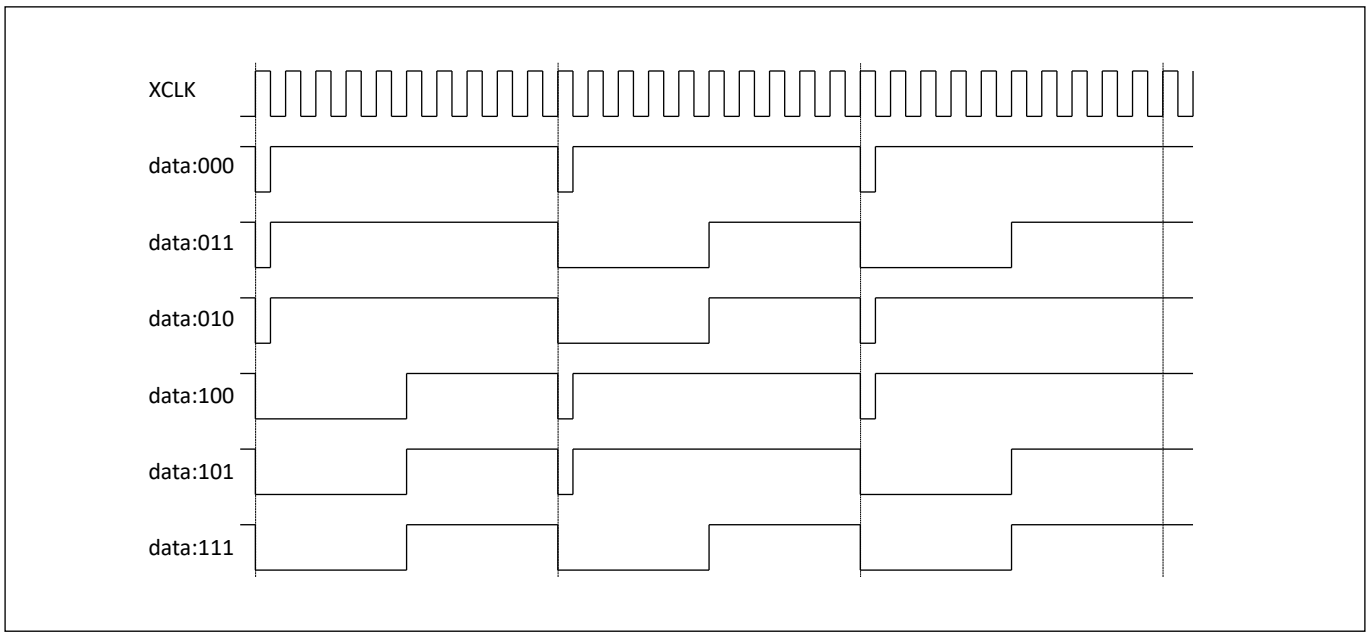


Fig. 4.4: Output Waveforms of Inverted Levels of Logical ‘0’ and Logical ‘1’ in Default High Level

4.5.4 Cache Set/Clear Output Mode

- Set reg_gpio_xx_oe to 1 to enable the GPIO output mode
- Set reg_gpio_xx_func_sel to 11 to enter the SWGPIO mode
- Set reg_gpio_xx_mode to 3 to enable the Cache Set/Clear output function of I/O
- Set to enable/disable the internal pull-up and pull-down functions through reg_gpio_xx_pu and reg_gpio_xx_pd

When the cr_gpio_tx_en bit in the register GPIO_CFG142 is set to 1, the data written into the FIFO by the GPIO_CFG144 register will be written to the corresponding pins one by one in order, to set the level of pin. The size of the buffer area is 128 * 16 bits.

The level state output to the pin can be set freely. With XCLK as the clock source, the value set to the cr_code_total_time bit in the register GPIO_CFG142 is one cycle:

Every 8 pins form a group. The low 8 bits and high 8 bits of the register GPIO_CFG144 set the output high/low level of 8 pins respectively. If the low 8 bits are written to 1, the corresponding pin outputs a high level. If the high 8 bits are written to 1, that outputs a low level. The bits written to 0 in this register shall be invalid. When the corresponding bits in the high 8 bits and low 8 bits are written to 1 simultaneously for the same pin, the pin outputs a high level.

The cr_gpio_dma_out_sel_latch bit in the register GPIO_CFG143 shall be set to 1. cr_gpio_dma_park_value is used to set the default level of I/O, namely 1 for high level and 0 for low level.

4.6 I/O FIFO

The depth of I/O FIFO is 128 * 16-bit. The gpio_tx_fifo_cnt bit in the register GPIO_CFG143 indicates the current available space of FIFO (128 by default). Every time a value is written into the GPIO_CFG144 register, the value of gpio_tx_fifo_cnt will decrease by 1. After it decreases to 0, if a value is continuously written to the register GPIO_CFG144 and cr_gpio_tx_fer_en is 1, the error interrupt will be enabled and this interrupt will occur.

When the cr_gpio_tx_en bit in the GPIO_CFG142 register is 1, the data of I/O FIFO will be sent to I/O pins one by one, and the value of gpio_tx_fifo_cnt will increment. When it is incremented to greater than cr_gpio_tx_fifo_th and cr_gpio_tx_fifo_en is 1, the FIFO interrupt is enabled and this interrupt will occur.

If the cr_gpio_dma_tx_en bit in the register CR_GPIO_CFG143 is 1, DMA is enabled to send data. If cr_gpio_tx_fifo_th is less than gpio_tx_fifo_cnt , DMA will transfer the data from the preset RAM to the buffer, whereas the interrupt flag r_gpio_tx_fifo_int will be cleared automatically.

4.7 I/O Interrupt

I/O supports various external interrupts. Setting the reg_gpio_xx_int_mask in the register GPIO_CFGxx to 0 can enable the external interrupt of the corresponding pin. reg_gpio_xx_int_mode_set is used to set the external interrupt type of that pin.

The supported interrupt types are as follows:

- Synchronous Falling Edge Interrupt
  - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If a high level is followed by two low levels, a synchronous falling edge interrupt will be generated at this time
- Synchronous Rising Edge Interrupt
  - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If a low level is followed by two high levels, a synchronous rising edge interrupt will be generated at this time
- Synchronous Low Level Interrupt
  - Based on the f32k_clk clock, after detecting a low level, a synchronous low-level interrupt is generated at the rising edge of the third clock
- Synchronous High Level Interrupt
  - Based on the f32k_clk clock, after detecting a high level, a synchronous high-level interrupt is generated at the rising edge of the third clock
- Synchronous Double Edge Interrupt
  - Based on the f32k_clk clock, if a high level transition to low level (low level transition to high level) is detected, a falling edge (rising edge) event will be generated. After the event is generated, at the third rising edge of the clock, synchronous double edge interrupt will be generated
- Asynchronous Falling Edge Interrupt
  - When a high-to-low transition is detected, an asynchronous falling edge interrupt is triggered immediately
- Asynchronous Rising Edge Interrupt
  - When a low-to-high transition is detected, an asynchronous rising edge interrupt is triggered immediately
- Asynchronous Low Level Interrupt
  - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If it is low for 3 consecutive times, an asynchronous low-level interrupt is triggered
- Asynchronous High Level Interrupt
  - Based on the f32k_clk clock, the input pin level is sampled once on each rising edge of the clock. If it is high for 3 consecutive times, an asynchronous high-level interrupt will be triggered

In the interrupt function, you can obtain the interrupt-generating GPIO state through the gpio_xx_int_stat of the register GPIO_CFGxx, and clear the interrupt flag through reg_gpio_xx_int_clr .

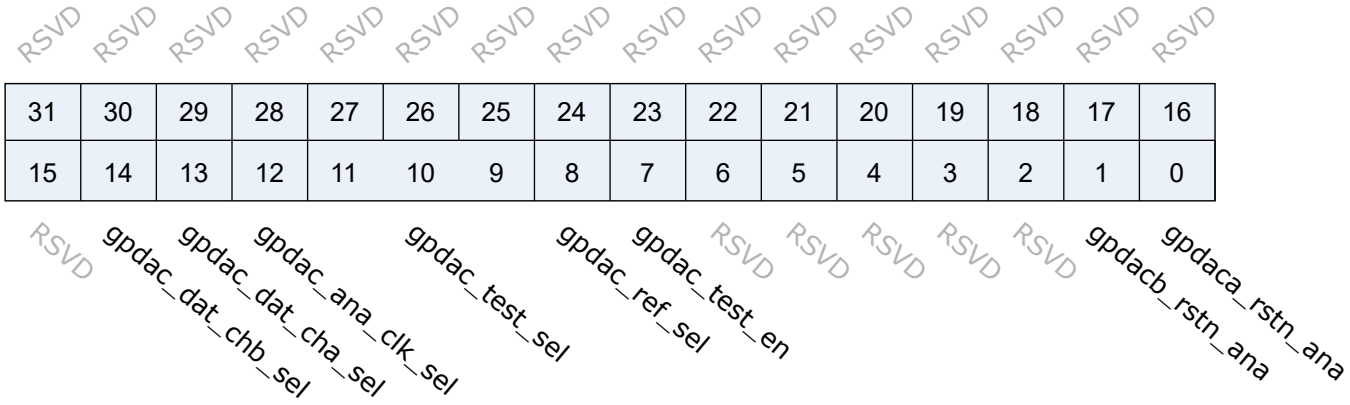
4.8 Register description

Name	Description
dac_cfg0	DAC source control register
dac_cfg1	DAC CHA control register
dac_cfg2	DAC CHB control register
dac_cfg3	DAC converter data register
gpio_cfg0	GPIO0 config register
gpio_cfg1	GPIO1 config register
gpio_cfg2	GPIO2 config register
gpio_cfg3	GPIO3 config register
gpio_cfg4	GPIO4 config register
gpio_cfg5	GPIO5 config register
gpio_cfg6	GPIO6 config register
gpio_cfg7	GPIO7 config register
gpio_cfg8	GPIO8 config register
gpio_cfg9	GPIO9 config register
gpio_cfg10	GPIO10 config register
gpio_cfg11	GPIO11 config register
gpio_cfg12	GPIO12 config register
gpio_cfg13	GPIO13 config register
gpio_cfg14	GPIO14 config register
gpio_cfg15	GPIO15 config register
gpio_cfg16	GPIO16 config register
gpio_cfg17	GPIO17 config register
gpio_cfg18	GPIO18 config register
gpio_cfg19	GPIO19 config register
gpio_cfg20	GPIO20 config register

Name	Description
gpio_cfg21	GPIO21 config register
gpio_cfg22	GPIO22 config register
gpio_cfg23	GPIO23 config register
gpio_cfg24	GPIO24 config register
gpio_cfg25	GPIO25 config register
gpio_cfg26	GPIO26 config register
gpio_cfg27	GPIO27 config register
gpio_cfg28	GPIO28 config register
gpio_cfg29	GPIO29 config register
gpio_cfg30	GPIO30 config register
gpio_cfg31	GPIO31 config register
gpio_cfg32	GPIO32 config register
gpio_cfg33	GPIO33 config register
gpio_cfg34	GPIO34 config register
gpio_cfg128	GPIO input value register0
gpio_cfg129	GPIO input value register1
gpio_cfg136	GPIO output value register0
gpio_cfg137	GPIO output value register1
gpio_cfg138	GPIO set register0
gpio_cfg139	GPIO set register1
gpio_cfg141	GPIO clear register1
gpio_cfg142	GPIO TX config register0
gpio_cfg143	GPIO TX config register1
gpio_cfg144	GPIO TX fifo register

4.8.1 dac_cfg0

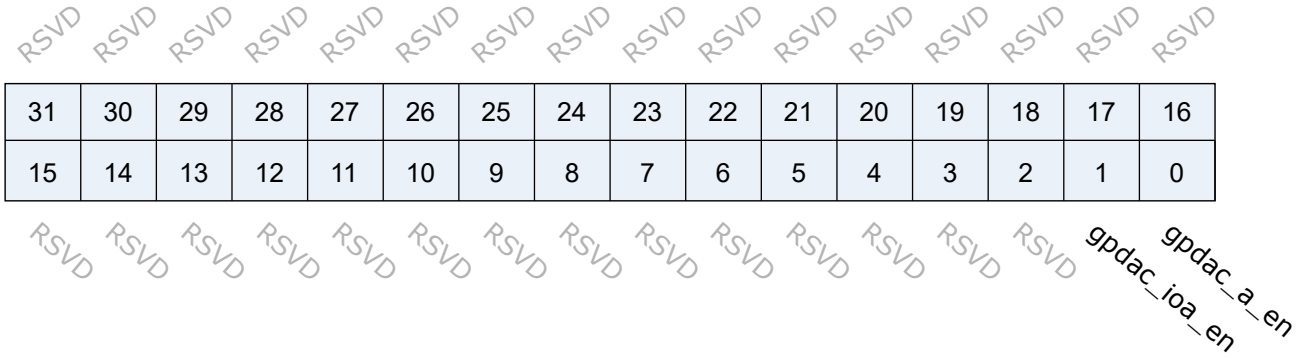
Address：0x20000120



Bits	Name	Type	Reset	Description
31:15	RSVD			
14	gpdac_dat_chb_sel	r/w	1'h0	0:data from gpip, 1:data from audio pwm
13	gpdac_dat_cha_sel	r/w	1'h0	0:data from gpip, 1:data from audio pwm
12	gpdac_ana_clk_sel	r/w	1'h0	0:clock from gpip, 1:clock from audio pwm
11:9	gpdac_test_sel	r/w	3'h0	select test point 0 7
8	gpdac_ref_sel	r/w	1'h0	Reference select 1'h0 Internal reference 1'h1 External reference
7	gpdac_test_en	r/w	1'h0	Test enable 1'h0 analog test disabled (ATEST is set in Hi-Z state) 1'h1 analog test point enabled to ATEST
6:2	RSVD			
1	gpdacb_rstn_ana	r/w	1'h1	Soft reset for DAC channel B, active low
0	gpdaca_rstn_ana	r/w	1'h1	Soft reset for DAC channel A, active low

4.8.2 dac_cfg1

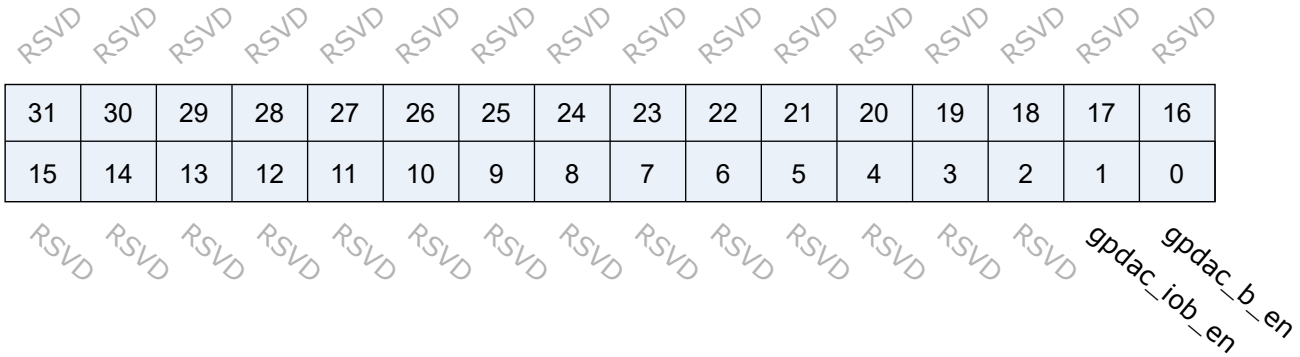
Address：0x20000124



Bits	Name	Type	Reset	Description
31:20	RSVD			
19:18	gpdac_a_rng	r/w	2'h3	Output voltage range control with internal/external reference
17:2	RSVD			
1	gpdac_ioa_en	r/w	1'h0	Channel A conversion output to pad enable 1'h0 Disable channel A conversion result to GPIO 1'h1 Enable channel A conversion result to GPIO
0	gpdac_a_en	r/w	1'h0	Channel A enable/disable signal 1'h0 Disable channel A conversion. 1'h1 Enable channel A conversion

4.8.3 dac_cfg2

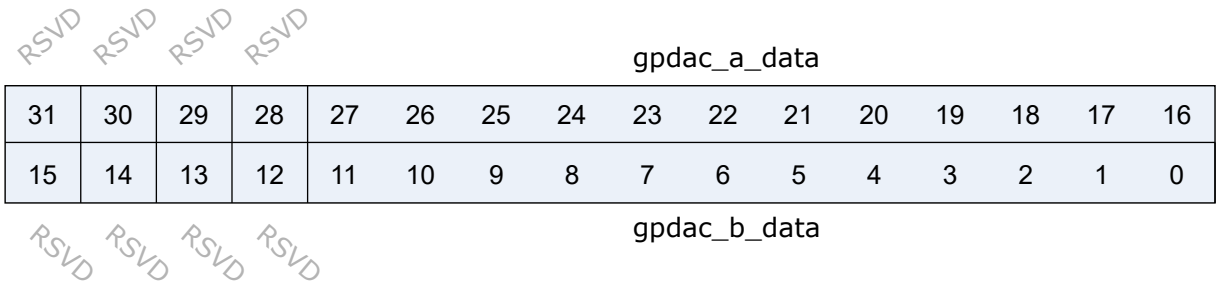
Address：0x20000128



Bits	Name	Type	Reset	Description
31:20	RSVD			
19:18	gpdac_b_rng	r/w	2'h3	Output voltage range control with internal/external reference
17:2	RSVD			
1	gpdac_iob_en	r/w	1'h0	channel B conversion output to pad enable 1'h0 Disable channel B conversion result to GPIO 1'h1 Enable channel B conversion result to GPIO
0	gpdac_b_en	r/w	1'h0	channel B enable/disable signal 1'h0 Disable channel B conversion. 1'h1 Enable channel B conversion

4.8.4 dac_cfg3

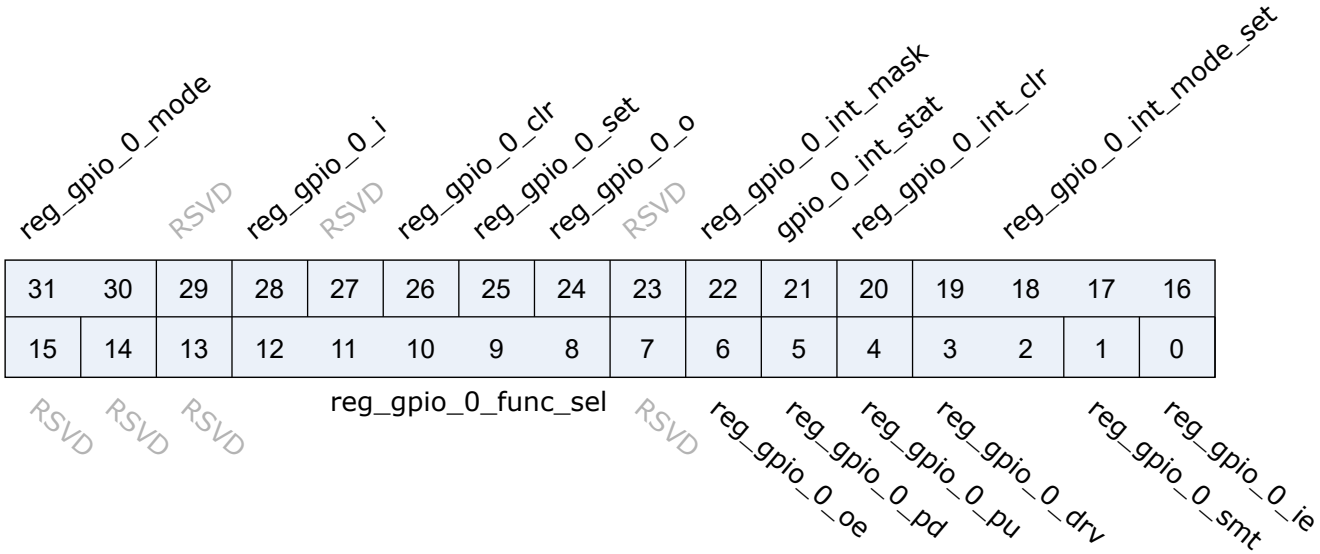
Address：0x2000012c



Bits	Name	Type	Reset	Description
31:28	RSVD			
27:16	gpdac_a_data	r/w	12'h0	Channel A Data input
15:12	RSVD			
11:0	gpdac_b_data	r/w	12'h0	Channel B Data input
-1:21	RSVD			
20	wifipll_en_rf_div3_hw	r	1'h1	
19:0	RSVD			

4.8.5 gpio_cfg0

Address：0x200008c4

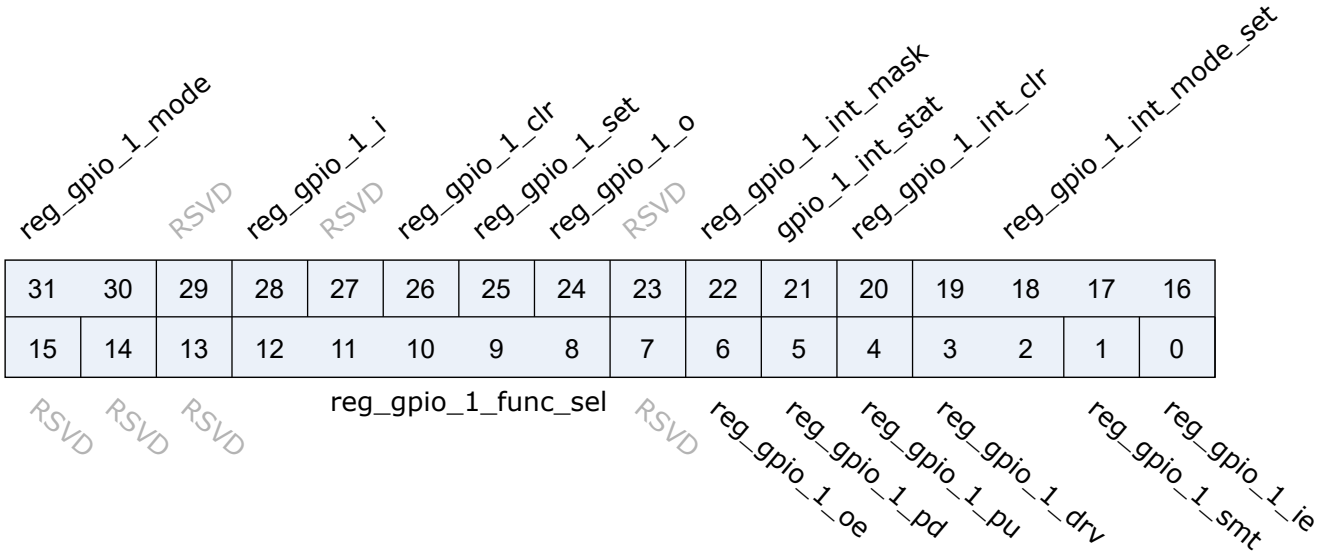


Bits	Name	Type	Reset	Description
31:30	reg_gpio_0_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_0_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_0_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_0_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_0_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_0_int_mask	r/w	1	mask interrupt (1)
21	gpio_0_int_stat	r	0	interrupt status
20	reg_gpio_0_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_0_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_0_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_0_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_0_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_0_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_0_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_0_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_0_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.6 gpio_cfg1

Address：0x200008c8

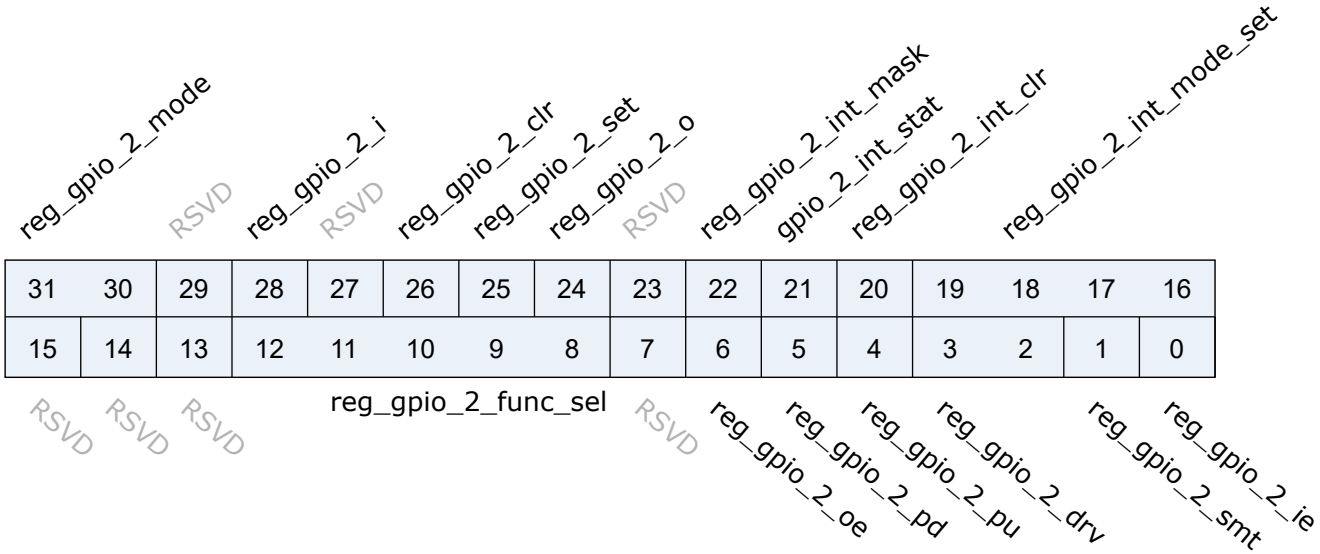


Bits	Name	Type	Reset	Description
31:30	reg_gpio_1_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_1_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_1_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_1_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_1_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_1_int_mask	r/w	1	mask interrupt (1)
21	gpio_1_int_stat	r	0	interrupt status
20	reg_gpio_1_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_1_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_1_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_1_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_1_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_1_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_1_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_1_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_1_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.7 gpio_cfg2

Address：0x200008cc

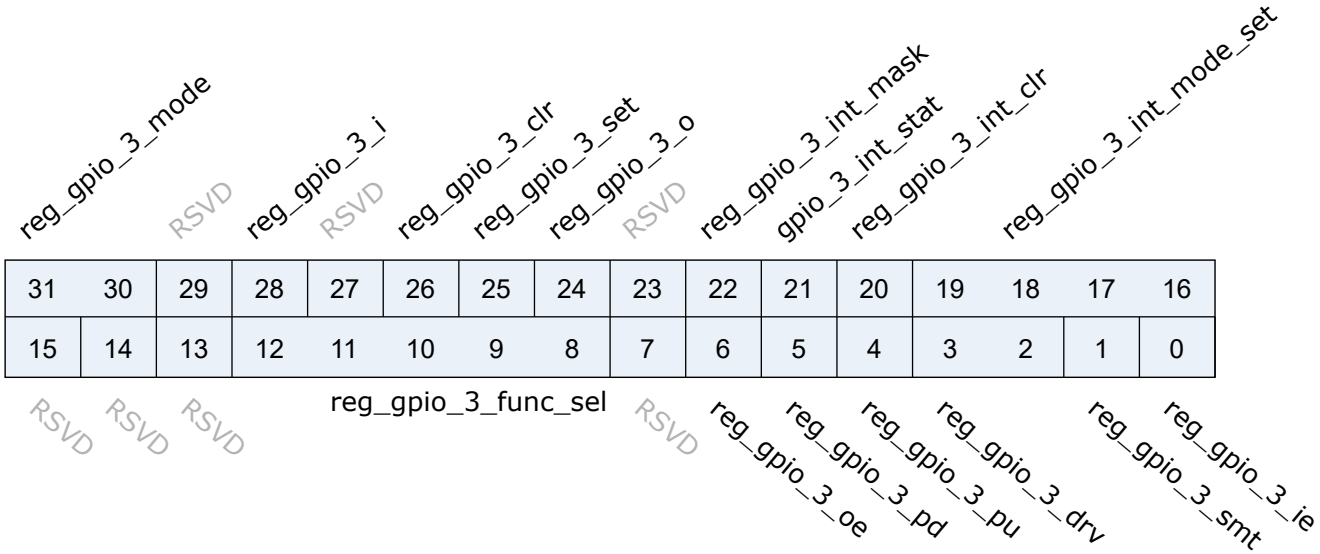


Bits	Name	Type	Reset	Description
31:30	reg_gpio_2_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_2_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_2_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_2_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_2_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_2_int_mask	r/w	1	mask interrupt (1)
21	gpio_2_int_stat	r	0	interrupt status
20	reg_gpio_2_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_2_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_2_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_2_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_2_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_2_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_2_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_2_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_2_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.8 gpio_cfg3

Address：0x200008d0

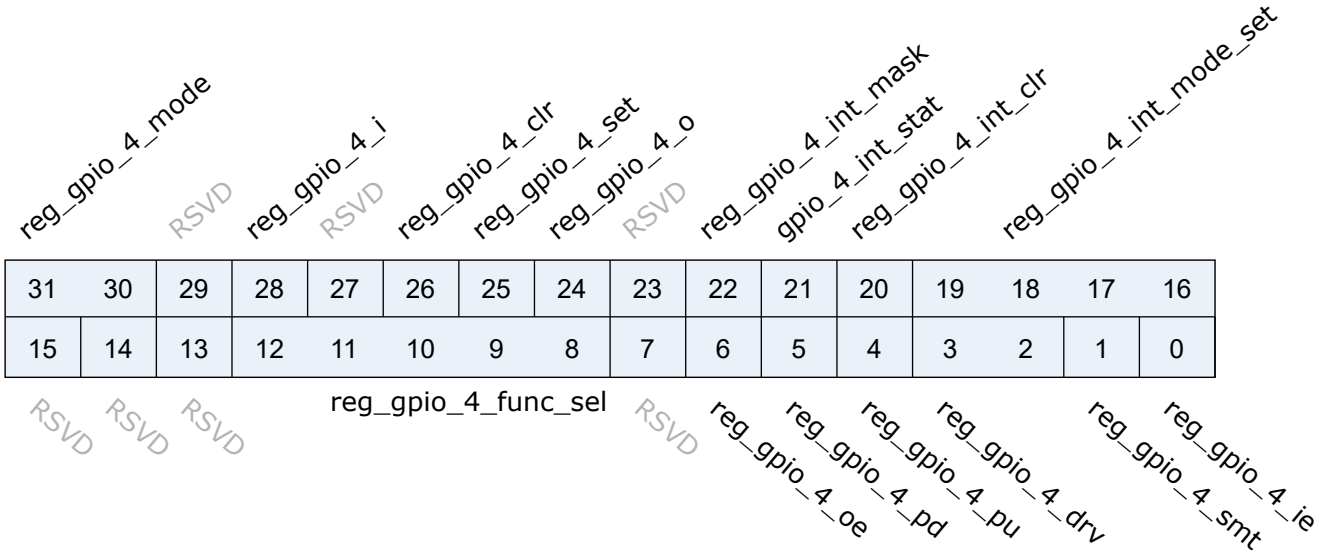


Bits	Name	Type	Reset	Description
31:30	reg_gpio_3_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_3_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_3_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_3_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_3_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_3_int_mask	r/w	1	mask interrupt (1)
21	gpio_3_int_stat	r	0	interrupt status
20	reg_gpio_3_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_3_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_3_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_3_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_3_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_3_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_3_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_3_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_3_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.9 gpio_cfg4

Address：0x200008d4

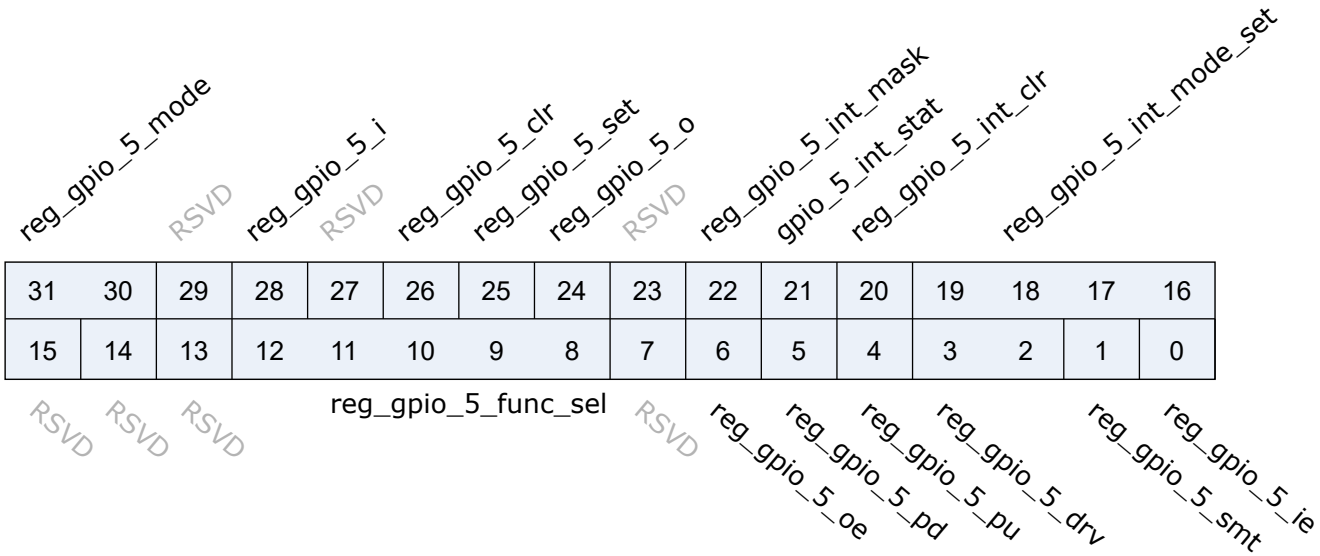


Bits	Name	Type	Reset	Description
31:30	reg_gpio_4_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_4_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_4_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_4_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_4_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_4_int_mask	r/w	1	mask interrupt (1)
21	gpio_4_int_stat	r	0	interrupt status
20	reg_gpio_4_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_4_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_4_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_4_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_4_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_4_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_4_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_4_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_4_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.10 gpio_cfg5

Address：0x200008d8

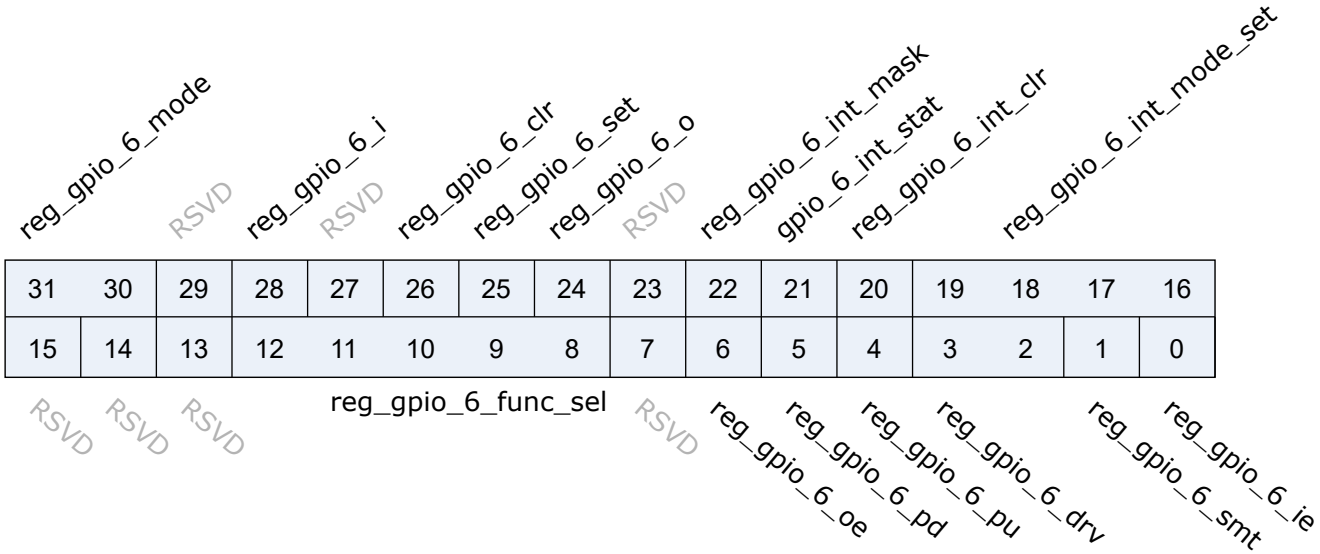


Bits	Name	Type	Reset	Description
31:30	reg_gpio_5_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_5_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_5_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_5_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_5_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_5_int_mask	r/w	1	mask interrupt (1)
21	gpio_5_int_stat	r	0	interrupt status
20	reg_gpio_5_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_5_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_5_func_sel	r/w	5'hB	GPIO Function Select (Default : SWGPIO)
7	RSVD			
6	reg_gpio_5_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_5_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_5_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_5_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_5_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_5_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.11 gpio_cfg6

Address：0x200008dc

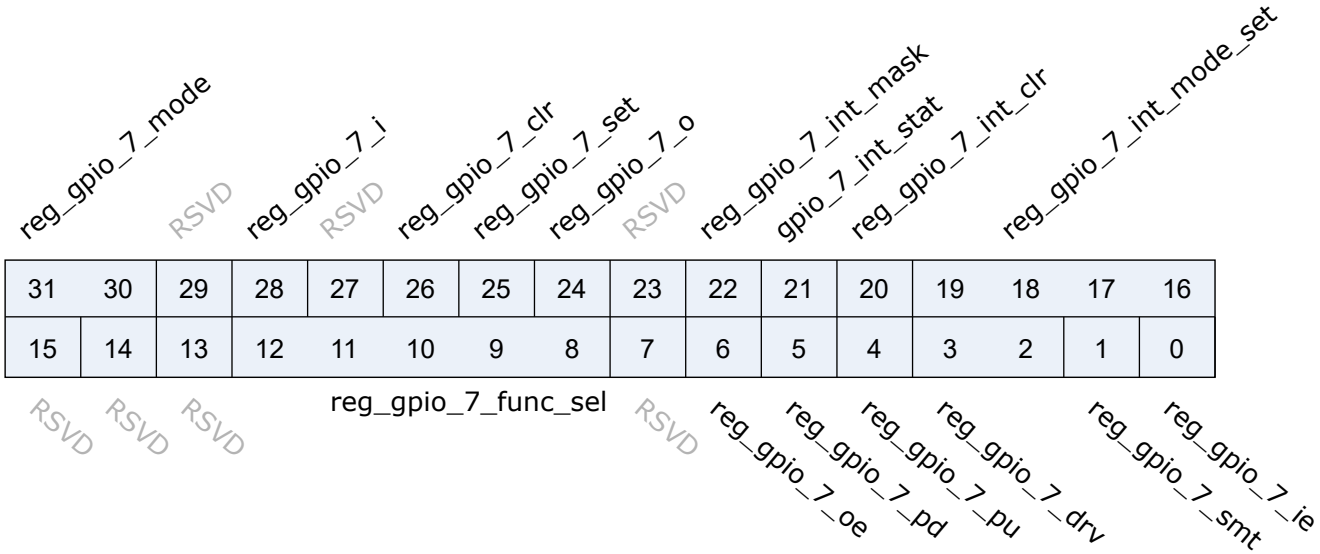


Bits	Name	Type	Reset	Description
31:30	reg_gpio_6_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_6_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_6_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_6_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_6_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_6_int_mask	r/w	1	mask interrupt (1)
21	gpio_6_int_stat	r	0	interrupt status
20	reg_gpio_6_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_6_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_6_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_6_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_6_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_6_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_6_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_6_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_6_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.12 gpio_cfg7

Address：0x200008e0

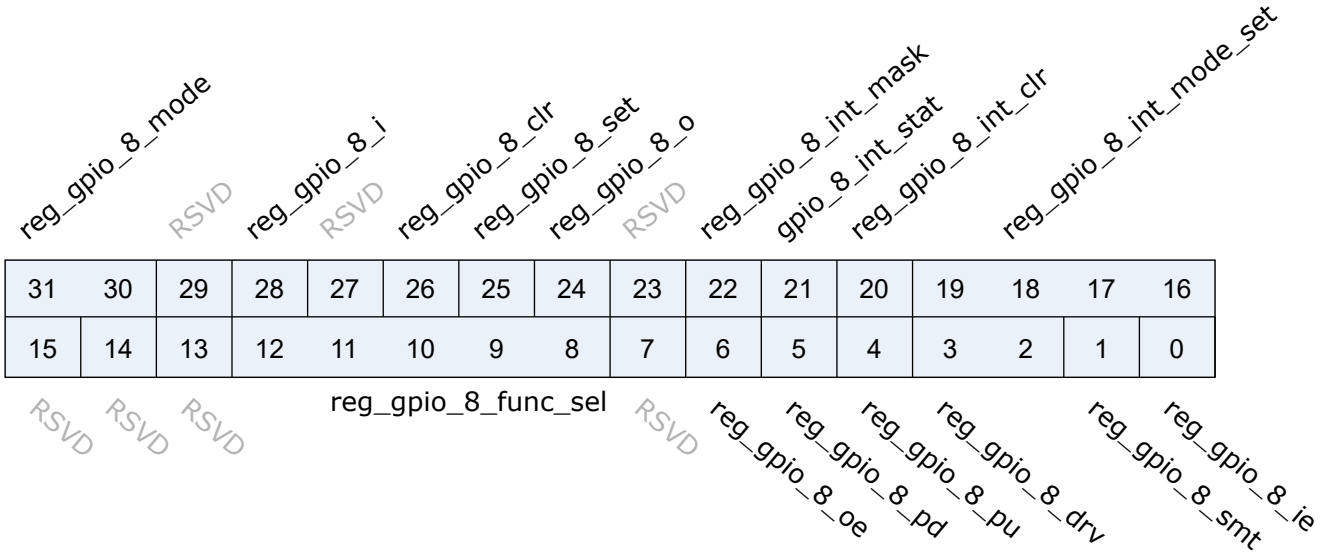


Bits	Name	Type	Reset	Description
31:30	reg_gpio_7_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_7_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_7_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_7_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_7_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_7_int_mask	r/w	1	mask interrupt (1)
21	gpio_7_int_stat	r	0	interrupt status
20	reg_gpio_7_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_7_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_7_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_7_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_7_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_7_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_7_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_7_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_7_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.13 gpio_cfg8

Address：0x200008e4

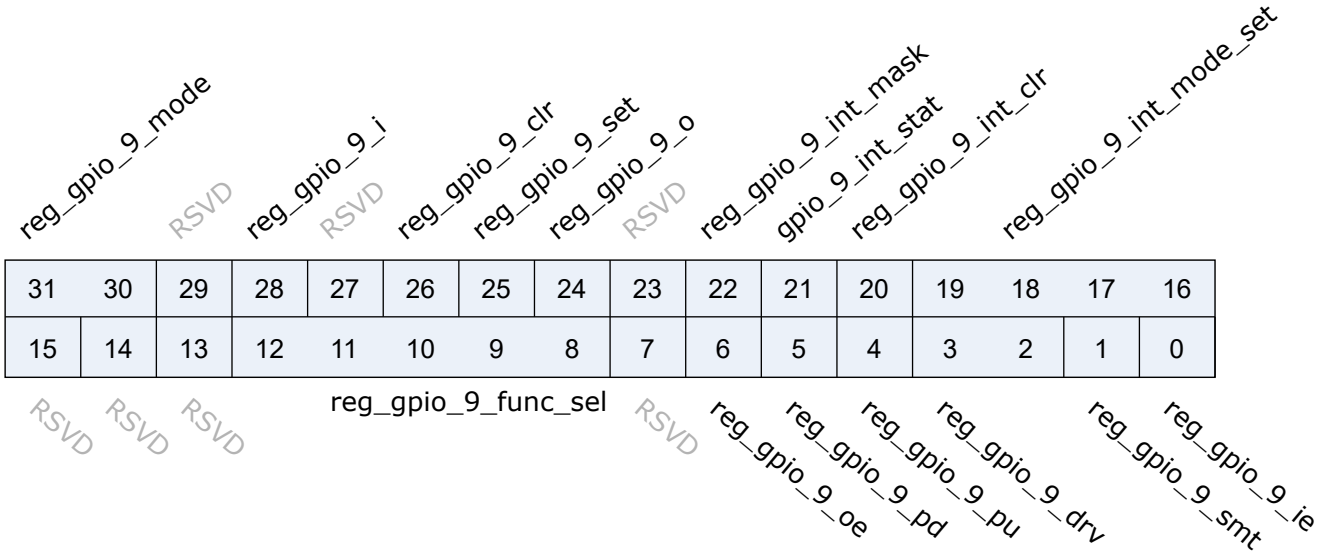


Bits	Name	Type	Reset	Description
31:30	reg_gpio_8_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_8_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_8_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_8_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_8_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_8_int_mask	r/w	1	mask interrupt (1)
21	gpio_8_int_stat	r	0	interrupt status
20	reg_gpio_8_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_8_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_8_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_8_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_8_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_8_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_8_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_8_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_8_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.14 gpio_cfg9

Address：0x200008e8

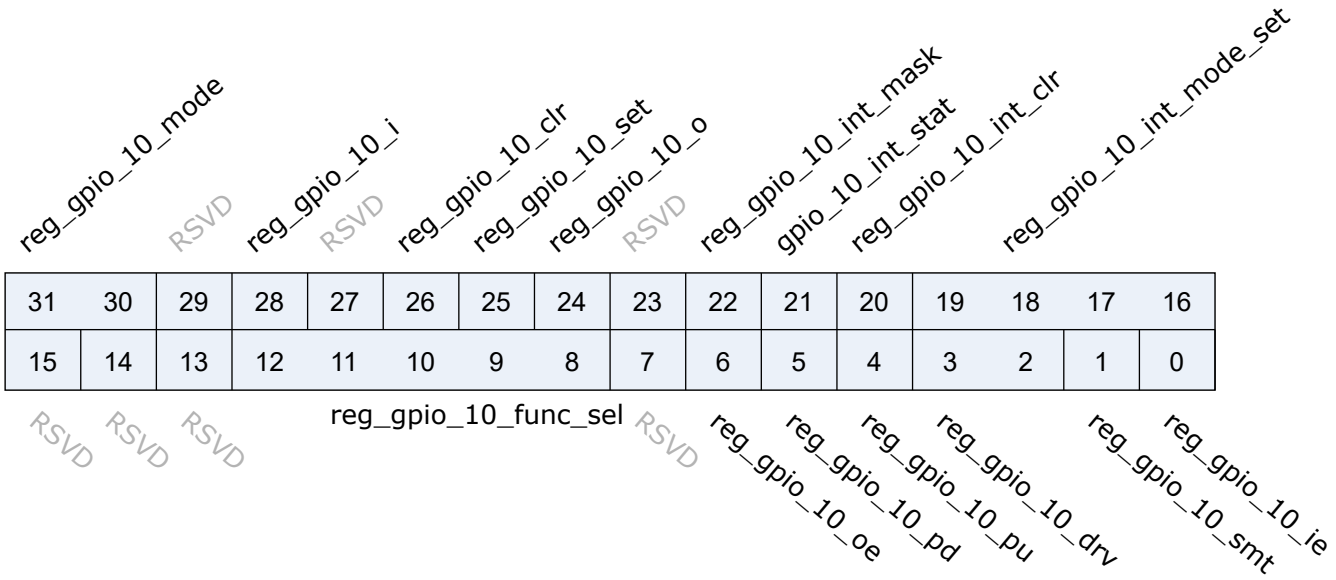


Bits	Name	Type	Reset	Description
31:30	reg_gpio_9_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_9_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_9_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_9_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_9_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1
23	RSVD			

Bits	Name	Type	Reset	Description
22	reg_gpio_9_int_mask	r/w	1	mask interrupt (1)
21	gpio_9_int_stat	r	0	interrupt status
20	reg_gpio_9_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_9_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_9_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_9_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_9_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_9_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_9_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_9_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_9_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.15 gpio_cfg10

Address：0x200008ec

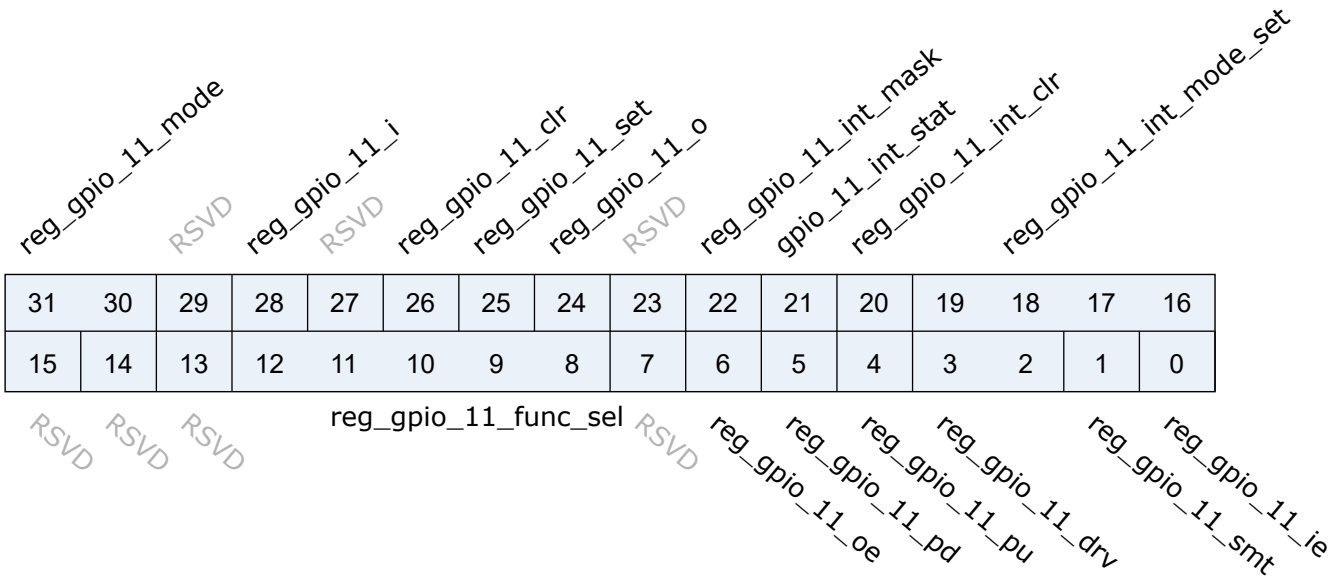


Bits	Name	Type	Reset	Description
31:30	reg_gpio_10_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_10_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_10_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_10_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_10_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_10_int_mask	r/w	1	mask interrupt (1)
21	gpio_10_int_stat	r	0	interrupt status
20	reg_gpio_10_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_10_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_10_func_sel	r/w	5'hF	GPIO Function Select (Default : CCI)
7	RSVD			
6	reg_gpio_10_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_10_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_10_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_10_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_10_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_10_ie	r/w	1	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.16 gpio_cfg11

Address：0x200008f0

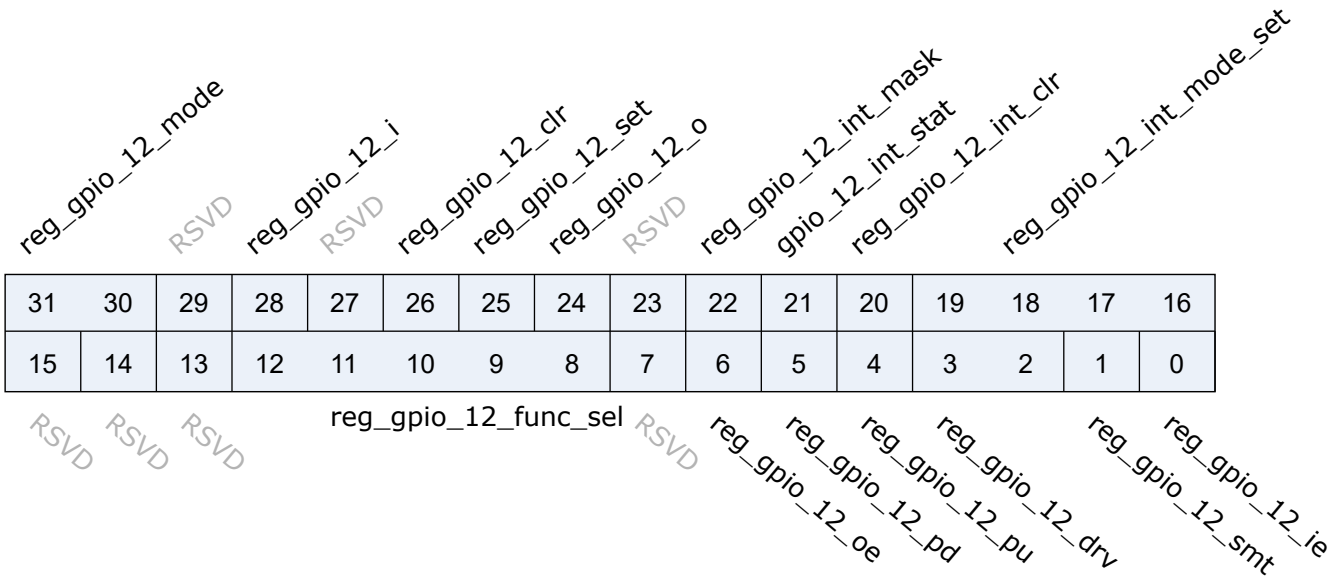


Bits	Name	Type	Reset	Description
31:30	reg_gpio_11_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_11_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_11_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_11_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_11_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_11_int_mask	r/w	1	mask interrupt (1)
21	gpio_11_int_stat	r	0	interrupt status
20	reg_gpio_11_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_11_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_11_func_sel	r/w	5'hF	GPIO Function Select (Default : CCI)
7	RSVD			
6	reg_gpio_11_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_11_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_11_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_11_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_11_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_11_ie	r/w	1	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.17 gpio_cfg12

Address：0x200008f4

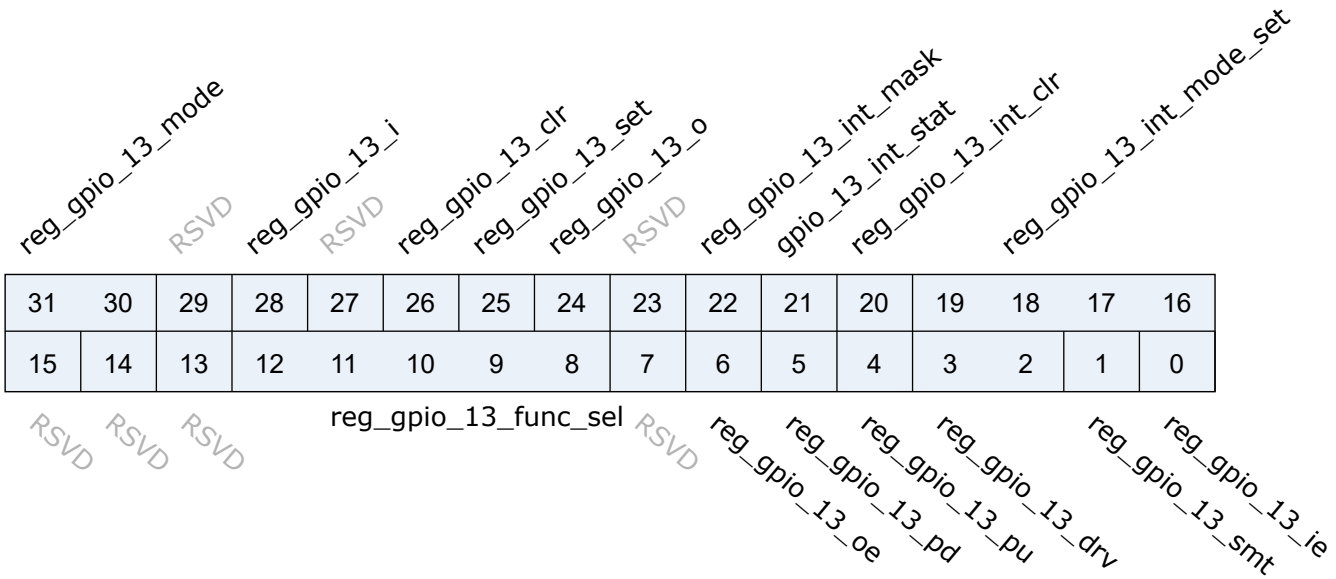


Bits	Name	Type	Reset	Description
31:30	reg_gpio_12_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_12_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_12_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_12_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_12_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_12_int_mask	r/w	1	mask interrupt (1)
21	gpio_12_int_stat	r	0	interrupt status
20	reg_gpio_12_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_12_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_12_func_sel	r/w	5'hF	GPIO Function Select (Default : CCI)
7	RSVD			
6	reg_gpio_12_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_12_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_12_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_12_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_12_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_12_ie	r/w	1	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.18 gpio_cfg13

Address：0x200008f8

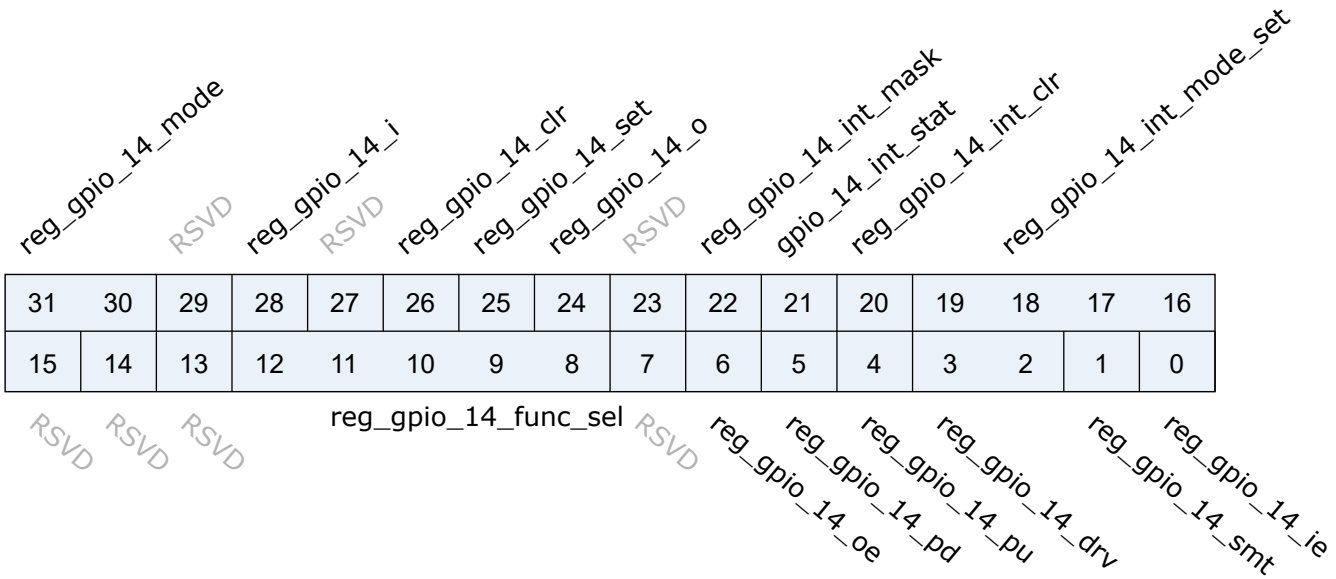


Bits	Name	Type	Reset	Description
31:30	reg_gpio_13_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_13_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_13_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_13_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_13_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_13_int_mask	r/w	1	mask interrupt (1)
21	gpio_13_int_stat	r	0	interrupt status
20	reg_gpio_13_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_13_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_13_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_13_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_13_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_13_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_13_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_13_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_13_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.19 gpio_cfg14

Address：0x200008fc

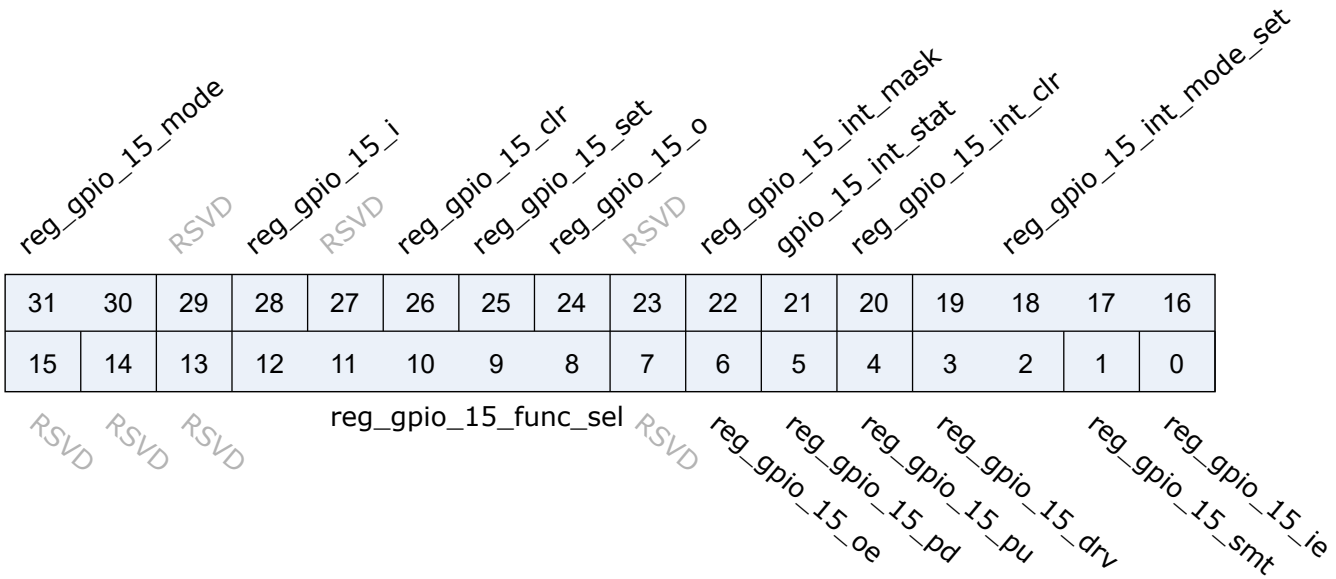


Bits	Name	Type	Reset	Description
31:30	reg_gpio_14_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_14_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_14_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_14_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_14_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_14_int_mask	r/w	1	mask interrupt (1)
21	gpio_14_int_stat	r	0	interrupt status
20	reg_gpio_14_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_14_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_14_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_14_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_14_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_14_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_14_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_14_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_14_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.20 gpio_cfg15

Address：0x20000900

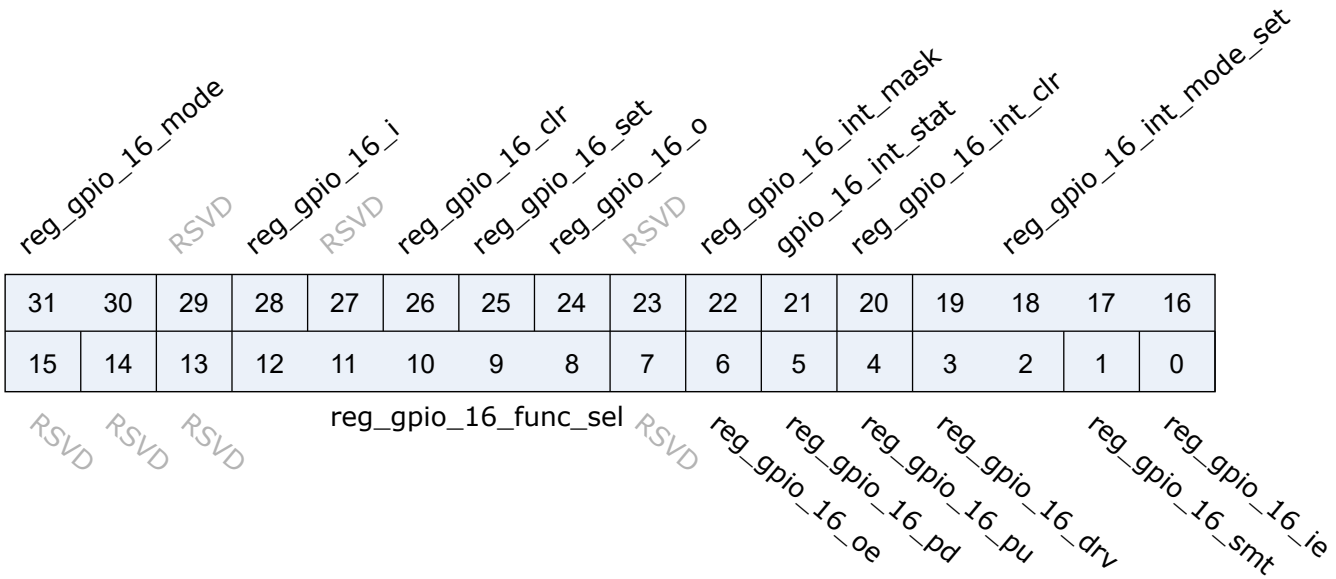


Bits	Name	Type	Reset	Description
31:30	reg_gpio_15_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_15_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_15_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_15_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_15_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_15_int_mask	r/w	1	mask interrupt (1)
21	gpio_15_int_stat	r	0	interrupt status
20	reg_gpio_15_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_15_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_15_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_15_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_15_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_15_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_15_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_15_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_15_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.21 gpio_cfg16

Address：0x20000904

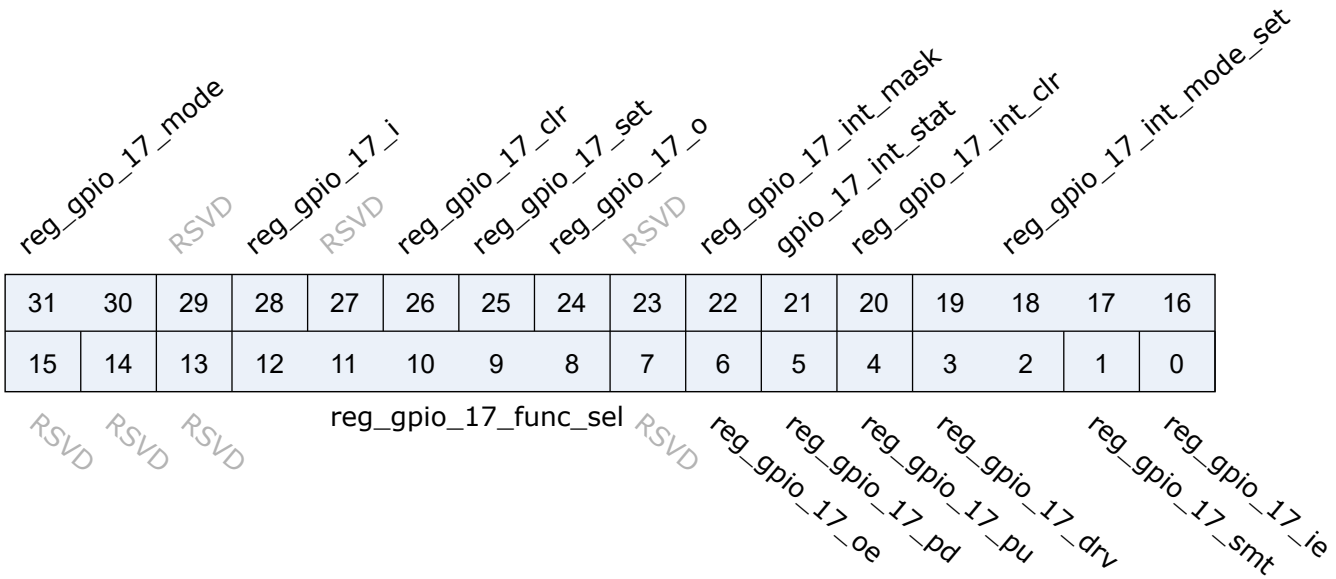


Bits	Name	Type	Reset	Description
31:30	reg_gpio_16_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_16_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_16_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_16_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_16_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_16_int_mask	r/w	1	mask interrupt (1)
21	gpio_16_int_stat	r	0	interrupt status
20	reg_gpio_16_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_16_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_16_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_16_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_16_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_16_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_16_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_16_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_16_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.22 gpio_cfg17

Address：0x20000908

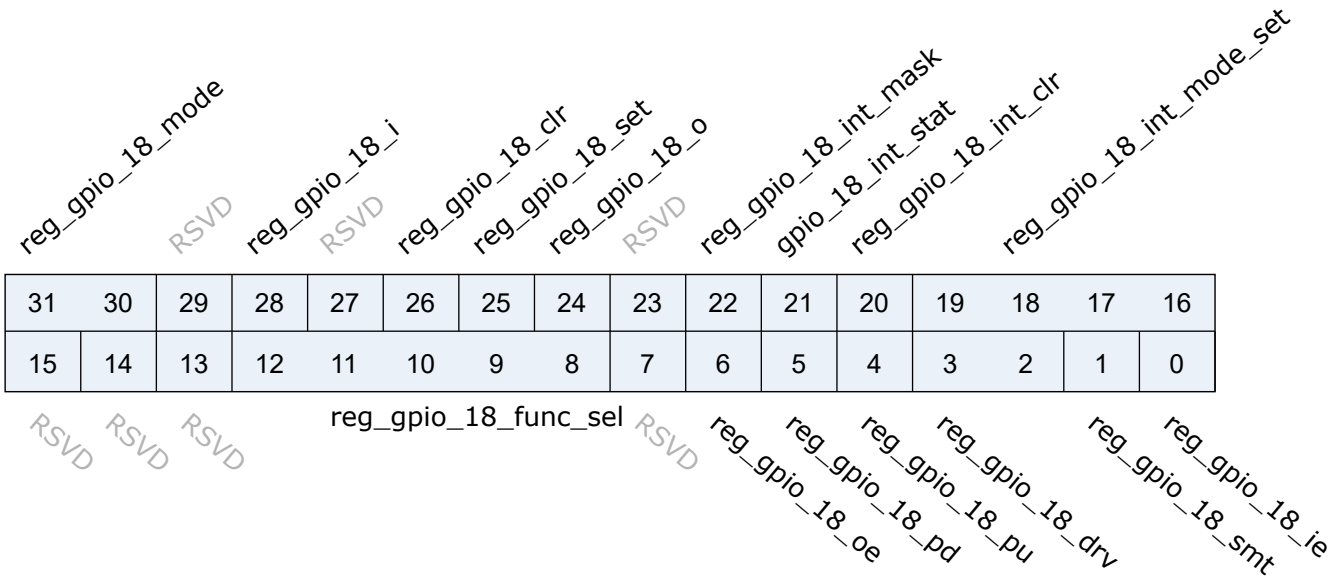


Bits	Name	Type	Reset	Description
31:30	reg_gpio_17_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_17_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_17_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_17_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_17_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_17_int_mask	r/w	1	mask interrupt (1)
21	gpio_17_int_stat	r	0	interrupt status
20	reg_gpio_17_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_17_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_17_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_17_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_17_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_17_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_17_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_17_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_17_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.23 gpio_cfg18

Address：0x2000090c

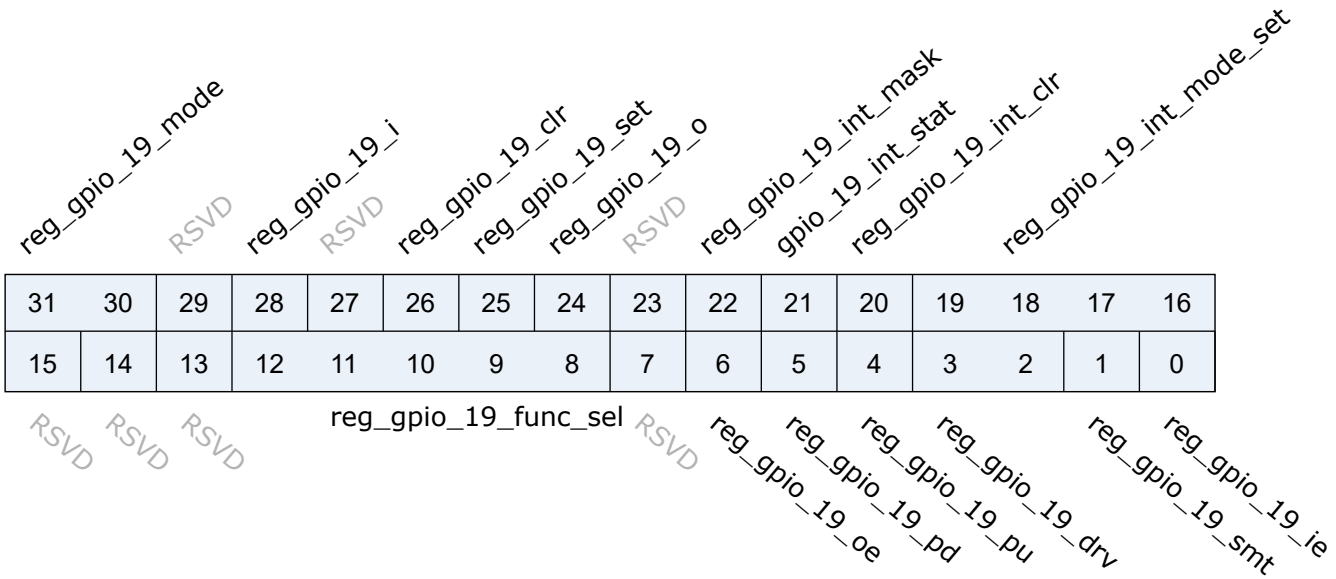


Bits	Name	Type	Reset	Description
31:30	reg_gpio_18_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_18_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_18_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_18_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_18_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_18_int_mask	r/w	1	mask interrupt (1)
21	gpio_18_int_stat	r	0	interrupt status
20	reg_gpio_18_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_18_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_18_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_18_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_18_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_18_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_18_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_18_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_18_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.24 gpio_cfg19

Address：0x20000910

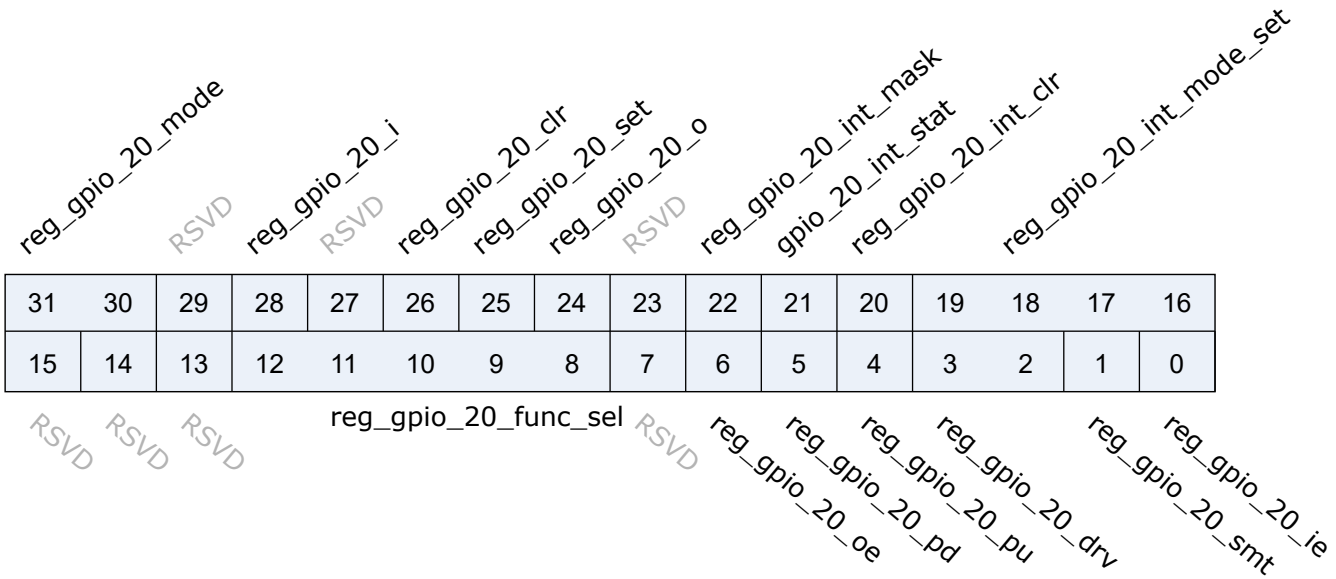


Bits	Name	Type	Reset	Description
31:30	reg_gpio_19_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_19_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_19_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_19_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_19_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_19_int_mask	r/w	1	mask interrupt (1)
21	gpio_19_int_stat	r	0	interrupt status
20	reg_gpio_19_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_19_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_19_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_19_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_19_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_19_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_19_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_19_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_19_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.25 gpio_cfg20

Address：0x20000914

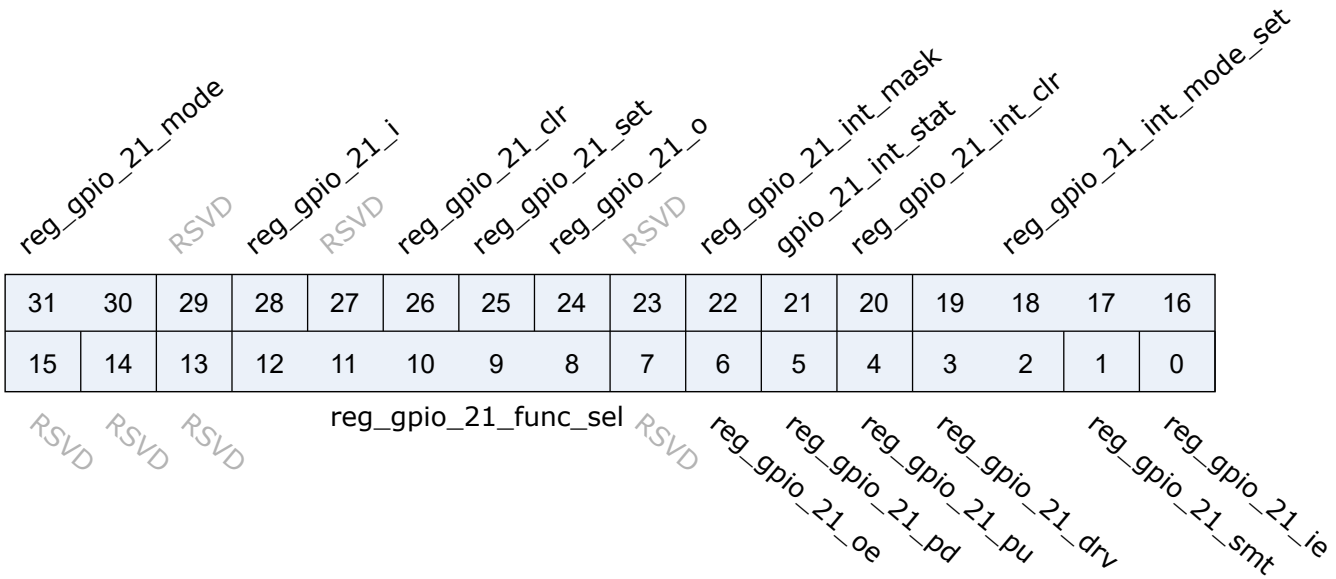


Bits	Name	Type	Reset	Description
31:30	reg_gpio_20_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_20_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_20_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_20_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_20_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_20_int_mask	r/w	1	mask interrupt (1)
21	gpio_20_int_stat	r	0	interrupt status
20	reg_gpio_20_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_20_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_20_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_20_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_20_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_20_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_20_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_20_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_20_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.26 gpio_cfg21

Address：0x20000918

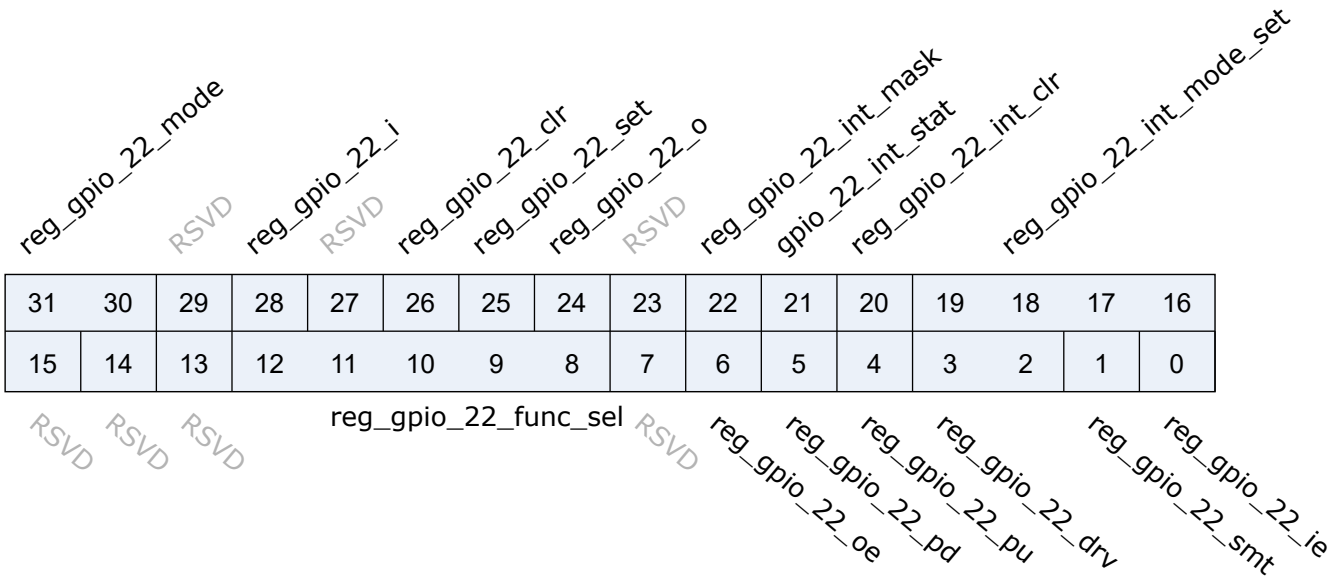


Bits	Name	Type	Reset	Description
31:30	reg_gpio_21_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_21_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_21_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_21_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_21_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_21_int_mask	r/w	1	mask interrupt (1)
21	gpio_21_int_stat	r	0	interrupt status
20	reg_gpio_21_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_21_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_21_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_21_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_21_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_21_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_21_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_21_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_21_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.27 gpio_cfg22

Address：0x2000091c

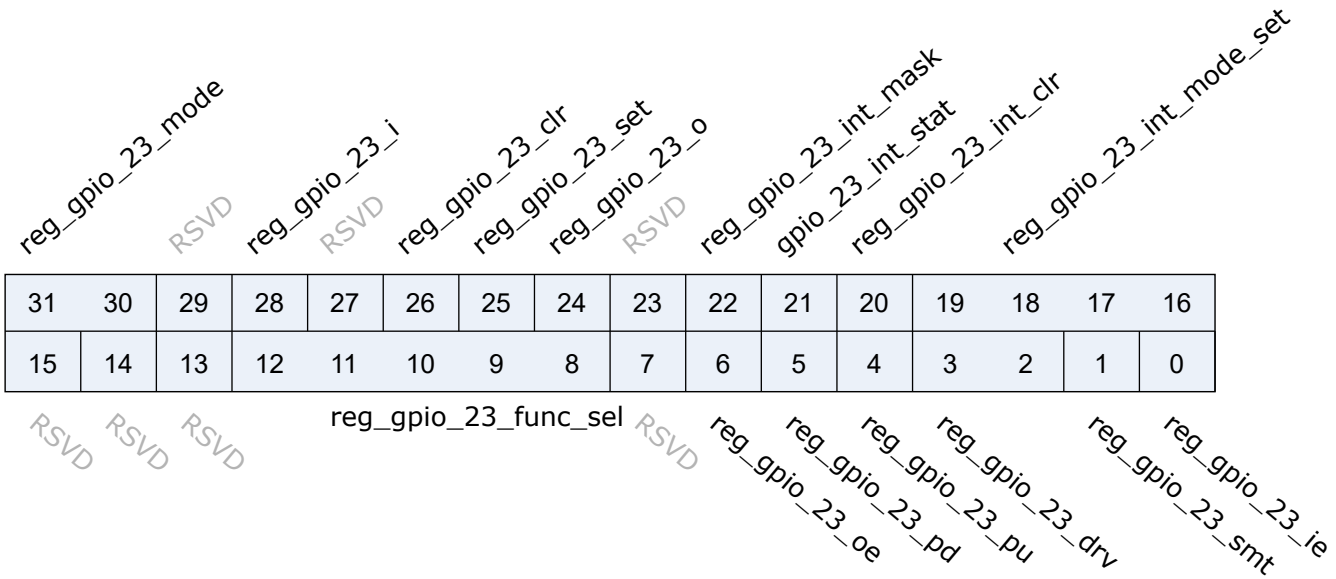


Bits	Name	Type	Reset	Description
31:30	reg_gpio_22_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_22_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_22_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_22_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_22_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_22_int_mask	r/w	1	mask interrupt (1)
21	gpio_22_int_stat	r	0	interrupt status
20	reg_gpio_22_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_22_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_22_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_22_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_22_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_22_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_22_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_22_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_22_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.28 gpio_cfg23

Address：0x20000920

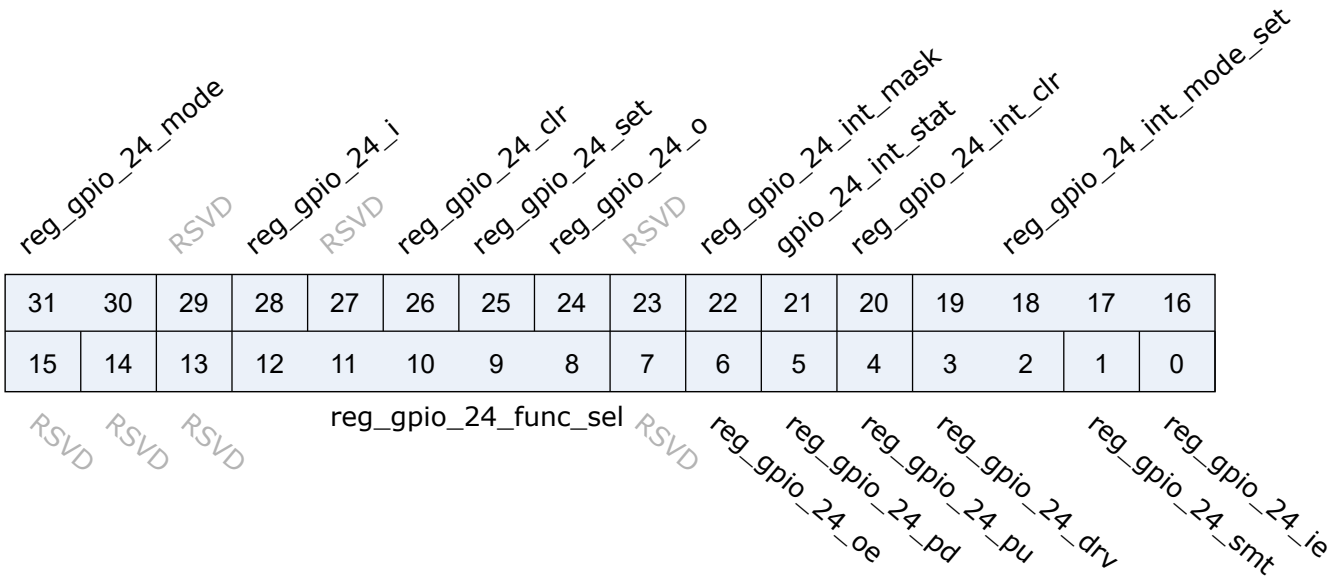


Bits	Name	Type	Reset	Description
31:30	reg_gpio_23_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_23_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_23_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_23_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_23_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_23_int_mask	r/w	1	mask interrupt (1)
21	gpio_23_int_stat	r	0	interrupt status
20	reg_gpio_23_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_23_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_23_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_23_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_23_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_23_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_23_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_23_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_23_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.29 gpio_cfg24

Address：0x20000924

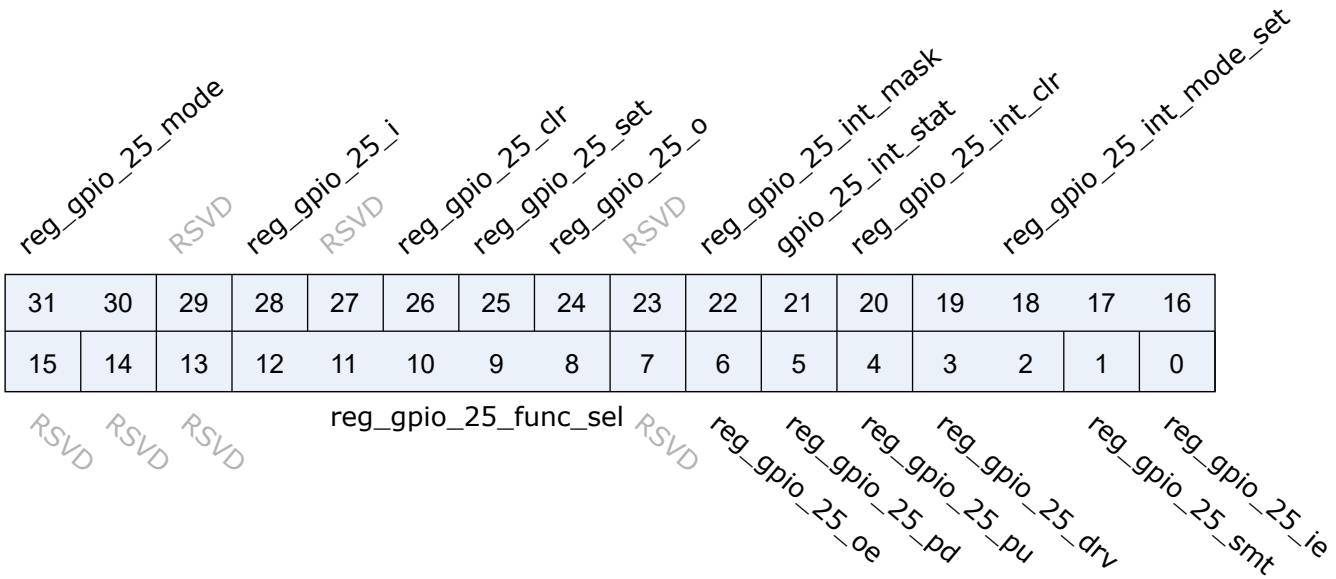


Bits	Name	Type	Reset	Description
31:30	reg_gpio_24_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_24_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_24_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_24_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_24_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_24_int_mask	r/w	1	mask interrupt (1)
21	gpio_24_int_stat	r	0	interrupt status
20	reg_gpio_24_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_24_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_24_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_24_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_24_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_24_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_24_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_24_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_24_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.30 gpio_cfg25

Address：0x20000928

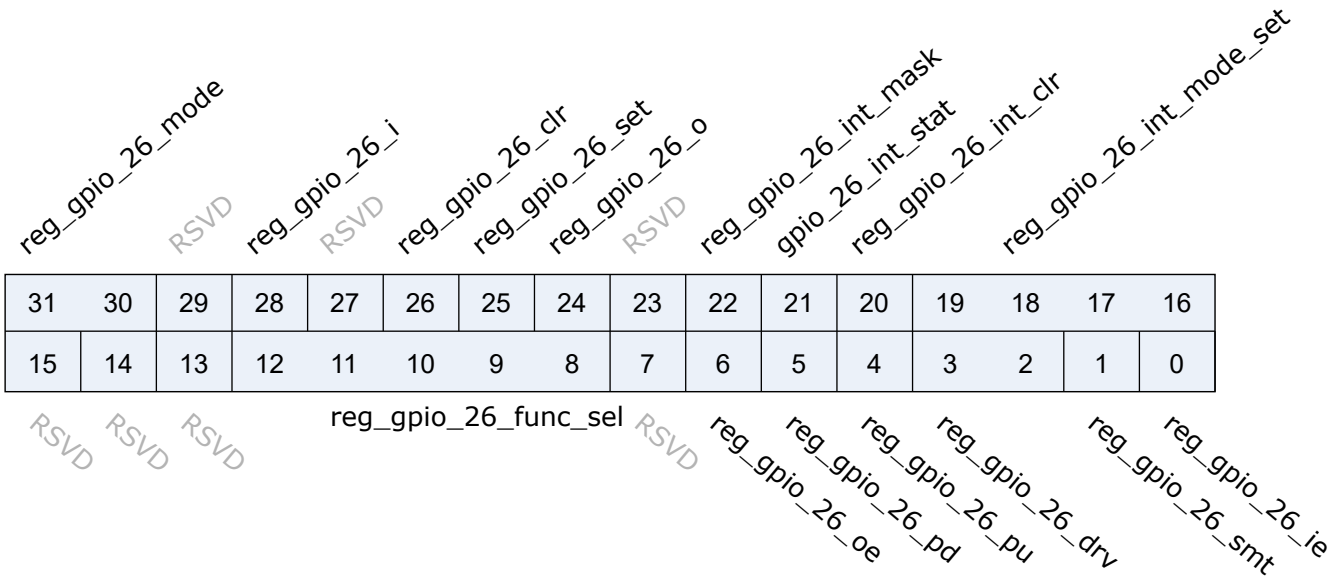


Bits	Name	Type	Reset	Description
31:30	reg_gpio_25_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_25_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_25_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_25_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_25_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_25_int_mask	r/w	1	mask interrupt (1)
21	gpio_25_int_stat	r	0	interrupt status
20	reg_gpio_25_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_25_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_25_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_25_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_25_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_25_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_25_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_25_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_25_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.31 gpio_cfg26

Address：0x2000092c

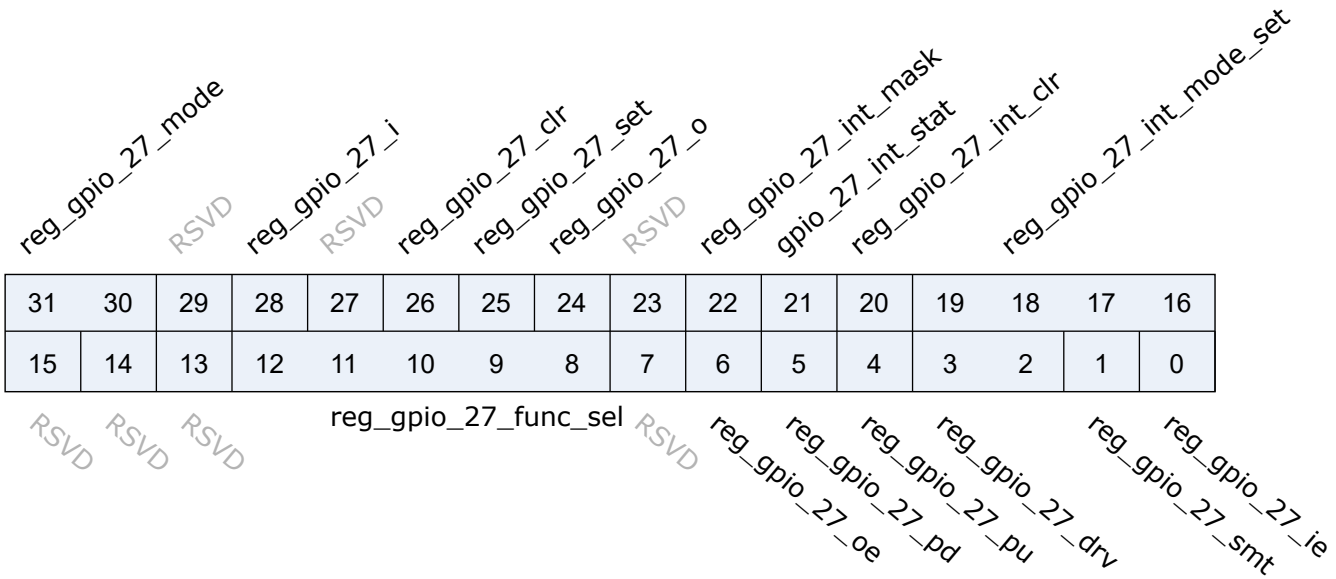


Bits	Name	Type	Reset	Description
31:30	reg_gpio_26_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_26_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_26_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_26_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_26_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_26_int_mask	r/w	1	mask interrupt (1)
21	gpio_26_int_stat	r	0	interrupt status
20	reg_gpio_26_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_26_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_26_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_26_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_26_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_26_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_26_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_26_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_26_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.32 gpio_cfg27

Address：0x20000930

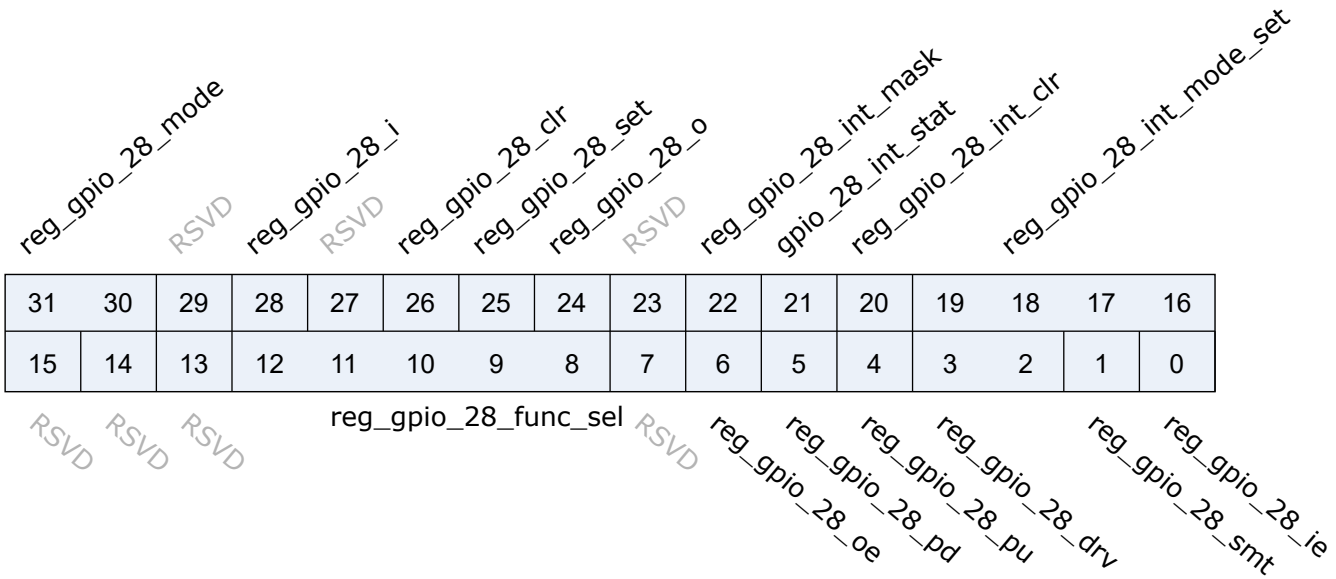


Bits	Name	Type	Reset	Description
31:30	reg_gpio_27_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_27_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_27_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_27_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_27_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_27_int_mask	r/w	1	mask interrupt (1)
21	gpio_27_int_stat	r	0	interrupt status
20	reg_gpio_27_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_27_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_27_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_27_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_27_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_27_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_27_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_27_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_27_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.33 gpio_cfg28

Address：0x20000934

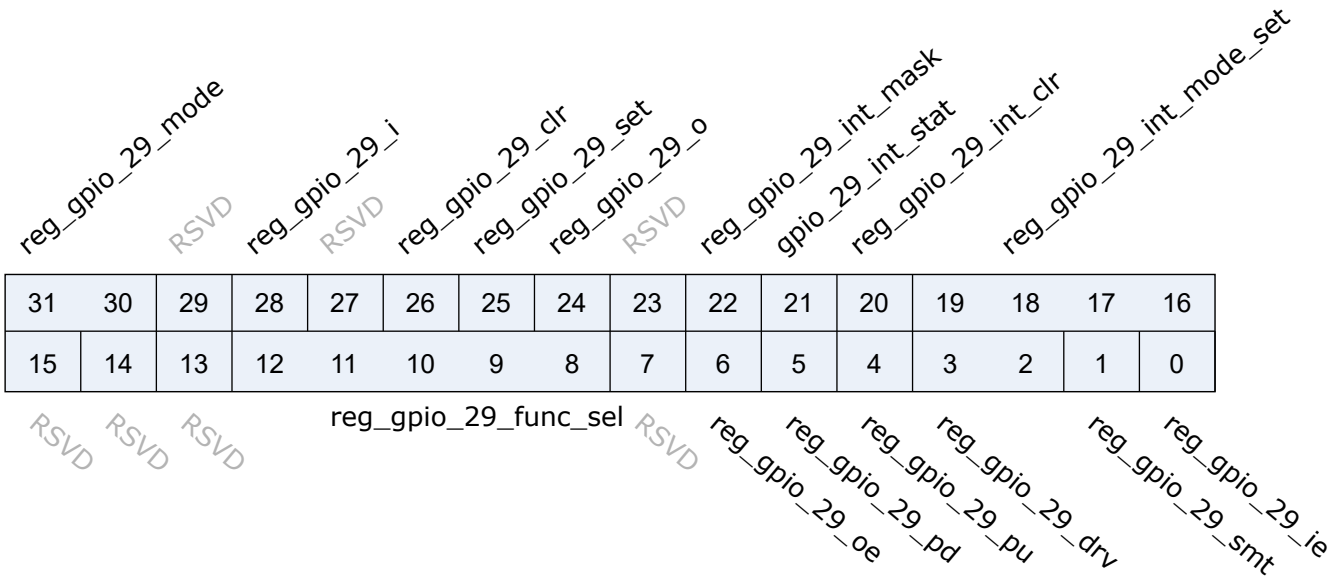


Bits	Name	Type	Reset	Description
31:30	reg_gpio_28_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_28_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_28_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_28_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_28_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_28_int_mask	r/w	1	mask interrupt (1)
21	gpio_28_int_stat	r	0	interrupt status
20	reg_gpio_28_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_28_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_28_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_28_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_28_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_28_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_28_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_28_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_28_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.34 gpio_cfg29

Address：0x20000938

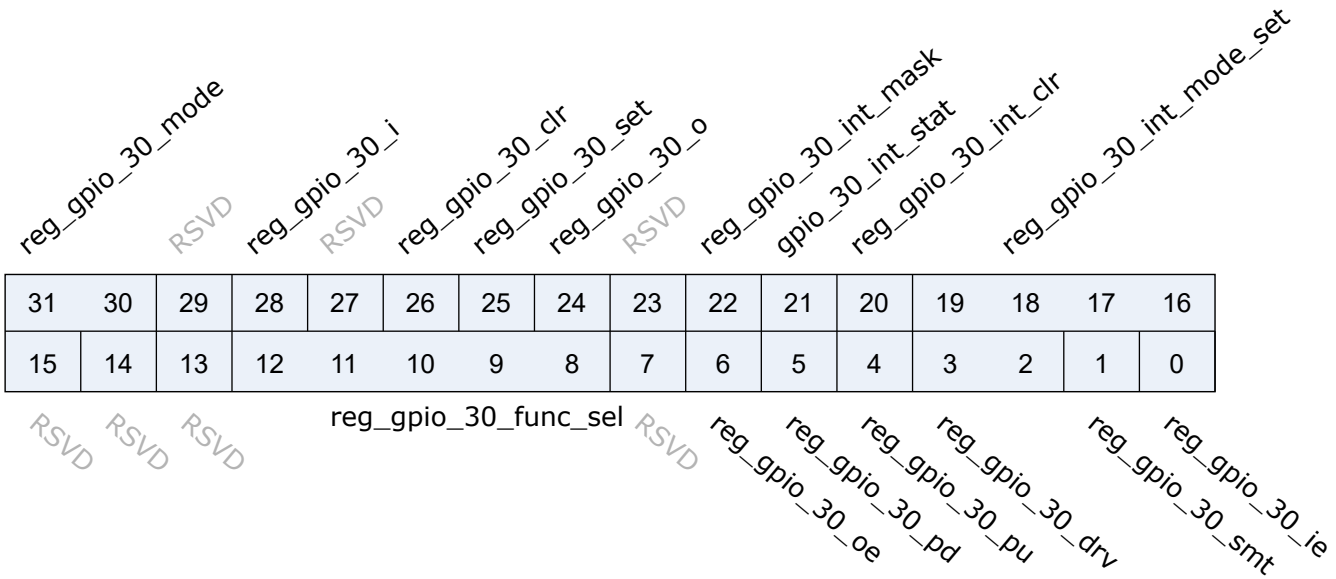


Bits	Name	Type	Reset	Description
31:30	reg_gpio_29_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_29_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_29_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_29_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_29_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_29_int_mask	r/w	1	mask interrupt (1)
21	gpio_29_int_stat	r	0	interrupt status
20	reg_gpio_29_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_29_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_29_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_29_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_29_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_29_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_29_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_29_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_29_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.35 gpio_cfg30

Address：0x2000093c

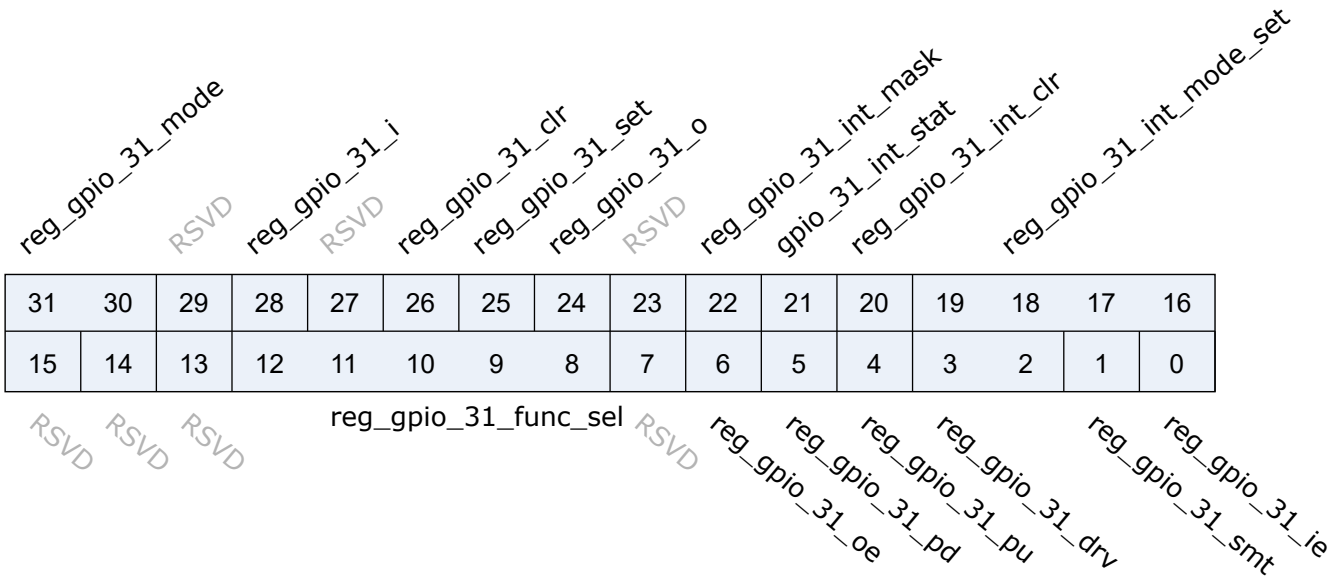


Bits	Name	Type	Reset	Description
31:30	reg_gpio_30_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_30_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_30_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_30_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_30_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_30_int_mask	r/w	1	mask interrupt (1)
21	gpio_30_int_stat	r	0	interrupt status
20	reg_gpio_30_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_30_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_30_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_30_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_30_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_30_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_30_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_30_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_30_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.36 gpio_cfg31

Address：0x20000940

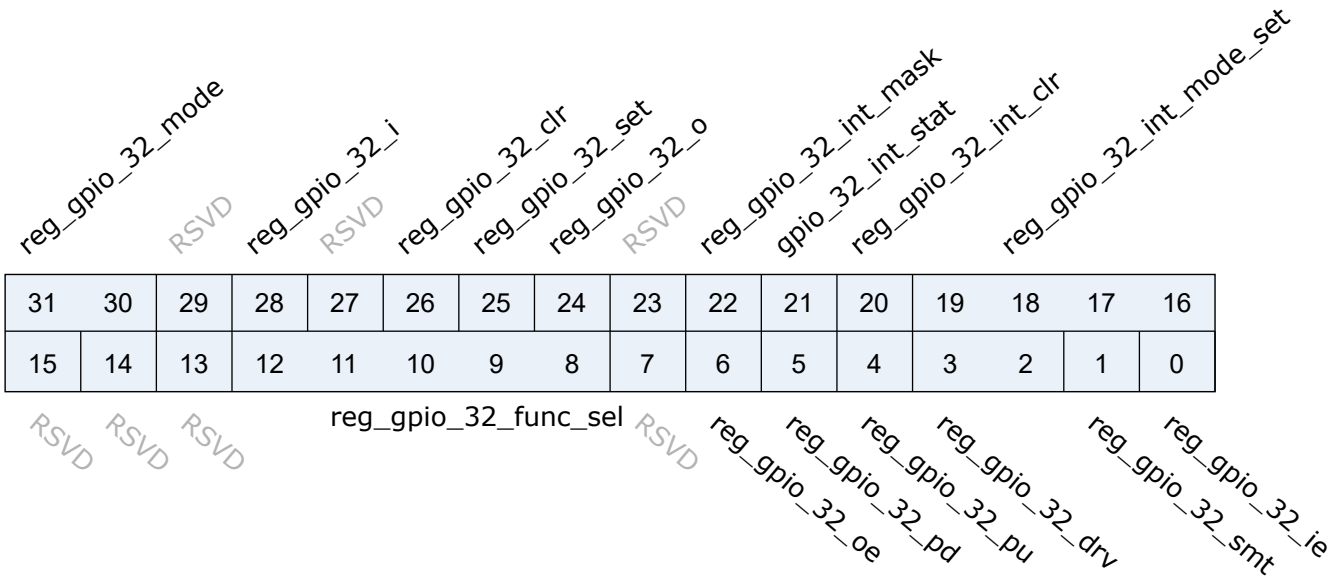


Bits	Name	Type	Reset	Description
31:30	reg_gpio_31_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_31_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_31_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_31_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_31_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_31_int_mask	r/w	1	mask interrupt (1)
21	gpio_31_int_stat	r	0	interrupt status
20	reg_gpio_31_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_31_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_31_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_31_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_31_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_31_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_31_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_31_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_31_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.37 gpio_cfg32

Address：0x20000944

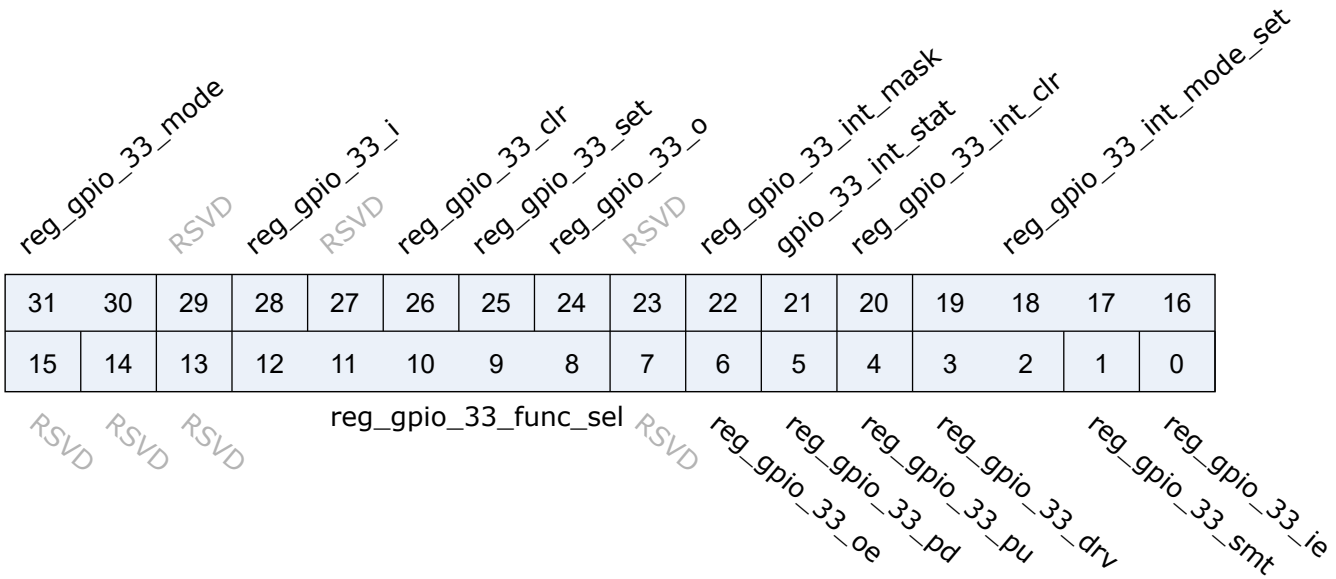


Bits	Name	Type	Reset	Description
31:30	reg_gpio_32_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_32_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_32_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_32_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_32_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_32_int_mask	r/w	1	mask interrupt (1)
21	gpio_32_int_stat	r	0	interrupt status
20	reg_gpio_32_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_32_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_32_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_32_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_32_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_32_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_32_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_32_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_32_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.38 gpio_cfg33

Address：0x20000948

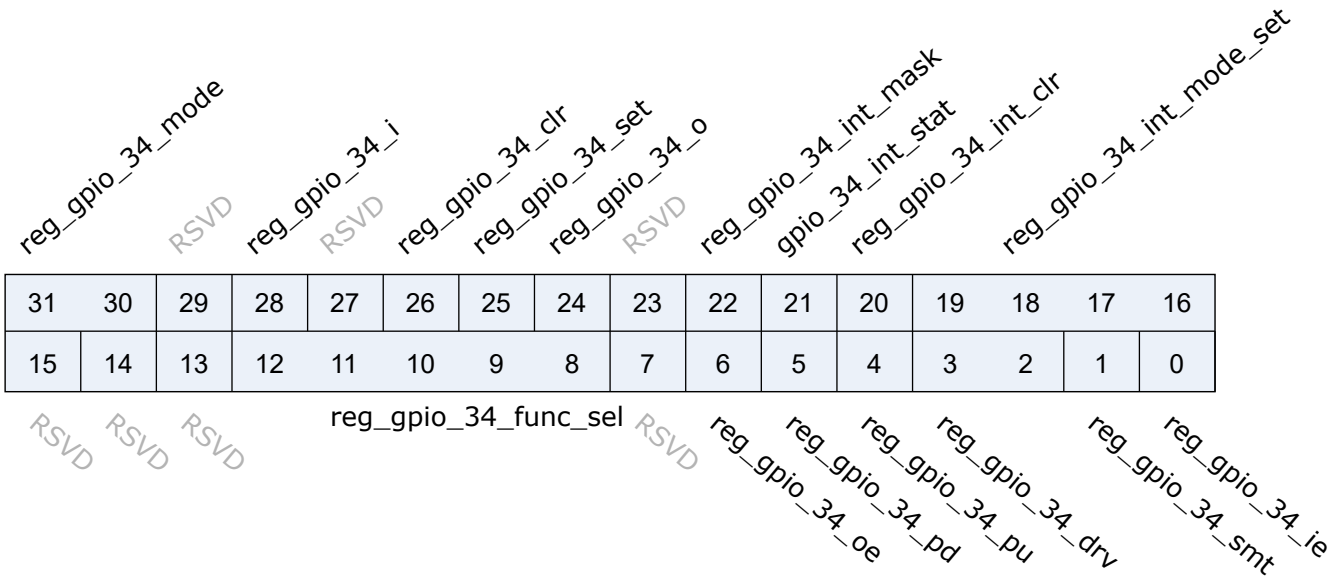


Bits	Name	Type	Reset	Description
31:30	reg_gpio_33_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_33_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_33_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_33_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_33_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_33_int_mask	r/w	1	mask interrupt (1)
21	gpio_33_int_stat	r	0	interrupt status
20	reg_gpio_33_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_33_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_33_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_33_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_33_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_33_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_33_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_33_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_33_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.39 gpio_cfg34

Address：0x2000094c

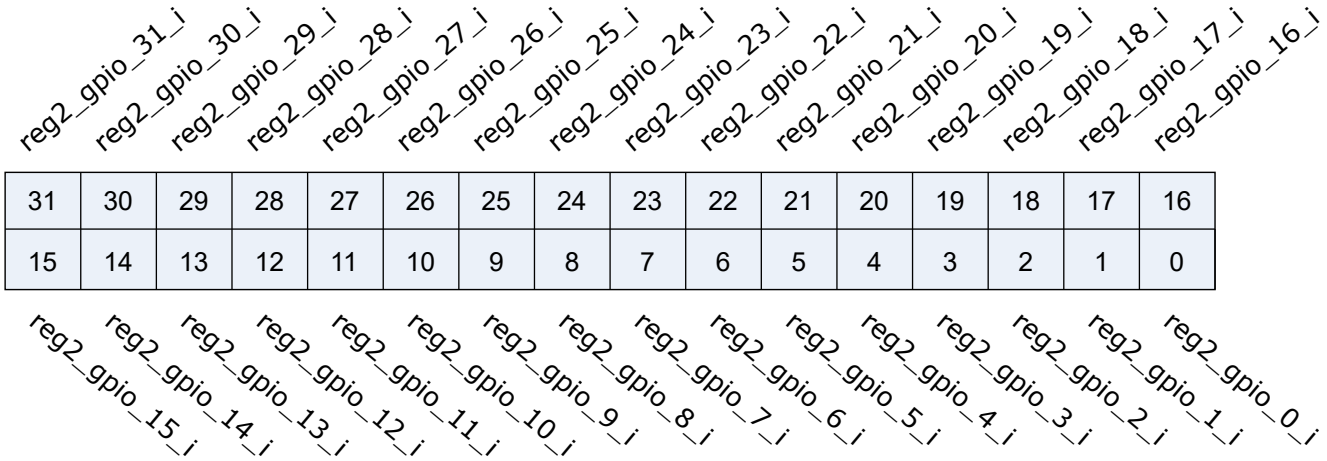


Bits	Name	Type	Reset	Description
31:30	reg_gpio_34_mode	r/w	0	When GPIO Function Selected to SWGPIO 00 (Output Value Mode): GPIO Output by reg_gpio_x_o Value 01 (Set/Celar Mode) :GPIO Output set by reg_gpio_x_set and clear by reg_gpio_x_clr 10 : SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Output value by gpio_dma_o 11: SWGPIO Source comes from GPIO DMA (GPIO DMA Mode), GPIO Outout value by gpio_dma_set/gpio_dma_clr
29	RSVD			
28	reg_gpio_34_i	r	0	GPIO input state
27	RSVD			
26	reg_gpio_34_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg_gpio_34_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
24	reg_gpio_34_o	r/w	0	When SWGPIO @ Output Value Mode 0 : set GPIO output Value to 0 1 : set GPIO output Value to 1

Bits	Name	Type	Reset	Description
23	RSVD			
22	reg_gpio_34_int_mask	r/w	1	mask interrupt (1)
21	gpio_34_int_stat	r	0	interrupt status
20	reg_gpio_34_int_clr	r/w	0	clear interrupt
19:16	reg_gpio_34_int_mode_set	r/w	0	0000 : sync falling edge trigger 0001 : sync rising edge trigger 0010 : sync low level trigger 0011 : sync high level trigger 01xx : sync rising & falling edge trigger 1000 : async falling edge trigger 1001 : async rising edge trigger 1010 : async low level trigger 1011 : async high level trigger
15:13	RSVD			
12:8	reg_gpio_34_func_sel	r/w	5'hB	GPIO Function Select (Default : SW-GPIO)
7	RSVD			
6	reg_gpio_34_oe	r/w	0	Register Controlled GPIO Output Enable (Used when GPIO Function select to Register Control GPIO) 0: disable GPIO output 1: enable GPIO output
5	reg_gpio_34_pd	r/w	0	GPIO Pull Down Control 0: not pull down 1: pull down
4	reg_gpio_34_pu	r/w	0	GPIO Pull Up Control 0: not pull up 1: pull up
3:2	reg_gpio_34_drv	r/w	0	GPIO Driving Control 0: driver level 0 1: driver level 1 2: driver level 2 3: driver level 3
1	reg_gpio_34_smt	r/w	1	GPIO SMT Control 0: disable schmitt trigger 1: enable schmitt trigger
0	reg_gpio_34_ie	r/w	0	GPIO Input Enable 0: disable GPIO input 1: enable GPIO input

4.8.40 gpio_cfg128

Address：0x20000ac4

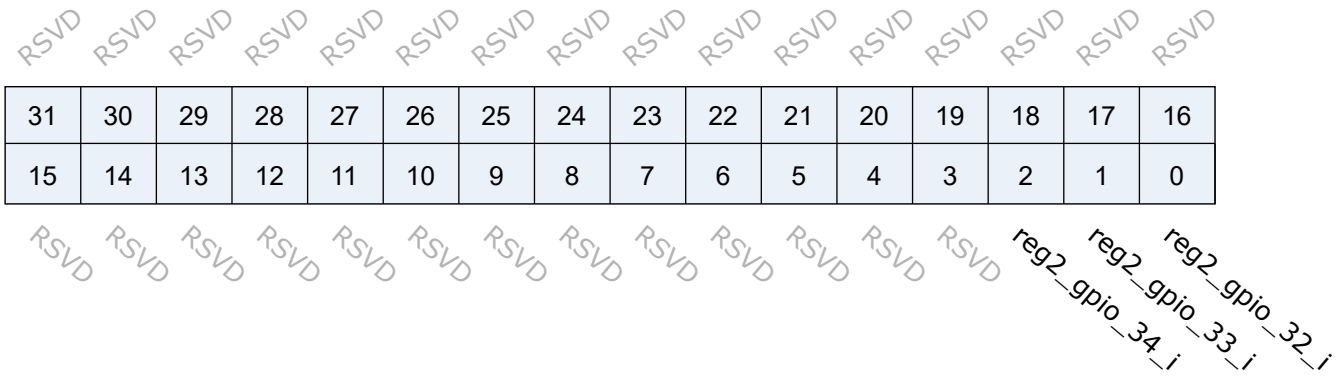


Bits	Name	Type	Reset	Description
31	reg2_gpio_31_i	r	0	Register Controlled GPIO Input value
30	reg2_gpio_30_i	r	0	Register Controlled GPIO Input value
29	reg2_gpio_29_i	r	0	Register Controlled GPIO Input value
28	reg2_gpio_28_i	r	0	Register Controlled GPIO Input value
27	reg2_gpio_27_i	r	0	Register Controlled GPIO Input value
26	reg2_gpio_26_i	r	0	Register Controlled GPIO Input value
25	reg2_gpio_25_i	r	0	Register Controlled GPIO Input value
24	reg2_gpio_24_i	r	0	Register Controlled GPIO Input value
23	reg2_gpio_23_i	r	0	Register Controlled GPIO Input value
22	reg2_gpio_22_i	r	0	Register Controlled GPIO Input value
21	reg2_gpio_21_i	r	0	Register Controlled GPIO Input value
20	reg2_gpio_20_i	r	0	Register Controlled GPIO Input value
19	reg2_gpio_19_i	r	0	Register Controlled GPIO Input value
18	reg2_gpio_18_i	r	0	Register Controlled GPIO Input value
17	reg2_gpio_17_i	r	0	Register Controlled GPIO Input value
16	reg2_gpio_16_i	r	0	Register Controlled GPIO Input value
15	reg2_gpio_15_i	r	0	Register Controlled GPIO Input value
14	reg2_gpio_14_i	r	0	Register Controlled GPIO Input value
13	reg2_gpio_13_i	r	0	Register Controlled GPIO Input value

Bits	Name	Type	Reset	Description
12	reg2_gpio_12_i	r	0	Register Controlled GPIO Input value
11	reg2_gpio_11_i	r	0	Register Controlled GPIO Input value
10	reg2_gpio_10_i	r	0	Register Controlled GPIO Input value
9	reg2_gpio_9_i	r	0	Register Controlled GPIO Input value
8	reg2_gpio_8_i	r	0	Register Controlled GPIO Input value
7	reg2_gpio_7_i	r	0	Register Controlled GPIO Input value
6	reg2_gpio_6_i	r	0	Register Controlled GPIO Input value
5	reg2_gpio_5_i	r	0	Register Controlled GPIO Input value
4	reg2_gpio_4_i	r	0	Register Controlled GPIO Input value
3	reg2_gpio_3_i	r	0	Register Controlled GPIO Input value
2	reg2_gpio_2_i	r	0	Register Controlled GPIO Input value
1	reg2_gpio_1_i	r	0	Register Controlled GPIO Input value
0	reg2_gpio_0_i	r	0	Register Controlled GPIO Input value

4.8.41 gpio_cfg129

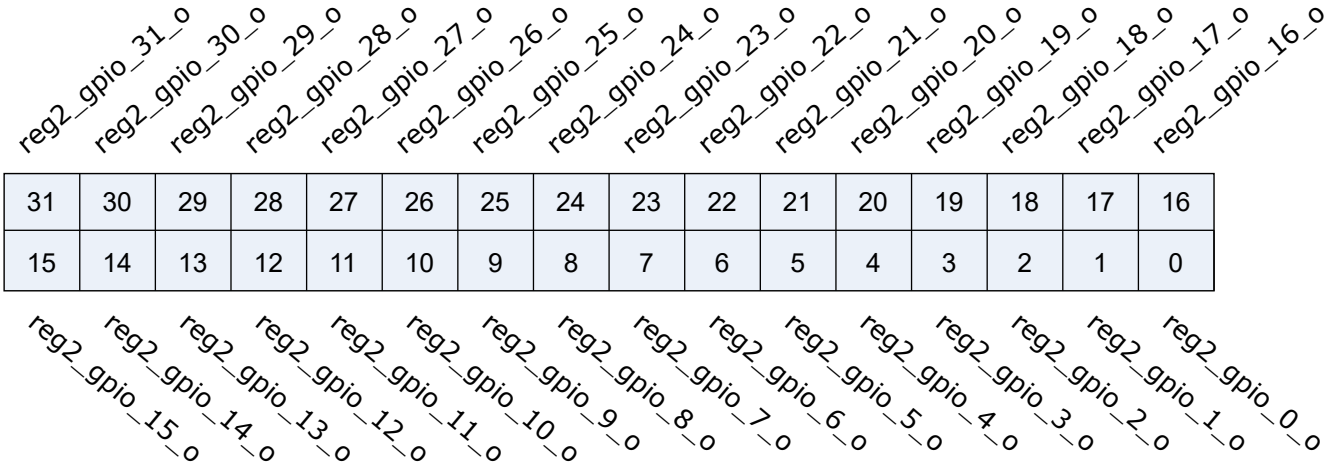
Address：0x20000ac8



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_i	r	0	Register Controlled GPIO Input value
1	reg2_gpio_33_i	r	0	Register Controlled GPIO Input value
0	reg2_gpio_32_i	r	0	Register Controlled GPIO Input value

4.8.42 gpio_cfg136

Address：0x20000ae4

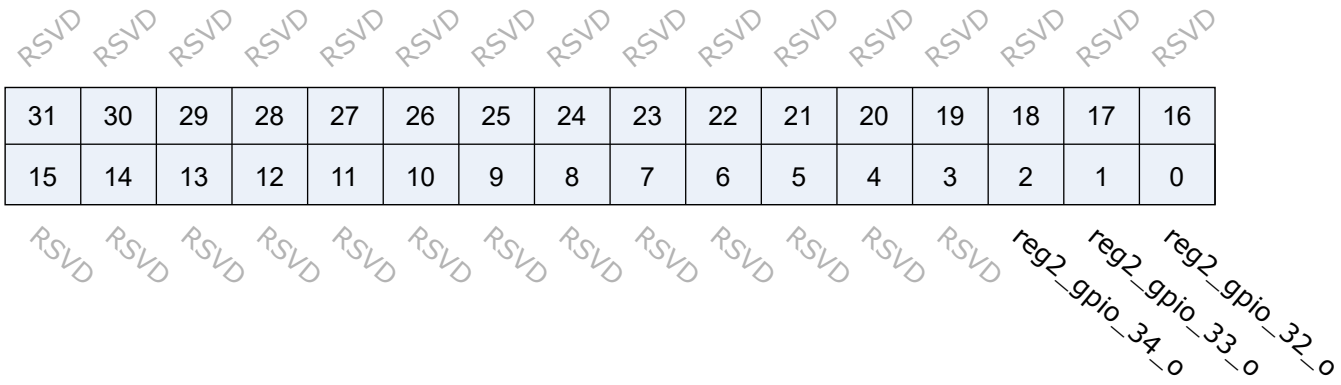


Bits	Name	Type	Reset	Description
31	reg2_gpio_31_o	r/w	0	Register Controlled GPIO Output Value
30	reg2_gpio_30_o	r/w	0	Register Controlled GPIO Output Value
29	reg2_gpio_29_o	r/w	0	Register Controlled GPIO Output Value
28	reg2_gpio_28_o	r/w	0	Register Controlled GPIO Output Value
27	reg2_gpio_27_o	r/w	0	Register Controlled GPIO Output Value
26	reg2_gpio_26_o	r/w	0	Register Controlled GPIO Output Value
25	reg2_gpio_25_o	r/w	0	Register Controlled GPIO Output Value
24	reg2_gpio_24_o	r/w	0	Register Controlled GPIO Output Value
23	reg2_gpio_23_o	r/w	0	Register Controlled GPIO Output Value
22	reg2_gpio_22_o	r/w	0	Register Controlled GPIO Output Value
21	reg2_gpio_21_o	r/w	0	Register Controlled GPIO Output Value
20	reg2_gpio_20_o	r/w	0	Register Controlled GPIO Output Value
19	reg2_gpio_19_o	r/w	0	Register Controlled GPIO Output Value
18	reg2_gpio_18_o	r/w	0	Register Controlled GPIO Output Value
17	reg2_gpio_17_o	r/w	0	Register Controlled GPIO Output Value
16	reg2_gpio_16_o	r/w	0	Register Controlled GPIO Output Value
15	reg2_gpio_15_o	r/w	0	Register Controlled GPIO Output Value
14	reg2_gpio_14_o	r/w	0	Register Controlled GPIO Output Value
13	reg2_gpio_13_o	r/w	0	Register Controlled GPIO Output Value
12	reg2_gpio_12_o	r/w	0	Register Controlled GPIO Output Value

Bits	Name	Type	Reset	Description
11	reg2_gpio_11_o	r/w	0	Register Controlled GPIO Output Value
10	reg2_gpio_10_o	r/w	0	Register Controlled GPIO Output Value
9	reg2_gpio_9_o	r/w	0	Register Controlled GPIO Output Value
8	reg2_gpio_8_o	r/w	0	Register Controlled GPIO Output Value
7	reg2_gpio_7_o	r/w	0	Register Controlled GPIO Output Value
6	reg2_gpio_6_o	r/w	0	Register Controlled GPIO Output Value
5	reg2_gpio_5_o	r/w	0	Register Controlled GPIO Output Value
4	reg2_gpio_4_o	r/w	0	Register Controlled GPIO Output Value
3	reg2_gpio_3_o	r/w	0	Register Controlled GPIO Output Value
2	reg2_gpio_2_o	r/w	0	Register Controlled GPIO Output Value
1	reg2_gpio_1_o	r/w	0	Register Controlled GPIO Output Value
0	reg2_gpio_0_o	r/w	0	Register Controlled GPIO Output Value

4.8.43 gpio_cfg137

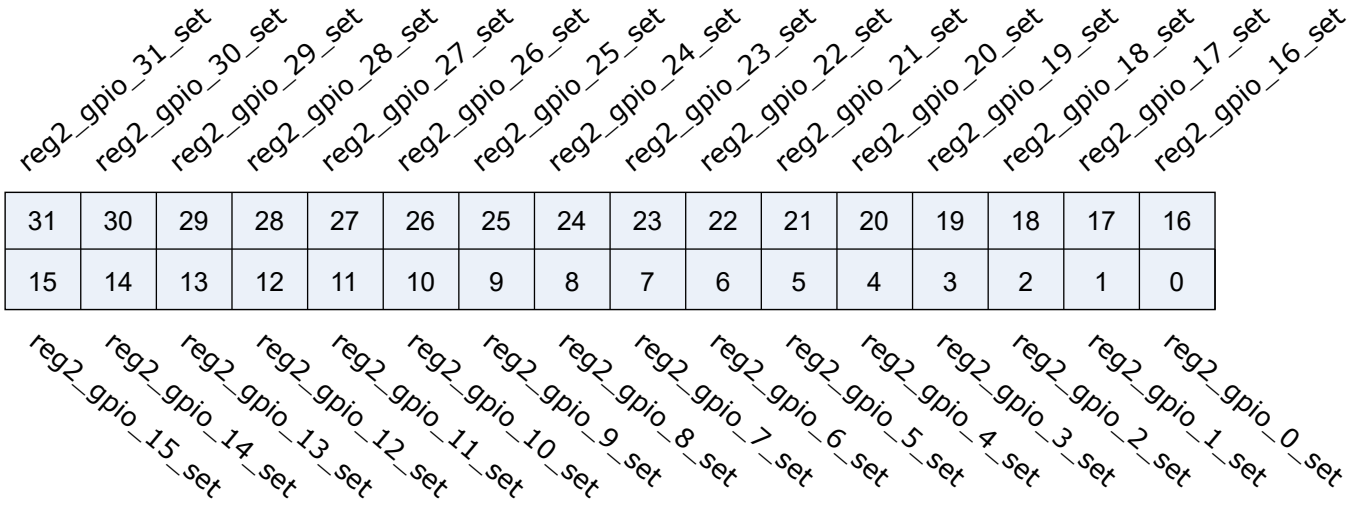
Address：0x20000ae8



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_o	r/w	0	Register Controlled GPIO Output Value
1	reg2_gpio_33_o	r/w	0	Register Controlled GPIO Output Value
0	reg2_gpio_32_o	r/w	0	Register Controlled GPIO Output Value

4.8.44 gpio_cfg138

Address：0x20000aec



Bits	Name	Type	Reset	Description
31	reg2_gpio_31_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
30	reg2_gpio_30_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
29	reg2_gpio_29_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
28	reg2_gpio_28_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
27	reg2_gpio_27_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
26	reg2_gpio_26_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
25	reg2_gpio_25_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

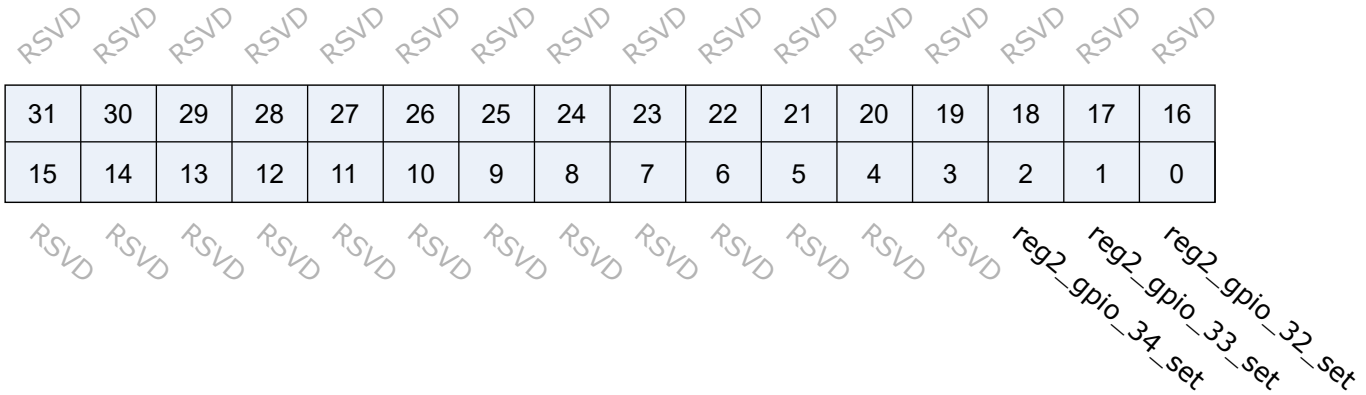
Bits	Name	Type	Reset	Description
24	reg2_gpio_24_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
23	reg2_gpio_23_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
22	reg2_gpio_22_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
21	reg2_gpio_21_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
20	reg2_gpio_20_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
19	reg2_gpio_19_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
18	reg2_gpio_18_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
17	reg2_gpio_17_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
16	reg2_gpio_16_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
15	reg2_gpio_15_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
14	reg2_gpio_14_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
13	reg2_gpio_13_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
12	reg2_gpio_12_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
11	reg2_gpio_11_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
10	reg2_gpio_10_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
9	reg2_gpio_9_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
8	reg2_gpio_8_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
7	reg2_gpio_7_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
6	reg2_gpio_6_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
5	reg2_gpio_5_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
4	reg2_gpio_4_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
3	reg2_gpio_3_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
2	reg2_gpio_2_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
1	reg2_gpio_1_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
0	reg2_gpio_0_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect

4.8.45 gpio_cfg139

Address：0x20000af0



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
1	reg2_gpio_33_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
0	reg2_gpio_32_set	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will set GPIO output value to 1,when set/clr at the same time, only set take effect
-1:32	RSVD			
31	reg2_gpio_31_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
30	reg2_gpio_30_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
29	reg2_gpio_29_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
28	reg2_gpio_28_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
27	reg2_gpio_27_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
26	reg2_gpio_26_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
25	reg2_gpio_25_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
24	reg2_gpio_24_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
23	reg2_gpio_23_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
22	reg2_gpio_22_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
21	reg2_gpio_21_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
20	reg2_gpio_20_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
19	reg2_gpio_19_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
18	reg2_gpio_18_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
17	reg2_gpio_17_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
16	reg2_gpio_16_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
15	reg2_gpio_15_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
14	reg2_gpio_14_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
13	reg2_gpio_13_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
12	reg2_gpio_12_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
11	reg2_gpio_11_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
10	reg2_gpio_10_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
9	reg2_gpio_9_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
8	reg2_gpio_8_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
7	reg2_gpio_7_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
6	reg2_gpio_6_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
5	reg2_gpio_5_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
4	reg2_gpio_4_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
3	reg2_gpio_3_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
2	reg2_gpio_2_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
1	reg2_gpio_1_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
0	reg2_gpio_0_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

4.8.46 gpio_cfg141

Address：0x20000af8

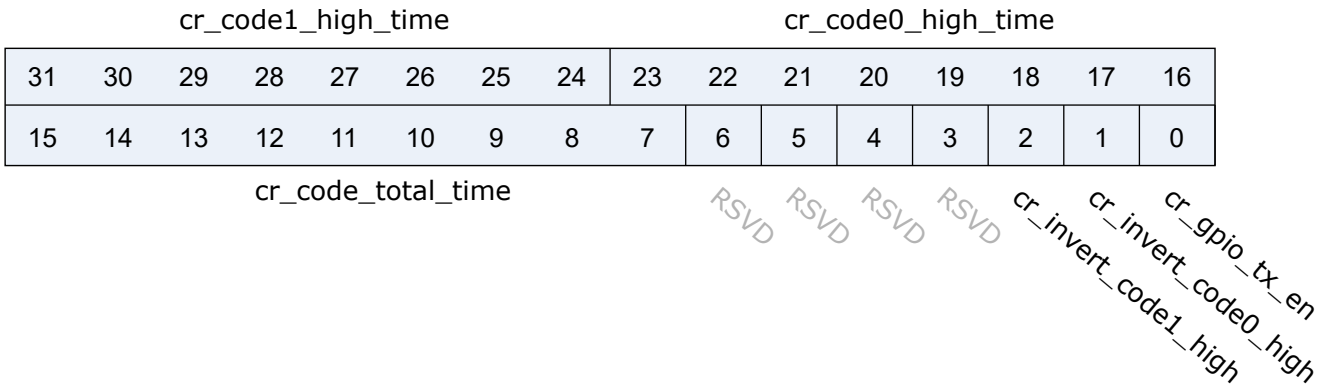
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	reg2_gpio_34_clr	reg2_gpio_33_clr	reg2_gpio_32_clr

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	reg2_gpio_34_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

Bits	Name	Type	Reset	Description
1	reg2_gpio_33_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect
0	reg2_gpio_32_clr	w1p	0	When SWGPIO @ Set/Clear Mode Set this bit will clear GPIO output value to 0,when set/clr at the same time, only set take effect

4.8.47 gpio_cfg142

Address：0x20000afc

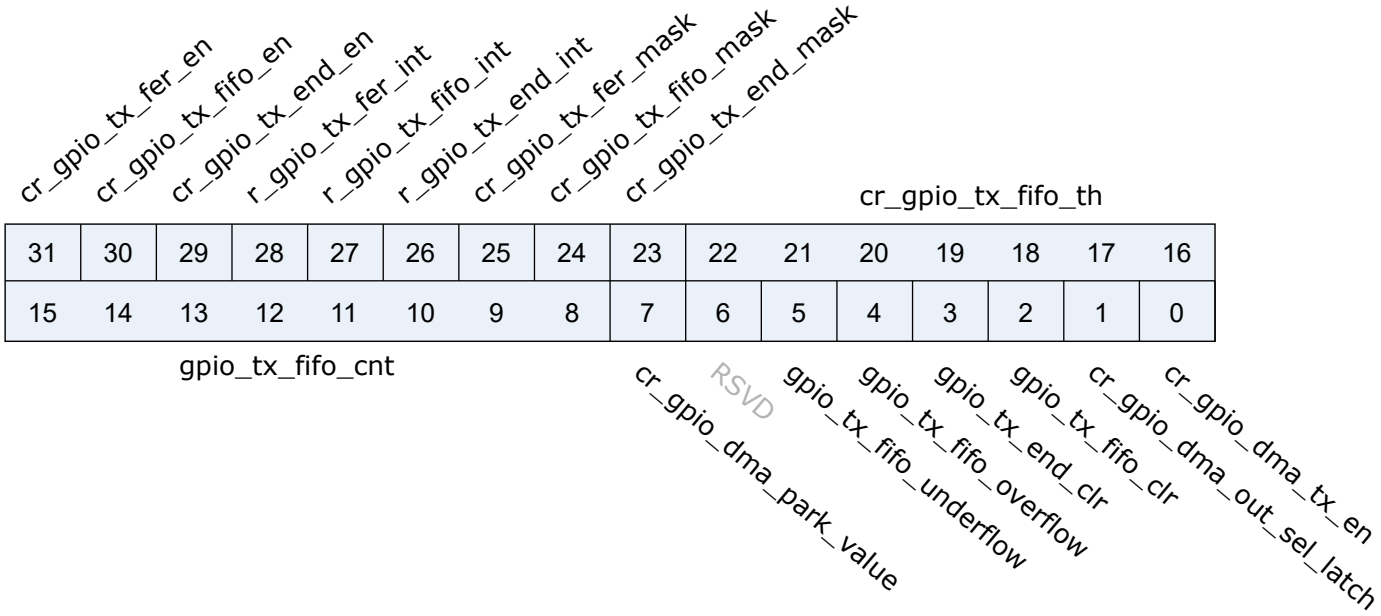


Bits	Name	Type	Reset	Description
31:24	cr_code1_high_time	r/w	8'd200	Used to generate Code 1 Duty Cycle Waveform (in units of xclk (XTAL or RC32M) clock cycle) waveform keep high during cr_code1_high_time waveform keep low during cr_code1_low_time (cr_code_-total_time - cr_code1_high_time)
23:16	cr_code0_high_time	r/w	8'd200	Used to generate Code 0 Duty Cycle Waveform (in units of xclk (XTAL or RC32M) clock cycle) waveform keep high during cr_code0_high_time waveform keep low during cr_code0_low_time (cr_code_-total_time - cr_code0_high_time)
15:7	cr_code_total_time	r/w	9'd400	Used to define Code0/Code1 total waveform time
6:3	RSVD			
2	cr_invert_code1_high	r/w	1'b0	code1 phase 0: first output high level and then output low level 1: first output low level and then output high level

Bits	Name	Type	Reset	Description
1	cr_invert_code0_high	r/w	1'b0	code0 phase 0: first output high level and then output low level 1: first output low level and then output high level
0	cr_gpio_tx_en	r/w	1'b0	Enable GPIO DMA OUT/GPIO DMA Latch 0: disable GPIO TX function 1: enable GPIO TX function

4.8.48 gpio_cfg143

Address：0x20000b00



Bits	Name	Type	Reset	Description
31	cr_gpio_tx_fer_en	r/w	1'b1	Interrupt enable of gpio_tx_fer_int (GPIO DMA FIFO Under-flow or Overflow)
30	cr_gpio_tx_fifo_en	r/w	1'b1	Interrupt enable of gpio_tx_fifo_int
29	cr_gpio_tx_end_en	r/w	1'b1	Interrupt enable of gpio_tx_end_int
28	r_gpio_tx_fer_int	r	1'b0	GPIO TX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
27	r_gpio_tx_fifo_int	r	1'b0	GPIO TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
26	r_gpio_tx_end_int	r	1'b0	GPIO TX END Interrupt (GPIO DMA FIFO Empty)
25	cr_gpio_tx_fer_mask	r/w	1'b1	Interrupt mask of gpio_tx_fer_int

Bits	Name	Type	Reset	Description
24	cr_gpio_tx_fifo_mask	r/w	1'b1	Interrupt mask of gpio_tx_fifo_int
23	cr_gpio_tx_end_mask	r/w	1'b1	Interrupt mask of gpio_tx_end_int
22:16	cr_gpio_tx_fifo_th	r/w	7'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_-fifo_cnt is less than this value
15:8	gpio_tx_fifo_cnt	r	8'd128	TX FIFO available count
7	cr_gpio_dma_park_value	r/w	1'b0	0: park at low level when TX FIFO empty 1: park at high level when TX FIFO empty
6	RSVD			
5	gpio_tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	gpio_tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	gpio_tx_end_clr	w1c	1'b0	Interrupt clear of gpio_tx_end_int
2	gpio_tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	cr_gpio_dma_out_sel_latch	r/w	1'b0	select output format 0: select 16bit output value 1: select latch format (8bit set/8bit clr)
0	cr_gpio_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface 0: disable DMA interface, FIFO is write by cpu 1: enable DMA interface, FIFO is write by DMA

4.8.49 gpio_cfg144

Address：0x20000b04

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
gpio_tx_data_to_fifo															

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	gpio_tx_data_to_fifo	w	x	GPIO TX data FIFO

5 ADC

5.1 Overview

The chip integrates one 12-bit SAR ADC, which supports 12 external analog input signals and several internal analog signals. ADC can work in single conversion mode and multi-channel scanning mode, and the conversion result is 12/14/16-bit LeftJustified mode. ADC has a FIFO with a depth of 32, which supports multiple interrupts and DMA. In addition to measuring ordinary analog signals, ADC can also measure power supply voltage, and detect temperature by measuring the internal/external diode voltage.

5.2 Features

- High performance
  - Selectable 12-bit, 14-bit, 16-bit conversion result output
  - Single-channel continuous conversion mode up to 2M sample rate
  - Other conversion modes up to 500K sample rate
  - Support 2.0V, 3.2V optional reference voltage
  - DMA support for transferring conversion results to memory
  - Single-channel single conversion, single-channel continuous conversion, multi-channel single conversion, multi-channel continuous conversion
  - Supports both single-ended and differential input modes
  - Support jitter compensation
  - Support user-set conversion result offset value
- Number of analog channels
  - Twelve external analog channels

- Two internal DAC channels
- One VBAT/2 channel
- One TSEN channel

5.3 Functional Description

The block diagram of ADC module is shown as follows.

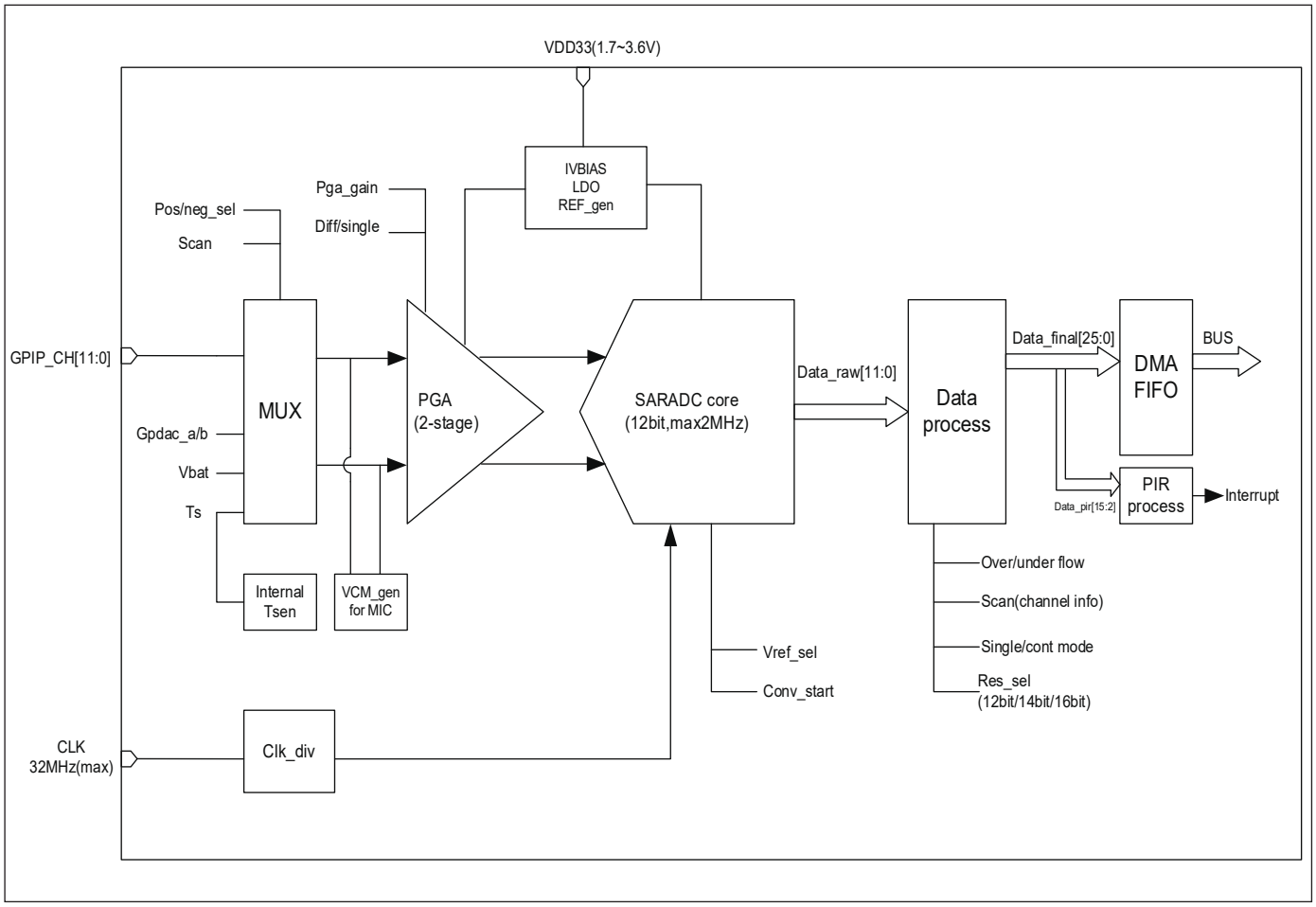


Fig. 5.1: Block Diagram of ADC

The ADC module consists of front-end input channel selector, programmable amplifier, ADC sampling module, data processing module, and FIFO.

The input channel selector is used to select the channel to be sampled, including internal and external analog signals.

The programmable amplifier is used to further process the input signal, and it is set based on the characteristics of the input signal (DC/AC) to get a more accurate conversion value.

The ADC sampling module is the most important functional module, which gets the result of conversion from analog signal to digital signal by successive comparison.

The result of the conversion is 12/14/16 bit, and the data processing module further processes the conversion result, including adding channel information, etc.

The final data will be pushed to the back-end FIFO.

5.3.1 ADC Pins and Internal Signals

Table 5.1: ADC internal signal

Internal Signal	Type	Description
VBAT/2	Input	Voltage signal divided from the power supply pin
TSEN	Input	Output voltage of internal temperature sensor
VREF	Input	Reference voltage of internal analog module
DACOUTA	Input	DAC module output
DACOUTB	Input	DAC module output

Table 5.2: ADC external pin

External Pin	Signal Type	Signal Description
VDDA	Input	Positive supply voltage of analog module
VSSA	Input	Analog module ground
ADC_CHX	Input	Analog input pins, 12 channels

5.3.2 ADC Channels

The selectable channels of ADC sampling include the input signals of external analog pins and the selectable signals inside the chip:

- ADC CH0
- ADC CH1
- ADC CH2
- ADC CH3
- ADC CH4
- ADC CH5
- ADC CH6
- ADC CH7

- ADC CH8
- ADC CH9
- ADC CH10
- ADC CH11
- DAC OUTA
- DAC OUTB
- VBAT/2
- TSEN
- VREF
- GND

It is worth noting that if VBAT/2 or TSEN is selected as the input signal to be sampled, gpadc_vbat_en or gpadc_ts_en shall be set. The ADC module supports single-ended input or differential input. For the former, GND shall be selected as the negative input channel.

5.3.3 ADC Clocks

The following figure illustrates the working clock source of the ADC module.

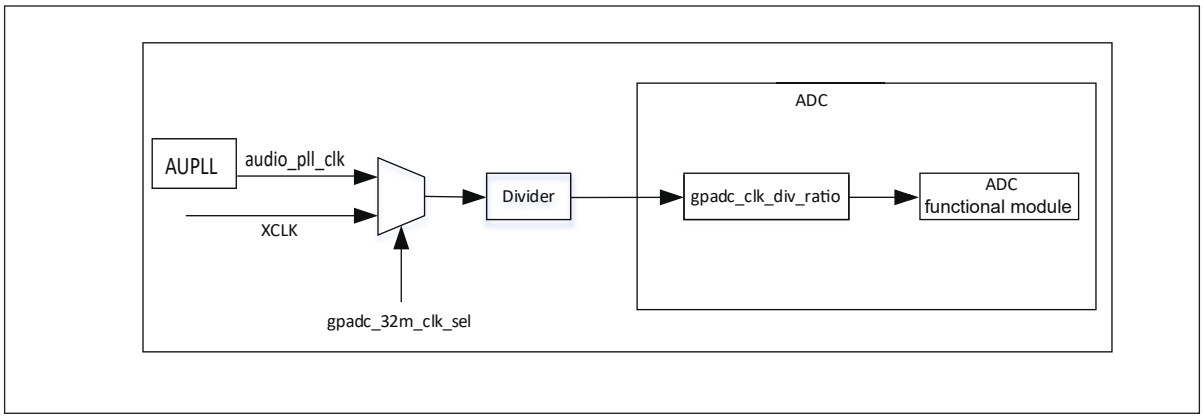


Fig. 5.2: ADC Clocks

The clock source of ADC can be the audio_pll_clk from AUPLL, XTAL or internal RC32M. The GLB module controls the setting of clock source selection and provides clock frequency division. For example: the clock source of ADC is XTAL and the clock frequency division is 1. The clock reaching the ADC module is 20M.

Inside the ADC module, a clock divider is provided. If you choose to divide by 20, the clock inside the ADC module is 1M. Users can adjust the ADC clock source and various frequency division coefficients by themselves according to the actual sampling requirements.

The gpadc_32m_clk_div frequency division register is 6 bits wide (max div=64). The frequency division formula is fout=fsource/(gpadc_32m_clk_div+1). The gpadc_clk_div_ratio frequency division register of 3 bits width is located inside the ADC module. Its frequency division values are defined as follows:

- 3’b000: div=1
- 3’b001: div=4
- 3’b010: div=8
- 3’b011: div=12
- 3’b100: div=16
- 3’b101: div=20
- 3’b110: div=24
- 3’b111: div=32

5.3.4 ADC Conversion Mode

The ADC supports single-channel conversion and scan conversion. In single-channel conversion mode, the user needs to select the positive input channel through gpadc_pos_sel, select the negative input channel through gpadc_neg_sel, and set the gpadc_scan_en control bit to 0, indicating single-channel conversion, then set the gpadc_conv_start control bit to start the conversion.

In the scanning conversion mode, the gpadc_scan_en control bit is set to 1. ADC performs conversion one by one according to the number of conversion channels set by the gpadc_scan_length control bit and the channel sequence set by gpadc_reg_scn_posX (X = 1, 2) and gpadc_reg_scn_negX (X = 1, 2) register sets. The conversion result is automatically pushed into the ADC FIFO. The channels set by the gpadc_reg_scn_posX (X = 1, 2) and gpadc_reg_scn_negX (X = 1, 2) register sets can be the same, which means that the user can sample a channel multiple times for conversion.

5.3.5 ADC Result

The register gpadc_raw_data stores the original result of ADC. In the single-ended mode, the valid bit of data is 12 bits, without sign bit. In the differential mode, the most significant bit (MSB) is the sign bit, and the remaining 11 bits represent the conversion result.

The register gpadc_data_out stores the ADC result, which contains the ADC result, sign bit, and channel information. The data format is as follows:

Table 5.3: Meaning of ADC Conversion Result

BitS	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
meaning	Positive channel number					Negative channel number					Conversion result															

In the above table, bit21-bit25 indicates the positive channel number, and bit16-bit20 indicates the negative channel number, while bit0-bit15 indicates the converted value.

The gpadc_res_sel control bit can set the bits of the conversion result to 12 bits, 14 bits, and 16 bits, of which 14 bits and 16 bits are the result of multiple sampling. The values and the number of samples that can be set are as follows (2M sampling clock as an example, non-single-channel continuous conversion mode requires a lower clock):

- 3'b000 12bit 2MS/s, OSR=1
- 3'b001 14bit 125kS/s, OSR=16
- 3'b010 14bit 31.25kS/s, OSR=64
- 3'b011 16bit 15.625KS/s, OSR=128
- 3'b100 16bit 7.8125KS/s, OSR=256

In the left-justified ADC conversion result, when 12 bits are selected, bit15-bit4 of the conversion result is valid; when 14 bits are selected, bit15-bit2 is valid; and when 16 bits are selected, bit15-bit0 is valid. Similarly, in the differential mode, the MSB is the sign bit. Namely, when 14 bits are selected, bit15 is the sign bit and bit14 is the MSB, while bit14-bit2 is the conversion result. In the single-ended mode, there is no sign bit. Namely, when 12 bits are selected, bit15-bit4 is the conversion result and bit15 is the MSB.

In practice, ADC results are generally pushed into FIFO, which is especially important in the multi-channel scanning mode. Therefore, users usually obtain conversion results from ADC FIFO. The data format of ADC FIFO is the same as that in the register gpadc_data_out.

5.3.6 ADC Interrupt

- ADC conversion completion interrupt * When the ADC conversion is completed and the result is stored in the FIFO, set the interrupt switch via gpadc_rdy_mask to select whether to trigger the ADC conversion completion interrupt.
- ADC positive (negative) sampling over-range interrupt * When the ADC is in positive sampling over-range and negative sampling over-range, set the interrupt switch via gpadc_pos_satur_mask, gpadc_neg_satur_mask to select whether to trigger the interrupt or not, when the interrupt is generated, you can query the interrupt via gpadc_pos_satur and gpadc_neg_satur registers. The interrupt status can be cleared by setting gpadc_pos_-satur_clr and gpadc_neg_satur_clr. This function can be used to determine if the input voltage is abnormal.

5.3.7 ADC FIFO

The ADC module has a FIFO with a depth of 32 and its data width is 26 bits. When the ADC completes conversion, the result will be automatically pushed into the FIFO. ADC FIFO has the following statuses and interrupt management functions:

- FIFO: full
- FIFO: non-empty
- FIFO Overrun interrupt
- FIFO Underrun interrupt

When an interrupt is generated, the interrupt flag can be cleared by the corresponding clear bit.

ADC FIFO enables users to obtain data through query, interrupt, and DMA modes.

Query mode

CPU polls the gpadc_rdy bit: When the control bit is set, it indicates that there are valid data in FIFO. CPU can get the amount of FIFO data through gpadc_fifo_data_count and read out these data from FIFO.

Interrupt mode

If gpadc_rdy_mask is set to 0 in CPU, ADC will generate an interrupt when data is pushed into FIFO. In the interrupt function, the user can get the amount of FIFO data through gpadc_fifo_data_count and read out these data from FIFO, and then set gpadc_rdy_clr to clear the interrupt.

DMA mode

If the user sets the gpadc_dma_en control bit, DMA can transfer the conversion data to the memory. In the DMA mode, after the data volume threshold is set for ADC FIFO to send DMA requests through gpadc_fifo_thl, when receiving a request, DMA will automatically transfer the specified number of results from FIFO to the corresponding memory according to the user defined parameters.

5.3.8 ADC Setup Process

Set ADC clock

According to the required ADC conversion speed, determine the working clock of ADC, set the ADC clock source and frequency division of the GLB module, and determine the final working clock frequency of the ADC module based on the gpadc_clk_div_ratio.

Set GPIO according to the channel used

According to the analog pin used, determine the channel number used, and initialize the corresponding GPIO to analog function. But note that when setting GPIO as analog input, set it as floating input instead of pull-up or pull-down.

Set the channel to convert

According to the used analog channel and conversion mode, set the corresponding channel register. For single channel conversion, set the conversion channel information in the registers gpadc_pos_sel and gpadc_neg_sel. In the multi-channel scanning mode, set gpadc_scan_length, gpadc_reg_scn_posX, and gpadc_reg_scn_negX according to the number of channels to be scanned and the scanning sequence.

Set the data read method

Based on the data reading mode introduced by ADC FIFO, select the mode used and set the register. If DMA is used, configure one channel of DMA, and assist ADC FIFO in transferring data.

Start conversion

Finally, set gpadc_res_sel to select the precision of data conversion result, and then set gpadc_global_en=1 and gpadc_conv_start=1 to start ADC conversion. When conversion is completed, to convert again, it is necessary to set gpadc_conv_start to 0 first and then to 1 to trigger the conversion again.

5.3.9 VBAT Measurement

The VBAT/2 here measures the voltage of the chip VDD33, not the external lithium battery voltage. If you need to measure the voltage of power supply sources like lithium battery, you can divide the voltage and input it into the GPIO analog channel of ADC. Measuring the voltage of VDD33 can reduce the use of GPIO.

The VBAT/2 voltage measured by the ADC module is divided, and the actual voltage input to the ADC module is half of that of VDD33, namely VBAT/2=VDD33/2. As the voltage is divided, to ensure a high accuracy, it is suggested to select 2.0 V reference voltage for ADC, and enable the single-ended mode. The positive input voltage is set to VBAT/2 and the negative input voltage is set to GND. gpadc_vbat_en is set to 1. After conversion starts, the corresponding conversion result can be multiplied by 2 to get the voltage of VDD33.

5.3.10 TSEN Measurement

ADC can measure the voltage of internal or external diode. As the voltage difference of diode is related to temperature, the ambient temperature can be calculated from the measured voltage of diode. So it is called temperature sensor (TSEN).

The measurement principle of TSEN is the curve fitted by the voltage difference (ΔV) with the change of temperature, where ΔV is produced by measuring two different currents on a diode. No matter an external or internal diode is measured, the final output value is related to temperature and can be expressed as Δ(ADC_out)=7.753T+X. As long as we know the voltage, we know the temperature T. X indicates an offset value, which can be used as a standard value. Before actual use, X shall be determined first. The chip manufacturer will measure Δ(ADC_out) at a standard temperature, such as 25°C room temperature, before the chip leaves the factory, to obtain X. In actual use, the user can get the temperature T according to the formula T=[Δ(ADC_out)X]/7.753.

When TSEN is used, it is recommended to set ADC to the 16bits mode, reduce error through multiple sampling, select 2.0 V reference voltage to improve accuracy, and set gpadc_ts_en to 1 to enable the TSEN function.

If internal diode is selected, gpadc_tsext_sel=0. If external diode is selected, gpadc_tsext_sel=1. The positive input

channel is selected according to actual needs, namely TSEN channel for the internal diode or the analog GPIO channel for the external diode. The negative input channel is set to GND.

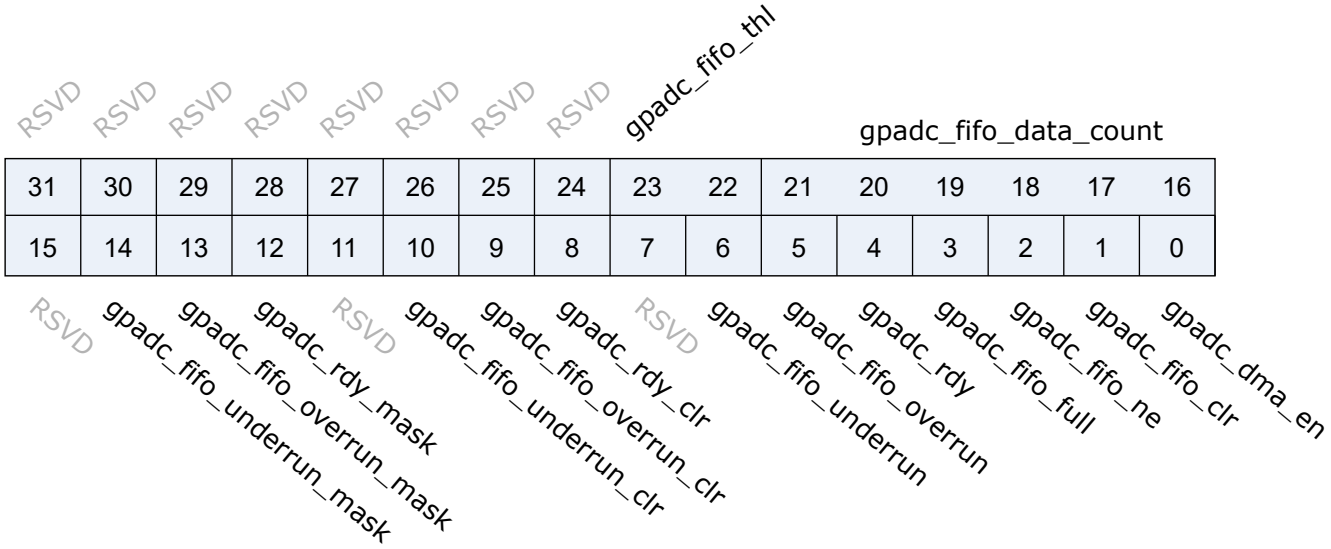
After finishing the above settings, set gpadc_tsvbe_low=0 to start measurement, and get the measurement result V0. Then set gpadc_tsvbe_low=1 to start measurement, and get the measurement result V1, Δ(ADC_out)=V1V0. The temperature T is calculated based on the formula T=[Δ(ADC_out)X]/7.753.

5.4 Register description

Name	Description
gpadc_config	gpadc fifo config
gpadc_dma_rdata	gpadc fifo
gpadc_reg_cmd	gpadc config0
gpadc_reg_config1	gpadc config1
gpadc_reg_config2	gpadc config2
gpadc_reg_scn_pos1	gpadc positive converation sequence 1
gpadc_reg_scn_pos2	gpadc positive converation sequence 2
gpadc_reg_scn_neg1	gpadc negative converation sequence 1
gpadc_reg_scn_neg2	gpadc negative converation sequence 2
gpadc_reg_status	gpadc ready status
gpadc_reg_isr	gpadc saturation interrupt config

5.4.1 gpadc_config

Address：0x20002000

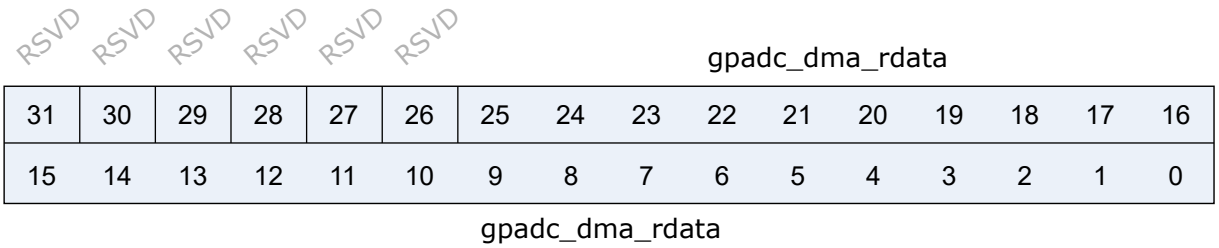


Bits	Name	Type	Reset	Description
31:24	RSVD			
23:22	gpadc_fifo_thl	r/w	2'd0	fifo threshold 2'b00: 1 data 2'b01: 4 data 2'b10: 8 data 2'b11: 16 data
21:16	gpadc_fifo_data_count	r	6'd0	fifo data number
15	RSVD			
14	gpadc_fifo_underrun_mask	r/w	1'b0	write 1 mask
13	gpadc_fifo_overrun_mask	r/w	1'b0	write 1 mask
12	gpadc_rdy_mask	r/w	1'b0	write 1 mask
11	RSVD			
10	gpadc_fifo_underrun_clr	r/w	1'b0	Write 1 to clear flag
9	gpadc_fifo_overrun_clr	r/w	1'b0	Write 1 to clear flag
8	gpadc_rdy_clr	r/w	1'b0	Write 1 to clear flag
7	RSVD			
6	gpadc_fifo_underrun	r	1'b0	FIFO underrun interrupt flag
5	gpadc_fifo_overrun	r	1'b0	FIFO overrun interrupt flag
4	gpadc_rdy	r	1'b0	Conversion data ready interrupt flag
3	gpadc_fifo_full	r	1'b0	FIFO full flag
2	gpadc_fifo_ne	r	1'b0	FIFO not empty flag
1	gpadc_fifo_clr	w1c	1'b0	FIFO clear signal

Bits	Name	Type	Reset	Description
0	gpadc_dma_en	r/w	1'b0	GPADC DMA enbale

5.4.2 gpadc_dma_rdata

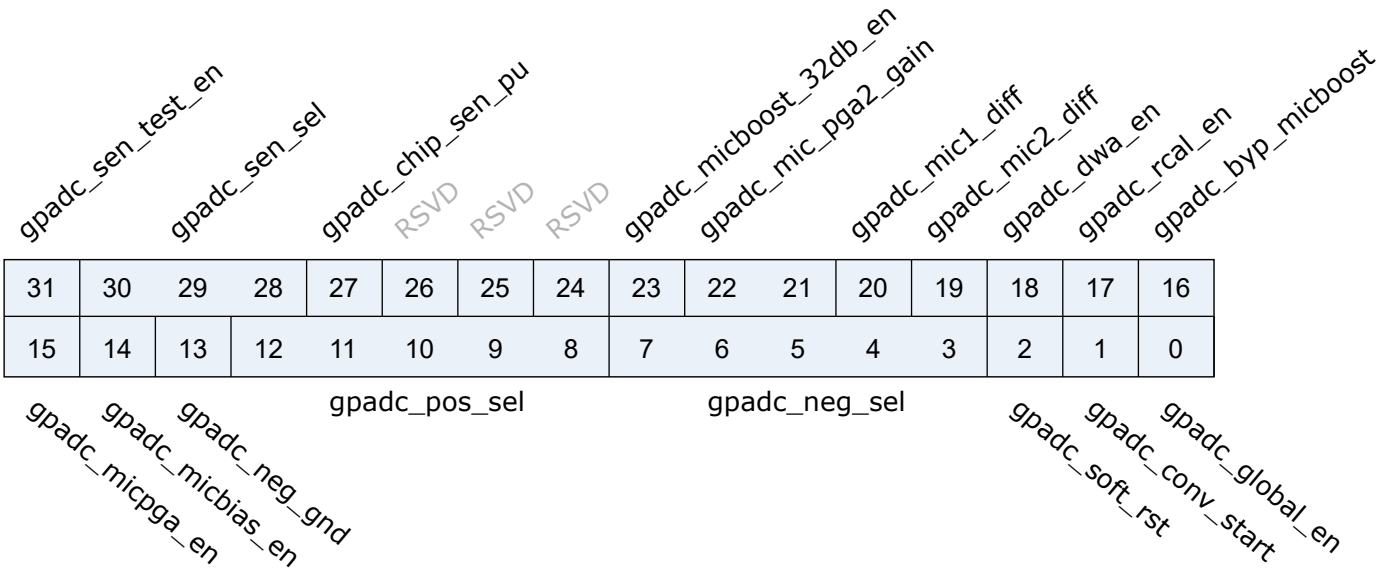
Address：0x20002004



Bits	Name	Type	Reset	Description
31:26	RSVD			
25:0	gpadc_dma_rdata	r	26'd0	GPADC finial conversion result stored in the FIFO

5.4.3 gpadc_reg_cmd

Address：0x2000f90c



Bits	Name	Type	Reset	Description
31	gpadc_sen_test_en	r/w	1'b0	enable sensor dc test mux

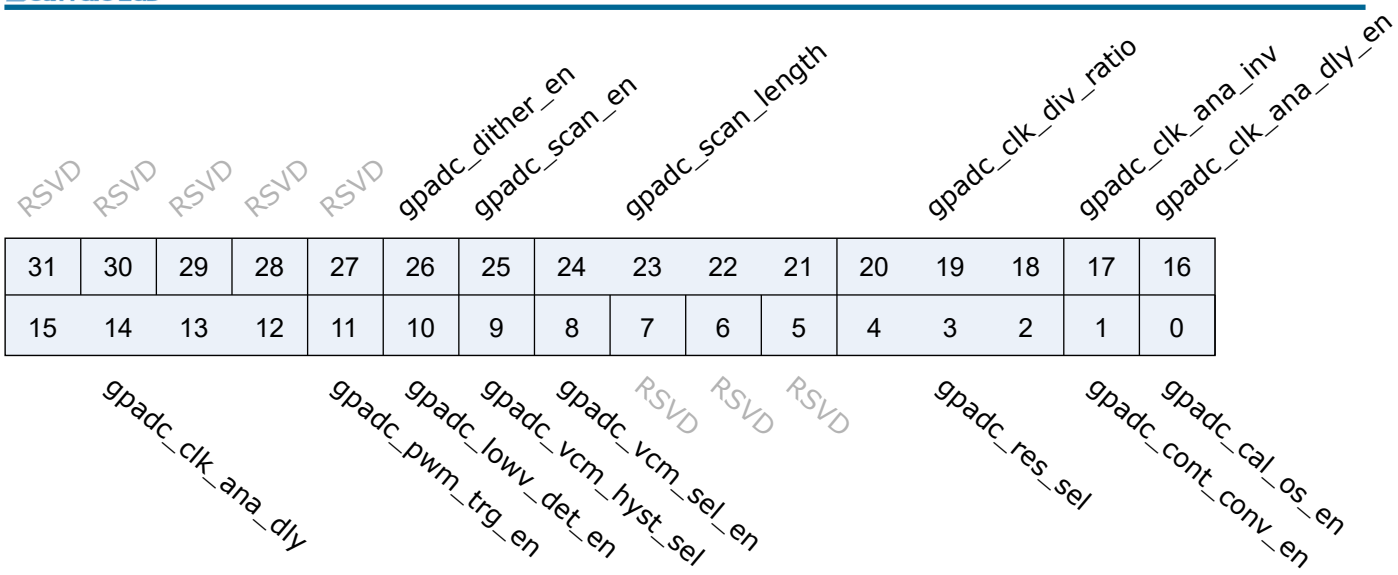
Bits	Name	Type	Reset	Description
30:28	gpadc_sen_sel	r/w	3'h0	selected output current channel and measurement channel 2'h0: 1st channel 2'h1: 2nd channel 2'h2: 3rd channel 2'h3: 4th channel
27	gpadc_chip_sen_pu	r/w	1'b0	enable chip sensor test 1'b0: disable 1'b1: enable
26:24	RSVD			
23	gpadc_micboost_32db_en	r/w	1'b0	micboost 32db enable 1'b0: 16dB 1'b1: 32dB
22:21	gpadc_mic_pga2_gain	r/w	2'h0	mic_pga2_gain 2'h0: 0dB 2'h1: 6dB 2'h2: -6dB 2'h3: 12dB
20	gpadc_mic1_diff	r/w	1'b0	mic1 diff enable 1'b0: single 1'b1: diff
19	gpadc_mic2_diff	r/w	1'b0	mic2 diff enable 1'b0: single 1'b1: diff
18	gpadc_dwa_en	r/w	1'b0	dwa enable 1'b0: dwa disable 1'b1: dwa enable
17	gpadc_rcal_en	r/w	1'b0	rcal enable 1'b0: rcal disable 1'b1: rcal enable
16	gpadc_byp_micboost	r/w	1'b0	micboost amp bypass 1'b0: not bypass 1'b1: bypass
15	gpadc_micpga_en	r/w	1'b0	micpga enable 1'b0: micpga disable 1'b1: miapga enable
14	gpadc_micbias_en	r/w	1'b0	enable micbias 1'b0: micbias power down 1'b1: miabias power on

Bits	Name	Type	Reset	Description
13	gpadc_neg_gnd	r/w	1'b0	set negative input of adc to ground 1'b0: disable 1'b1: enable
12:8	gpadc_pos_sel	r/w	5'hf	select adc positive input in none-scan mode 5‘h0 gpio0 5'h1 gpio1 5'h2 gpio2 5‘h3 gpio3 5'h4 gpio4 5'h5 gpio5 5‘h6 gpio6 5'h7 gpio7 5'h8 gpio8 5‘h9 gpio9 5'h10 gpio10 5'h11 gpio11 5‘h12 daca 5'h13 dacb 5'h14 temp_p 5‘h15 temp_n 5'h16 vref 5'h17 atest 5‘h18 vbat/2 5'h19 vp3_diode 5'h20 vp2_diode 5‘h21 vp1_diode 5'h22 vp0_diode 5'h23 31 avss

Bits	Name	Type	Reset	Description
7:3	gpadc_neg_sel	r/w	5'hf	select adc negative input in none-scan mode 5‘h0 gpio0 5'h1 gpio1 5'h2 gpio2 5‘h3 gpio3 5'h4 gpio4 5'h5 gpio5 5‘h6 gpio6 5'h7 gpio7 5'h8 gpio8 5‘h9 gpio9 5'h10 gpio10 5'h11 gpio11 5‘h12 daca 5'h13 dacb 5'h14 temp_p 5‘h15 temp_n 5'h16 vref 5'h17 atest 5‘h18 vbat/2 5'h19 vn3_diode 5'h20 vn2_diode 5‘h21 vn1_diode 5'h22 vn0_diode 5'h23 31 avss
2	gpadc_soft_rst	r/w	1'b0	user reset the whole block 1'h0: not reset 1'h1: reset
1	gpadc_conv_start	r/w	1'b0	1'h0: stop converation 1'h1: start converation
0	gpadc_global_en	r/w	1'b0	1'h0: disable ADC 1'h1: enable ADC

5.4.4 gpadc_reg_config1

Address：0x2000f910



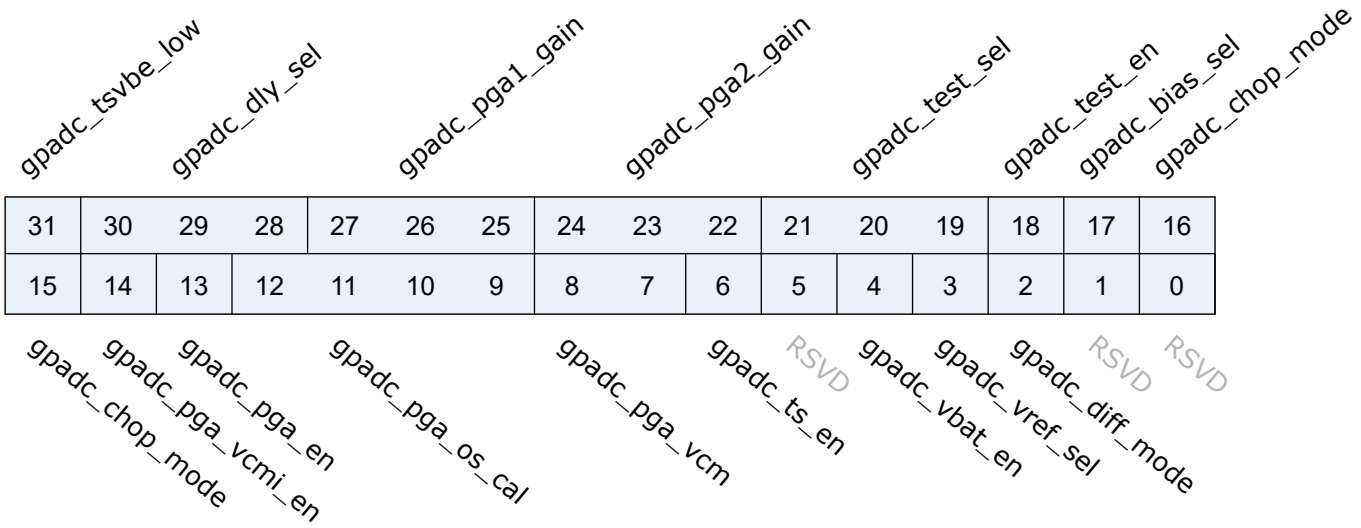
Bits	Name	Type	Reset	Description
31	RSVD			
30:29	gpadc_v18_sel	r/w	2'h0	internal vdd18 select
28:27	gpadc_v11_sel	r/w	2'h0	internal vdd11 select
26	gpadc_dither_en	r/w	1'h0	Dither compensation enable
25	gpadc_scan_en	r/w	1'h0	select scan mode enable: 0: select gpadc_pos/neg_sel;1: select : select gpadc_scan_pos_x and gpadc_scan_neg_x

Bits	Name	Type	Reset	Description
24:21	gpadc_scan_length	r/w	4'h0	select scan mode length 4'b0000 : select gpadc_scan_pos_0 and gpadc_scan_neg_0 4'b0001 : select gpadc_scan_pos_1 and gpadc_scan_neg_1 4'b0010 : select gpadc_scan_pos_2 and gpadc_scan_neg_2 4'b0011 : select gpadc_scan_pos_3 and gpadc_scan_neg_3 4'b0100 : select gpadc_scan_pos_4 and gpadc_scan_neg_4 4'b0101 : select gpadc_scan_pos_5 and gpadc_scan_neg_5 4'b0110 : select gpadc_scan_pos_6 and gpadc_scan_neg_6 4'b0111 : select gpadc_scan_pos_7 and gpadc_scan_neg_7 4'b1000 : select gpadc_scan_pos_8 and gpadc_scan_neg_8 4'b1001 : select gpadc_scan_pos_9 and gpadc_scan_neg_9 4'b1010 : select gpadc_scan_pos_10 and gpadc_scan_neg_10 4'b1011 : select gpadc_scan_pos_11 and gpadc_scan_neg_11
20:18	gpadc_clk_div_ratio	r/w	3'h3	analog 32M clock division ratio 3'b000: div=1 3'b001: div=4 3'b010: div=8 3'b011: div=12 3'b100: div=16 3'b101: div=20 3'b110: div=24 3'b111: div=32
17	gpadc_clk_ana_inv	r/w	1'b0	analog clock 2M inverted
16	gpadc_clk_ana_dly_en	r/w	1'b0	analog clock 2M delay enable
15:12	gpadc_clk_ana_dly	r/w	4'd0	analog clock 2M delay cycle count
11	gpadc_pwm_trg_en	r/w	1'b0	Enable signal for PWM to trigger ADC sampling
10	gpadc_lowv_det_en	r/w	1'b0	Low power supply detected enable
9	gpadc_vcm_hyst_sel	r/w	1'b0	pga vcm hystersis select when vcm_sel_en is enabled

Bits	Name	Type	Reset	Description
8	gpadc_vcm_sel_en	r/w	1'b0	pga vcm selected when lowv_det_en is enable
7:5	RSVD			
4:2	gpadc_res_sel	r/w	3'h0	adc resolution/over-sample rate select 3'b000 12bit 2MS/s, OSR=1 3'b001 14bit 125kS/s, OSR=16 3'b010 14bit 31.25kS/s, OSR=64 3'b011 16bit 15.625KS/s, OSR=128 (voice mode16KS/s) 3'b100 16bit 7.8125KS/s, OSR=256 (voice mode 8KS/s)
1	gpadc_cont_conv_en	r/w	1'b1	To enable continuous conversion 1'h0: one shot conversion 1'h1: continuous conversion
0	gpadc_cal_os_en	r/w	1'b0	offset calibration enable

5.4.5 gpadc_reg_config2

Address：0x2000f914



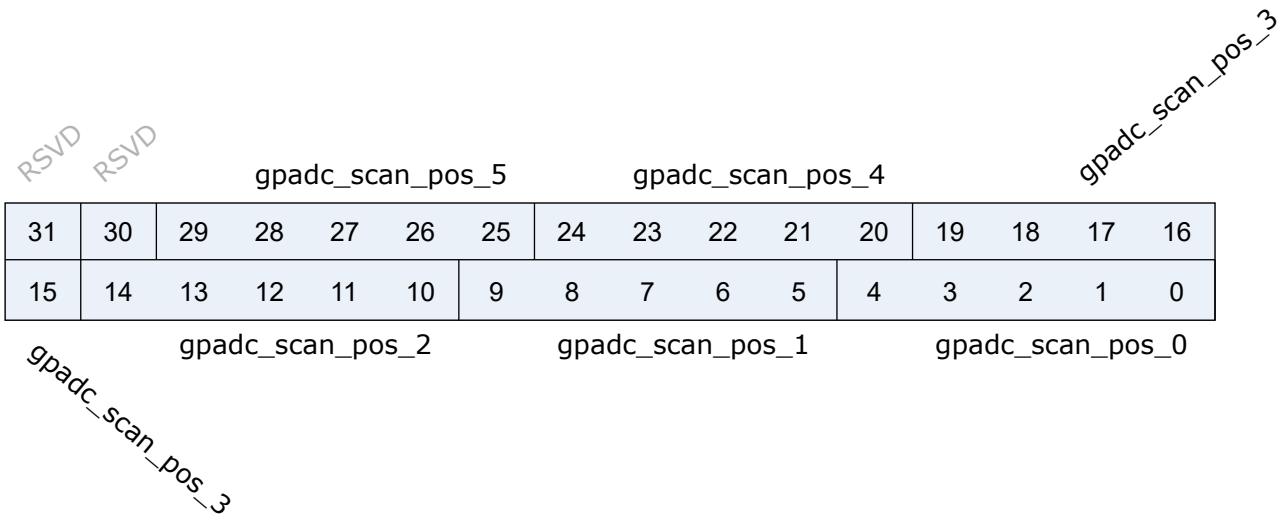
Bits	Name	Type	Reset	Description
31	gpadc_tsvbe_low	r/w	1'b0	tsen diode current
30:28	gpadc_dly_sel	r/w	3'h0	adc conversion speed

Bits	Name	Type	Reset	Description
27:25	gpadc_pga1_gain	r/w	3'h0	3'h0: disable 3'h1: gain=1 3'h2: gain=2 3'h3: gain=4 3'h4: gain=8 3'h5: gain=16 3'h6: gain=32 3'h7: gain=32
24:22	gpadc_pga2_gain	r/w	3'h0	3'h0: disable 3'h1: gain=1 3'h2: gain=2 3'h3: gain=4 3'h4: gain=8 3'h5: gain=16 3'h6: gain=32 3'h7: gain=32
21:19	gpadc_test_sel	r/w	3'h0	select test point 0 7
18	gpadc_test_en	r/w	1'b0	Analog test enable.
17	gpadc_bias_sel	r/w	1'b0	adc analog portion low power mode select 1'h0: bandgap system 1'h1:aon bandgap
16:15	gpadc_chop_mode	r/w	2'h3	2'b11 all off 2'b11 Vref AZ on 2'b11 Vref AZ and PGA chop on 2'b11 Vref AZ and PGA chop+RPC on
14	gpadc_pga_vcmi_en	r/w	1'b0	enable pga input vcm bias
13	gpadc_pga_en	r/w	1'b0	1'h0: disable PGA 1'h1 enable PGA
12:9	gpadc_pga_os_cal	r/w	4'h8	pga offset calibration
8:7	gpadc_pga_vcm	r/w	2'h2	Audio PGA output common mode control 2'b00: cm=1.3V 2'b11: cm=1.4V 2'b11: cm=1.5V 2'b11: cm=1.6V
6	gpadc_ts_en	r/w	1'b0	1'h0: disable temperature sensor 1'h1: enable temperature sensor
5	gpadc_tsext_sel	r/w	1'b0	1'h0: internal diode mode 1'h1: external diode mode
4	gpadc_vbat_en	r/w	1'b0	1'h0: disable VBAT sensor 1'h1 enable VBAT sensor

Bits	Name	Type	Reset	Description
3	gpadc_vref_sel	r/w	1'b0	ADC reference select 1'h0 3.2V 1'h1 2.0V
2	gpadc_diff_mode	r/w	1'b0	1'h0 single-ended 1'h1 differential
1:0	RSVD			

5.4.6 gpadc_reg_scn_pos1

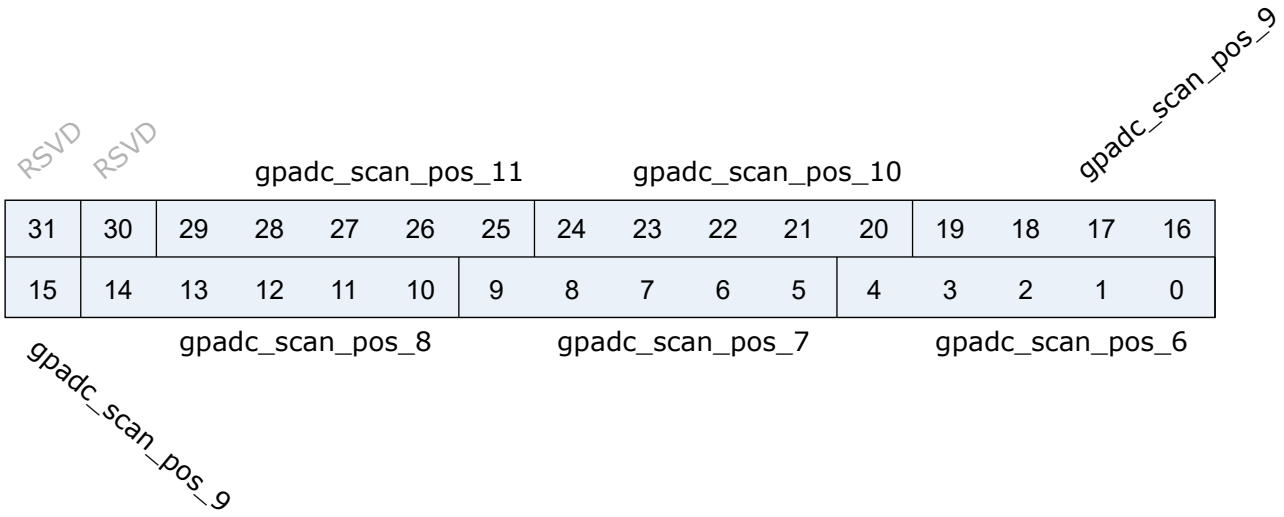
Address：0x2000f918



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:25	gpadc_scan_pos_5	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
24:20	gpadc_scan_pos_4	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
19:15	gpadc_scan_pos_3	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
14:10	gpadc_scan_pos_2	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
9:5	gpadc_scan_pos_1	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
4:0	gpadc_scan_pos_0	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel

5.4.7 gpadc_reg_scn_pos2

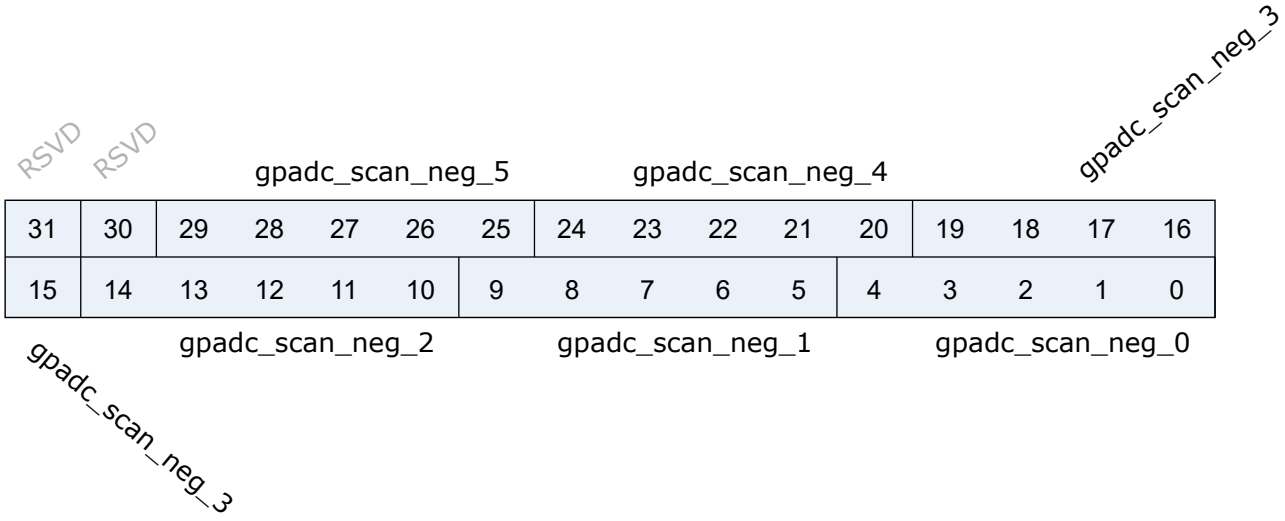
Address：0x2000f91c



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:25	gpadc_scan_pos_11	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
24:20	gpadc_scan_pos_10	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
19:15	gpadc_scan_pos_9	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
14:10	gpadc_scan_pos_8	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
9:5	gpadc_scan_pos_7	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel
4:0	gpadc_scan_pos_6	r/w	5'hf	definition is the same as adc_reg_cmd.adc_pos_sel

5.4.8 gpadc_reg_scn_neg1

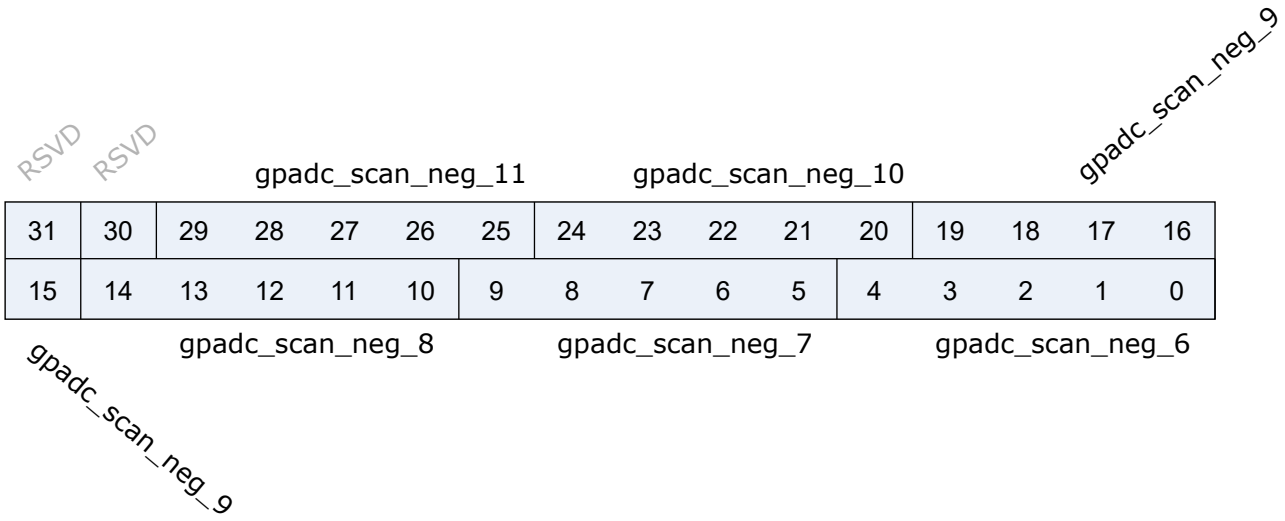
Address：0x2000f920



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:25	gpadc_scan_neg_5	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
24:20	gpadc_scan_neg_4	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
19:15	gpadc_scan_neg_3	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
14:10	gpadc_scan_neg_2	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
9:5	gpadc_scan_neg_1	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
4:0	gpadc_scan_neg_0	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel

5.4.9 gpadc_reg_scn_neg2

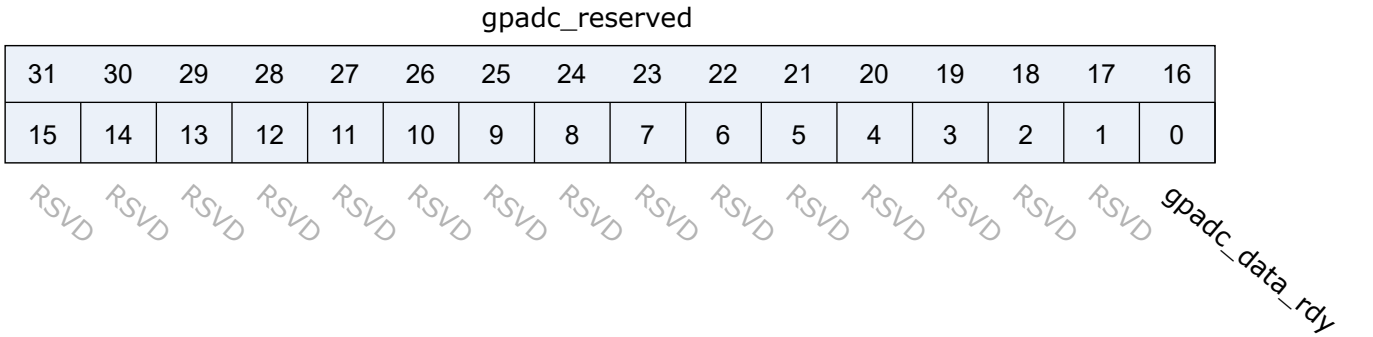
Address：0x2000f924



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:25	gpadc_scan_neg_11	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
24:20	gpadc_scan_neg_10	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
19:15	gpadc_scan_neg_9	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
14:10	gpadc_scan_neg_8	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
9:5	gpadc_scan_neg_7	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel
4:0	gpadc_scan_neg_6	r/w	5'hf	definition is the same as adc_reg_cmd.adc_neg_sel

5.4.10 gpadc_reg_status

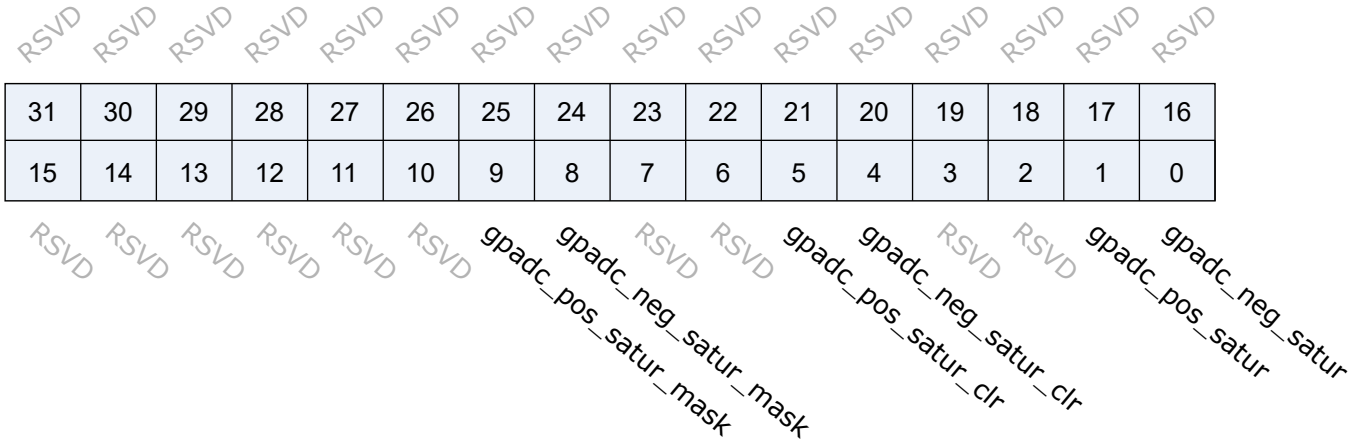
Address：0x2000f928



Bits	Name	Type	Reset	Description
31:16	gpadc_reserved	r/w	16'h0	
15:1	RSVD			
0	gpadc_data_rdy	r	1'b0	ADC final conversion data ready

5.4.11 gpadc_reg_isr

Address：0x2000f92c



Bits	Name	Type	Reset	Description
31:10	RSVD			
9	gpadc_pos_satur_mask	r/w	1'h0	write 1 mask
8	gpadc_neg_satur_mask	r/w	1'h0	write 1 mask
7:6	RSVD			
5	gpadc_pos_satur_clr	r/w	1'b0	Write 1 to clear flag
4	gpadc_neg_satur_clr	r/w	1'b0	Write 1 to clear flag
3:2	RSVD			
1	gpadc_pos_satur	r	1'b0	ADC data positive side saturation interrupt flag
0	gpadc_neg_satur	r	1'b0	ADC data negative side saturation interrupt flag

6 DAC

6.1 Overview

The DAC module is a 12-bit voltage output digital-to-analog converter that works with a DMA controller. The built-in DAC module of the chip has two output channels, and each channel has an independent converter, which can perform digital-to-analog conversion independently of each other. In addition, the converter of this DAC can also be used as the analog output channel of AudioDAC. Can be used for audio playback, transmitter voltage modulation and other applications.

6.2 Features

- DAC modulation accuracy is 12-bits
- The digital-to-analog converter of the DAC can be used as the analog output channel of the Audio DAC module
- DAC input clock can be selected as 32MHz, xclk or from AudioDAC module
- Each channel has DMA function and supports 10 data transfer formats
- Support DAC dual-channel simultaneous conversion
- The output pin of DAC is fixed as ChannelA is GPIO3, ChannelB is GPIO2
- Supports internal and external input reference voltages

6.3 Functional Description

The block diagram of DAC is shown as follows.

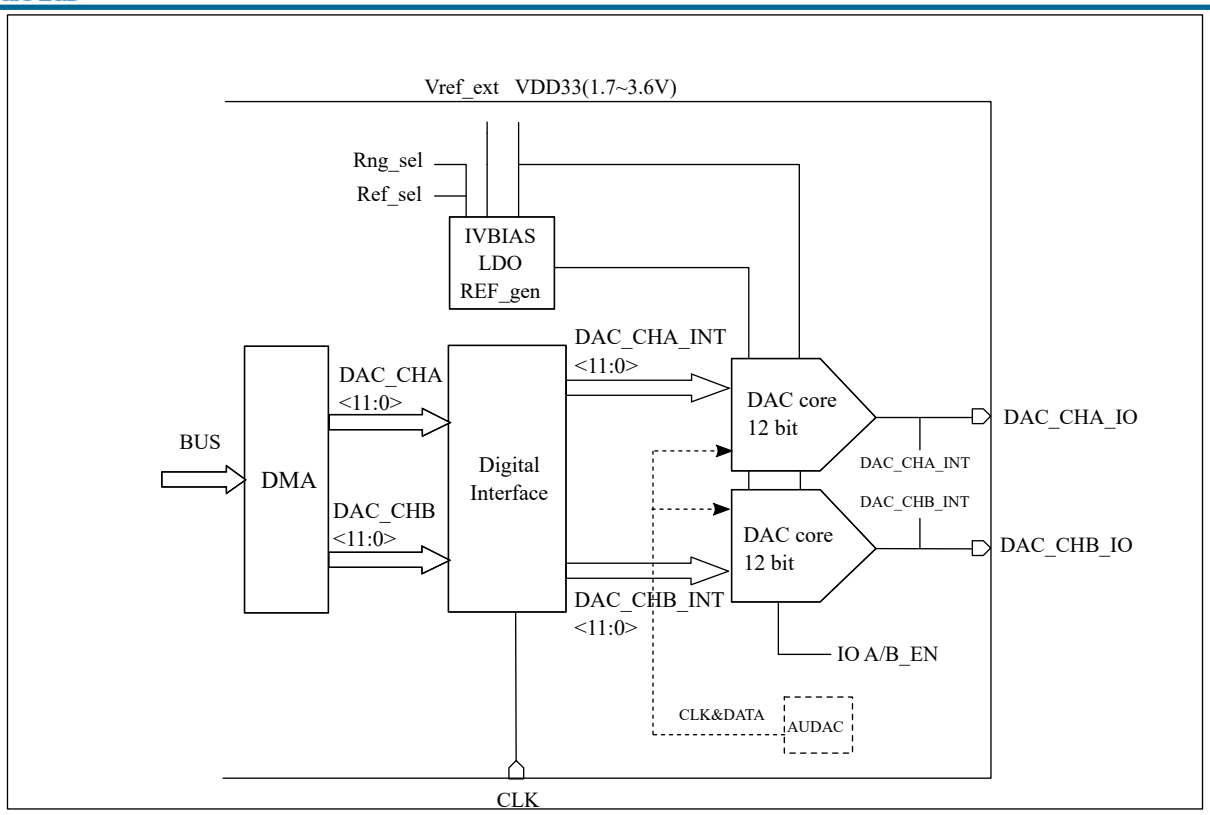


Fig. 6.1: Block Diagram of DAC

The DAC module contains two DAC analog-to-digital conversion circuits and the power circuit related to the modulated analog signal. The user can select whether the reference voltage of the DAC is external or internal through Ref_Sel, and select the output voltage range through Rng_Sel. The clock and data of the DAC digital-to-analog converter can come from its own digital control circuit (independent mode), or from the output of the AudioDAC in analog mode (combined mode).

When the DAC works in independent mode, the DAC modulation registers (gpdac_a_data, gpdac_b_data in the register dac_cfg3) can be directly written by the CPU, or transferred to the gpdac_dma_wdata register by the DMA.

When the DAC is in joint mode, its own FIFO, data format, clock configuration and other digital circuit configurations will be bypassed and no longer take effect, and the clock and data of the digital-to-analog converter will be taken over by the AudioDAC module.

6.3.1 DAC channel enable

Taking channel A as an example in independent mode, the configuration process is as follows:

1. Set the corresponding gpdac_en bit in register gpdac_config to 1 to enable the DAC
2. Write 0 to the corresponding gpdac_ana_clk_sel and gpdac_dat_cha_sel bits in register dac_cfg0 to work in standalone mode
3. Set the corresponding gpdac_a_en bit in register dac_cfg1 to 1 to enable DAC A channel conversion

4. Set the corresponding gpdac_ioa_en bit in the register dac_cfg1 to 1, and enable channel A conversion result to GPIO port

6.3.2 DAC data format

When using DMA to convert data, there are a total of 10 data transmission formats, which can be configured by setting the corresponding gpdac_dma_format bit value in the register gpdac_dma_config. The data transmission format stored in the gpdac_dma_wdata register (with a width of 32-bit), the corresponding relationship is as follows:

Table 6.1: Data transfer format

Value of gpdac_dma_format	gpdac_dma_wdata(Corresponding to the data transmission format of A and B channels)
0	[11:0]: {A0}, {A1}, {A2} •••
1	[27:16][11:0]: {B0,A0}, {B1,A1}, {B2,A2} •••
2	[27:16][11:0]: {A1,A0}, {A3,A2}, {A5,A4} •••
4	[15:4]: {A0}, {A1}, {A2} •••
5	[31:20][15:4]: {B0,A0}, {B1,A1}, {B2,A2} •••
6	[31:20][15:4]: {A1,A0}, {A3,A2}, {A5,A4} •••
8	[31:24][23:16][15:8][7:0]: {A3,A2,A1,A0}, {A7,A6,A5,A4} •••
9	[31:24][23:16][15:8][7:0]: {B1,B0,A1,A0}, {B3,B2,A3,A2} •••
10	[31:24][23:16][15:8][7:0]: {B1,A1,B0,A0}, {B3,A3,B2,A2} •••
11	[15:8][7:0]: {B0,A0}, {B1,A1}, {B2,A2} •••

6.3.3 DAC output voltage

The user can choose to use the external reference voltage or the internal reference voltage by setting the corresponding gpdac_ref_sel bit value in the dac_cfg0 register.

If the internal reference voltage is selected, the configuration and output voltage are shown in the table below. If external reference voltage is selected, connect the external voltage to fixed GPIO28.

Table 6.2: Internal reference voltage output voltage

gpdac_a_rng	gpdac_ref_sel	Output range (V)
00	0	0.2-1
01/10	0	0.225-1.425
11	0	0.2-1.8

6.3.4 DAC conversion

6.3.4.1 CPU mode in independent mode

In the independent mode, the CPU is used to carry the data for conversion. Taking the simultaneous conversion of channel A and channel B as an example, the configuration process is as follows:

1. Set the DAC clock: write 1 to the corresponding bit value of dig_clk_src_sel in the register dig_clk_cfg0, that is, select xclk as the DAC clock source
2. Write 0 to the corresponding gpdac_ana_clk_sel, gpdac_dat_chb_sel and gpdac_dat_cha_sel bits in register dac_cfg0, the data and clock come from itself, and work in independent mode.
3. Set the clock frequency division factor: the user sets the value of the dig_512k_div bit corresponding to the register dig_clk_cfg0 and the gpdac_mode bit corresponding to the register gpdac_config according to the needs
4. Initialize the GPIO pins of A and B channels
5. Initialize and enable DAC A channel and B channel
6. Write the data to be converted into the corresponding gpdac_a_data and gpdac_b_data bits in the register dac_cfg3 to complete the data conversion

6.3.4.2 DMA mode in independent mode

Each DAC channel has DMA capability. Taking channel A as an example for data conversion using DMA, the configuration process is as follows:

1. Set the DAC clock: write 1 to the corresponding bit value of dig_clk_src_sel in the register dig_clk_cfg0, that is, select xclk as the DAC clock source
2. Write 0 to the corresponding gpdac_ana_clk_sel and gpdac_dat_cha_sel bits in the register dac_cfg0, the data and clock come from itself, and work in independent mode.
3. Set the clock frequency division factor: the user sets the value of the dig_512k_div bit corresponding to the register dig_clk_cfg0 and the gpdac_mode bit corresponding to the register gpdac_config according to the needs
4. Initialize the GPIO pin of channel A as analog multiplexing
5. Initialize and enable DAC A channel
6. Initialize and enable the DMA channel: set the DMA transfer data width, source address, destination address and data transfer length, etc.
7. Enable DAC DMA mode: write 1 to the corresponding gpdac_dma_tx_en bit value in register gpdac_dma_config
8. Write the data to be converted into the register gpdac_dma_wdata, and act on the A or B channel according to different data formats to complete the data conversion

6.3.4.3 Combined mode as analog output of AudioDAC

When the DAC digital-to-analog converter is used as the analog signal output of AudioDAC, it can directly output audio analog signal to play music. Taking dual-channel differential mode as an example, the configuration process is as follows:

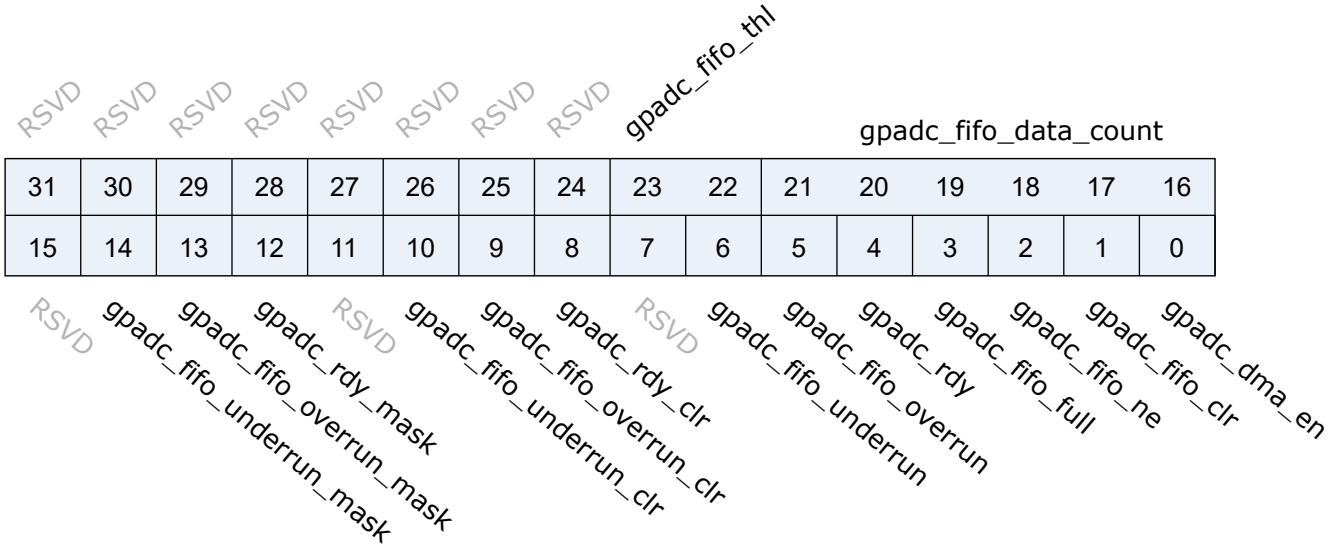
1. Write 1 to the corresponding gpdac_ana_clk_sel, gpdac_dat_chb_sel and gpdac_dat_cha_sel bits in the register dac_cfg0, the data and clock come from AudioDAC, and work in the combined mode.
2. Initialize and enable the A and B channels, and configure the corresponding pins as analog alternate functions.
3. Initialize the AudioDAC module and set it to GPDAC output mode, see AudioDAC chapter for details.
4. Start AudioDAC, start playing audio data, and DAC will start converting synchronously.

6.4 Register description

Name	Description
gpdac_config	dac data control register
gpdac_dma_config	dac dma control register
gpdac_dma_wdata	dac fifo data register
gpdac_tx_fifo_status	dac status register
dac_cfg0	DAC source control register
dac_cfg1	DAC CHA control register
dac_cfg2	DAC CHB control register
dac_cfg3	DAC converter data register

6.4.1 gpadc_config

Address：0x20002000

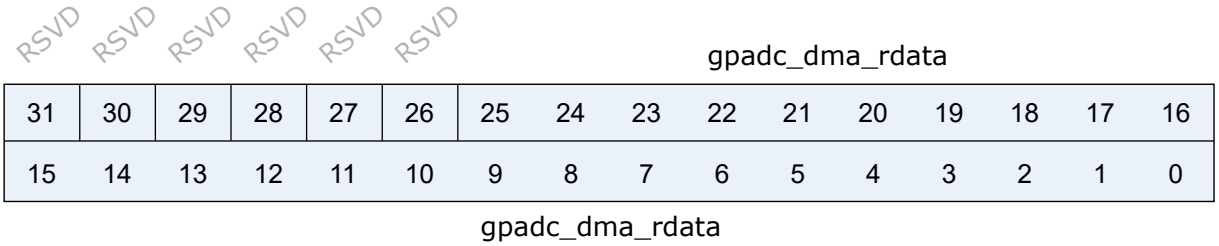


Bits	Name	Type	Reset	Description
31:24	RSVD			
23:22	gpadc_fifo_thl	r/w	2'd0	fifo threshold 2'b00: 1 data 2'b01: 4 data 2'b10: 8 data 2'b11: 16 data
21:16	gpadc_fifo_data_count	r	6'd0	fifo data number
15	RSVD			
14	gpadc_fifo_underrun_mask	r/w	1'b0	write 1 mask
13	gpadc_fifo_overrun_mask	r/w	1'b0	write 1 mask
12	gpadc_rdy_mask	r/w	1'b0	write 1 mask
11	RSVD			
10	gpadc_fifo_underrun_clr	r/w	1'b0	Write 1 to clear flag
9	gpadc_fifo_overrun_clr	r/w	1'b0	Write 1 to clear flag
8	gpadc_rdy_clr	r/w	1'b0	Write 1 to clear flag
7	RSVD			
6	gpadc_fifo_underrun	r	1'b0	FIFO underrun interrupt flag
5	gpadc_fifo_overrun	r	1'b0	FIFO overrun interrupt flag
4	gpadc_rdy	r	1'b0	Conversion data ready interrupt flag
3	gpadc_fifo_full	r	1'b0	FIFO full flag
2	gpadc_fifo_ne	r	1'b0	FIFO not empty flag
1	gpadc_fifo_clr	w1c	1'b0	FIFO clear signal

Bits	Name	Type	Reset	Description
0	gpadc_dma_en	r/w	1'b0	GPADC DMA enbale

6.4.2 gpadc_dma_rdata

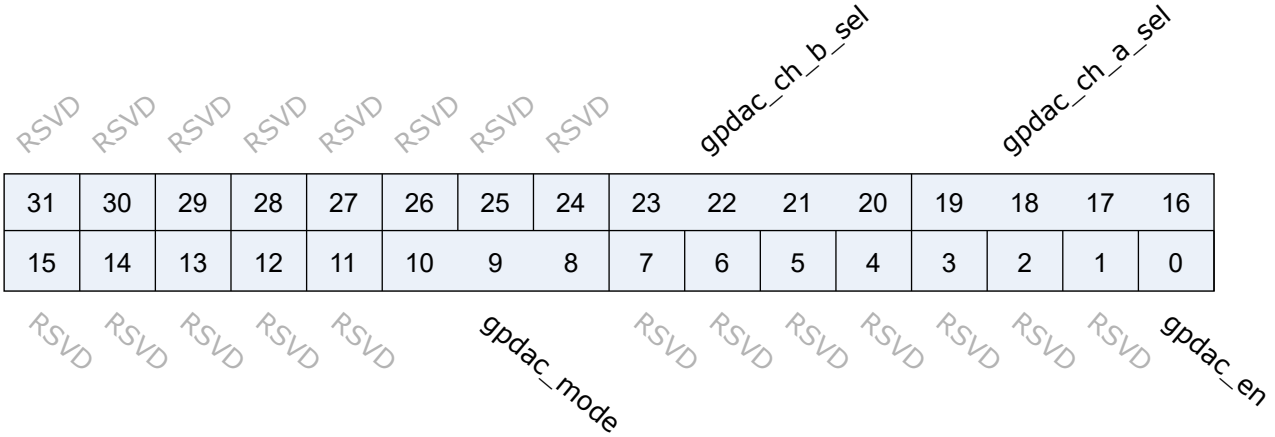
Address：0x20002004



Bits	Name	Type	Reset	Description
31:26	RSVD			
25:0	gpadc_dma_rdata	r	26'd0	GPADC finial conversion result stored in the FIFO

6.4.3 gpdac_config

Address：0x20002040

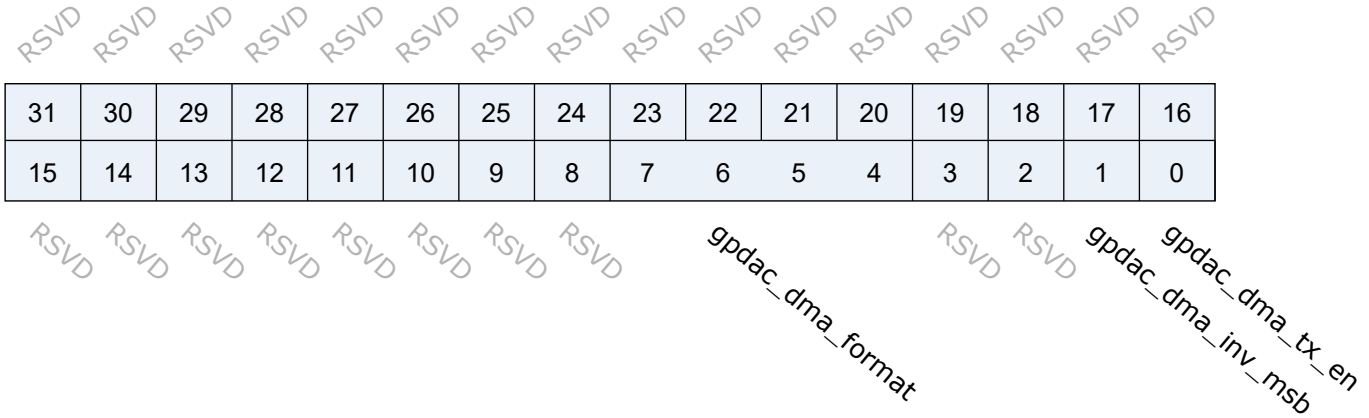


Bits	Name	Type	Reset	Description
31:24	RSVD			

Bits	Name	Type	Reset	Description
23:20	gpdac_ch_b_sel	r/w	0	Channel B Source Select 0: Reg 1: DMA 3: Sin Gen 4: A (The same as channel A) 5: A (Inverse of channel A)
19:16	gpdac_ch_a_sel	r/w	0	Channel A Source Select 0: Reg 1: DMA 3: Sin Gen
15:11	RSVD			
10:8	gpdac_mode	r/w	0	0:32k, 1:16k, 3:8k, 4:512k(for DMA only)
7:1	RSVD			
0	gpdac_en	r/w	0	GPDAC enable

6.4.4 gpdac_dma_config

Address：0x20002044



Bits	Name	Type	Reset	Description
31:8	RSVD			

Bits	Name	Type	Reset	Description
7:4	gpdac_dma_format	r/w	0	DMA TX format (Data 12-bit) 0: ([11:0]) A0, A1, A2… 1: ([27:16][11:0]) B0,A0, B1,A1, B2,A2… 2: ([27:16][11:0]) A1,A0, A3,A2, A5,A4… 4: ([15:4]) A0, A1, A2… 5: ([31:20][15:4]) B0,A0, B1,A1, B2,A2… 6: ([31:20][15:4]) A1,A0, A3,A2, A5,A4… 8: ([31:24][23:16][15:8][7:0]) A3,A2,A1,A0, A7,A6,A5,A4… 9: ([31:24][23:16][15:8][7:0]) B1,B0,A1,A0, B3,B2,A3,A2… 10: ([31:24][23:16][15:8][7:0]) B1,A1,B0,A0, B3,A3,B2,A2… 11: ([15:8][7:0]) B0,A0, B1,A1, B2,A2…
3:2	RSVD			
1	gpdac_dma_inv_msb	r/w	0	GPDAC DMA Data Inverse MSB
0	gpdac_dma_tx_en	r/w	0	GPDAC DMA TX enable

6.4.5 gpdac_dma_wdata

Address：0x20002048

gpdac_dma_wdata

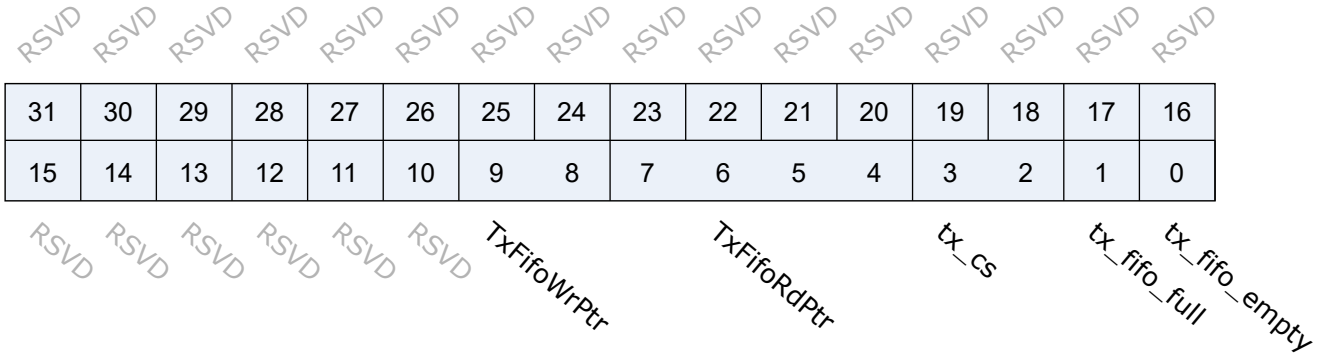
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpdac_dma_wdata

Bits	Name	Type	Reset	Description
31:0	gpdac_dma_wdata	w	x	GPDAC DMA TX data

6.4.6 gpdac_tx_fifo_status

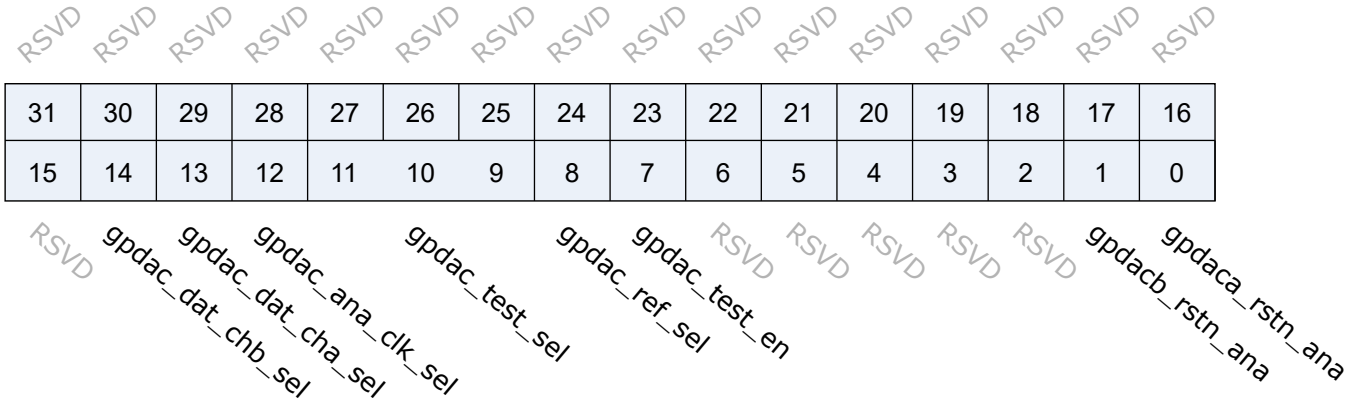
Address：0x2000204c



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	tx_fifo_full	r	0	fifo full flag
0	tx_fifo_empty	r	0	fifo empty flag

6.4.7 dac_cfg0

Address：0x20000120

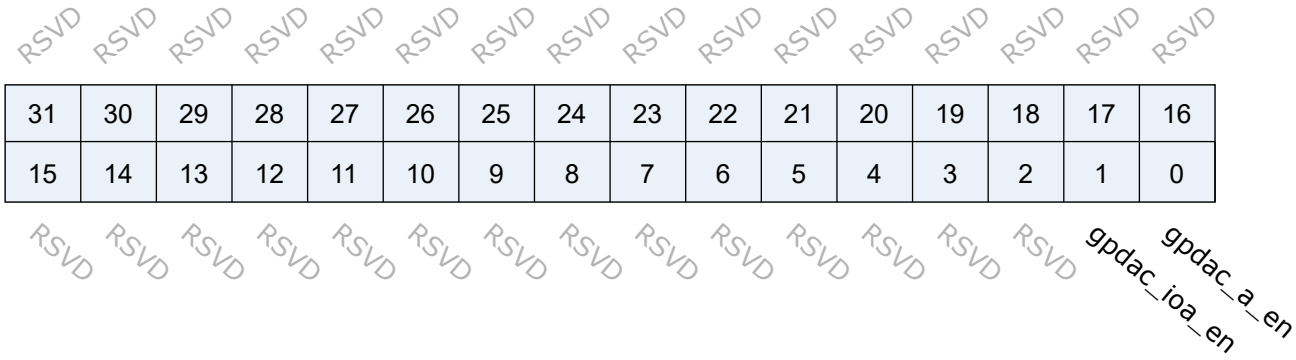


Bits	Name	Type	Reset	Description
31:15	RSVD			
14	gpdac_dat_chb_sel	r/w	1'h0	0:data from gpip, 1:data from audio pwm
13	gpdac_dat_cha_sel	r/w	1'h0	0:data from gpip, 1:data from audio pwm
12	gpdac_ana_clk_sel	r/w	1'h0	0:clock from gpip, 1:clock from audio pwm
11:9	gpdac_test_sel	r/w	3'h0	select test point 0 7

Bits	Name	Type	Reset	Description
8	gpdac_ref_sel	r/w	1'h0	Reference select 1'h0 Internal reference 1'h1 External reference
7	gpdac_test_en	r/w	1'h0	Test enable 1'h0 analog test disabled (ATEST is set in Hi-Z state) 1'h1 analog test point enabled to ATEST
6:2	RSVD			
1	gpdacb_rstn_ana	r/w	1'h1	Soft reset for DAC channel B, active low
0	gpdaca_rstn_ana	r/w	1'h1	Soft reset for DAC channel A, active low

6.4.8 dac_cfg1

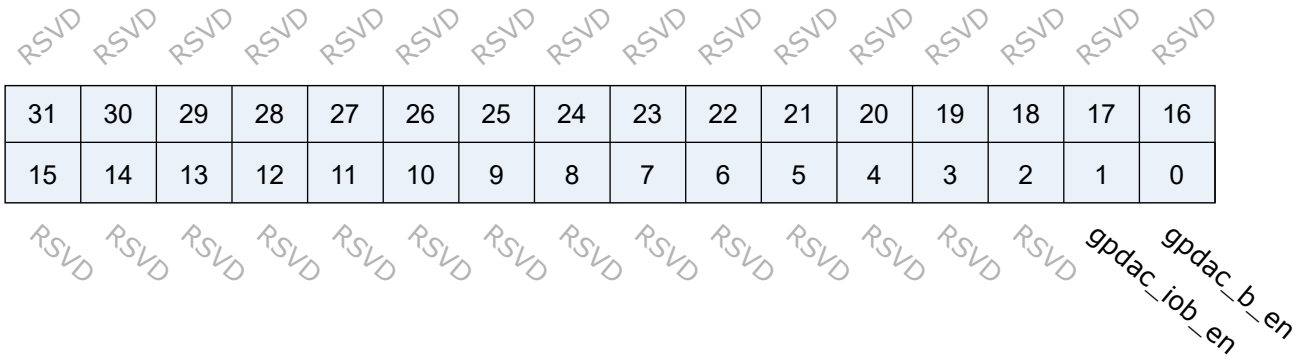
Address：0x20000124



Bits	Name	Type	Reset	Description
31:20	RSVD			
19:18	gpdac_a_rng	r/w	2'h3	Output voltage range control with internal/external reference
17:2	RSVD			
1	gpdac_ioa_en	r/w	1'h0	Channel A conversion output to pad enable 1'h0 Disable channel A conversion result to GPIO 1'h1 Enable channel A conversion result to GPIO
0	gpdac_a_en	r/w	1'h0	Channel A enable/disable signal 1'h0 Disable channel A conversion. 1'h1 Enable channel A conversion

6.4.9 dac_cfg2

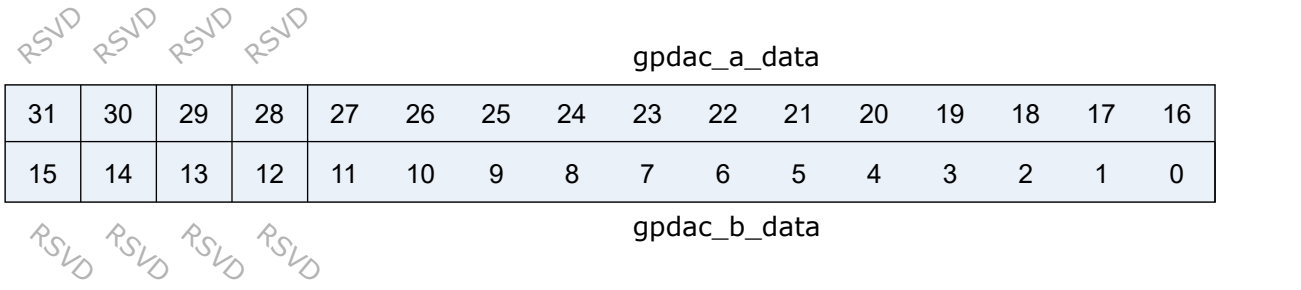
Address：0x20000128



Bits	Name	Type	Reset	Description
31:20	RSVD			
19:18	gpdac_b_rng	r/w	2'h3	Output voltage range control with internal/external reference
17:2	RSVD			
1	gpdac_iob_en	r/w	1'h0	channel B conversion output to pad enable 1'h0 Disable channel B conversion result to GPIO 1'h1 Enable channel B conversion result to GPIO
0	gpdac_b_en	r/w	1'h0	channel B enable/disable signal 1'h0 Disable channel B conversion. 1'h1 Enable channel B conversion

6.4.10 dac_cfg3

Address：0x2000012c



Bits	Name	Type	Reset	Description
31:28	RSVD			
27:16	gpdac_a_data	r/w	12'h0	Channel A Data input

Bits	Name	Type	Reset	Description
15:12	RSVD			
11:0	gpdac_b_data	r/w	12'h0	Channel B Data input

7 DMA

7.1 Overview

Direct Memory Access (DMA), a kind of memory access technique, can directly read and write system memory independently without relying on processor. Under the same processor load, DMA is a fast way to transfer data.

It is provided with 1 independent DMA controllers, has 4 independent dedicated channels, managing data transmission between peripherals and memory to enhance bus efficiency. DMA supports the transfer from memory to memory, memory to peripherals, and peripheral to memory, and provides the LLI linked list function. The software configures the data transfer size, data source address, and destination address.

7.2 Features

- 1 DMA controller with 4 independent dedicated channels
- Independent control source and destination access width (single byte, double bytes, and four bytes)
- Each channel acts as a read-write cache independently
- Each channel can be triggered by independent peripheral hardware or software
- Four kinds of process control
  - DMA process control, source memory, destination memory
  - DMA process control, source memory, destination peripherals
  - DMA process control, source peripherals, destination memory
  - DMA process control, source peripherals, destination peripherals
- Supports the LLI linked list function to improve DMA efficiency

7.3 Functional Description

7.3.1 Operating Principle

When a device tries to directly transfer data to another device through the bus, it will first send a DMA request signal to CPU. The peripheral submits a request through DMA to take over control of the bus from CPU. After receiving this signal, CPU will respond to the DMA signal according to the priority of signal and the order of the DMA request after the current bus cycle is over. When CPU responds to a DMA request for a device interface, it will give up control of the bus.

Under the control of DMA controllers, peripherals and memory exchange data directly without CPU intervention. After data is transferred, the device will send a DMA end signal to CPU and return the control of the bus.

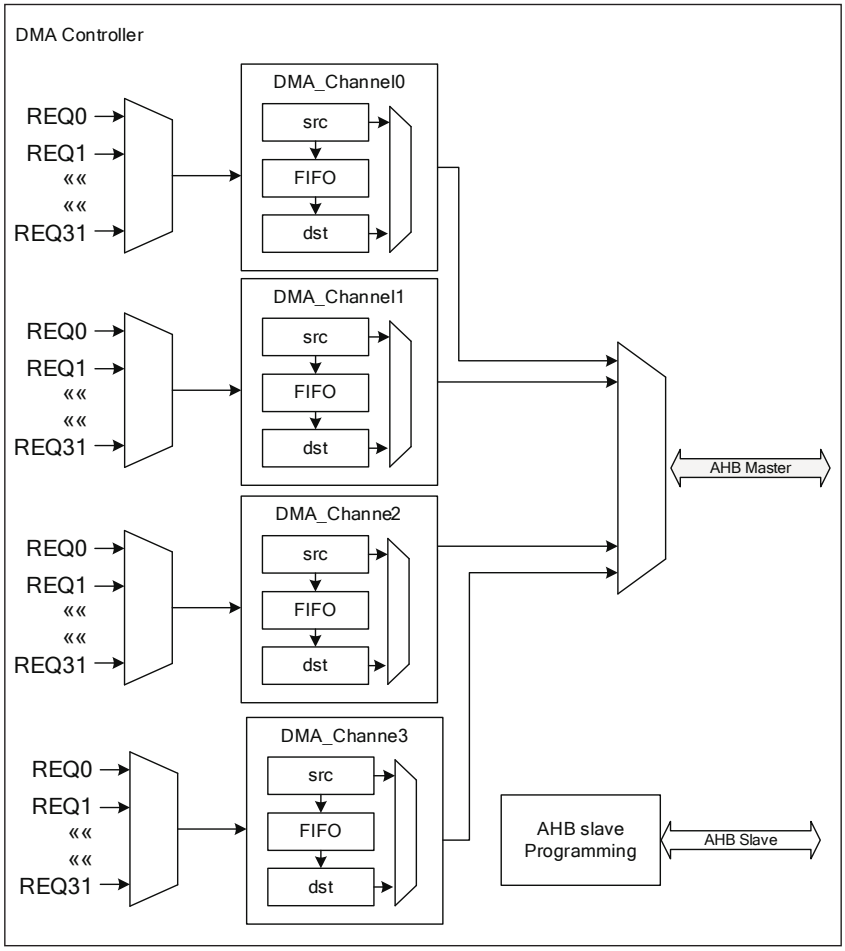


Fig. 7.1: Block Diagram of DMA

DMA has one AHB Master interfaces and one AHB Slave interface. Based on the current configuration requirements, the AHB Master interface actively accesses the memory or peripherals through system bus, and serves as a data transfer port. The AHB Slave interface for configuring DMA only supports 32bit access.

7.3.2 DMA Channel Configuration

DMA support 4 channels. All channels can run simultaneously without interference. Here is the configuration process of DMA channel x:

1. Set the 32-bit source address in the register DMA_C0SrcAddr.
2. Set the 32-bit destination address in the register DMA_C0DstAddr.
3. Automatic address accumulation: You can set to enable/disable the automatic address accumulation mode by configuring the SI (source) and DI (destination) in the register DMA_C0Control. When it is set to 1, this mode is enabled.
4. You can set the width of transferred data by configuring the SWidth (source) and DWidth (destination) bits in the register DMA_C0Control, including single byte, double bytes and four bytes options.
5. You can set the Burst type by configuring the SBSize (source) and DBSize (destination) bits in the register DMA_C0Control, including INCR1, INCR4, INCR8, and INCR16 options.
6. It is worth noting that in the configured combination, the single burst of DMA must not exceed 16 bytes.
7. Range of data transfer length:0-4095

7.3.3 Supported Peripherals

You can determine the peripherals supported by the current DMA by configuring the SrcPeripheral (source) and DstPeripheral (destination). The relationship between peripheral types and configuration values is shown as follows:

Value	DMA0
0	UART0_RX
1	UART0_TX
2	UART1_RX
3	UART1_TX
6	I2C0_RX
7	I2C0_TX
9	GPIO_TX
10	SPI0_RX
11	SPI0_TX
13	AUDIO_TX
14	I2C1_RX
15	I2C1_TX
16	I2S_RX
17	I2S_TX
20	DBI_TX
21	AUADC_RX
22	GPADC_RX
23	GPDAC_TX

Fig. 7.2: Peripheral Type Selection

Partial peripheral configuration examples:

UART uses DMA to transfer data

When UART sends data packets, DMA can greatly shorten the CPU’s processing time, so that CPU resources will not be highly wasted, especially when UART sends and receives a large number of data packets (such as sending and receiving instructions frequently).

For example, the UART0 transfer is configured as follows:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 1. That is, set Source peripheral to UART0_-TX
2. Set the value of the DstPeripheral in the register DMA_C0Config to 0. That is, set Destination peripheral to UART0_RX

I2C uses DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 7. That is, set Source peripheral to I2C0_TX
2. Set the value of the DstPeripheral in the register DMA_C0Config to 6. That is, set Destination peripheral to I2C0_-RX

SPI uses DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 11. That is, set Source peripheral to SPI0_TX
2. Set the value of the DstPeripheral in the register DMA_C0Config to 10. That is, set Destination peripheral to SPI0_RX

ADC use DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 22. That is, set Source peripheral to GPADC.

7.3.4 Linked List Mode

DMA supports the linked list working mode. During a DMA read or write operation, data can be filled in the next linked list. After the data transfer of the current linked list is completed, the start address of the next linked list can be obtained by reading the value of the register DMA_C0LLI, to directly transfer the data of the next linked list. This ensures continuous and uninterrupted work during DMA transfer, and improves the efficiency of CPU and DMA.

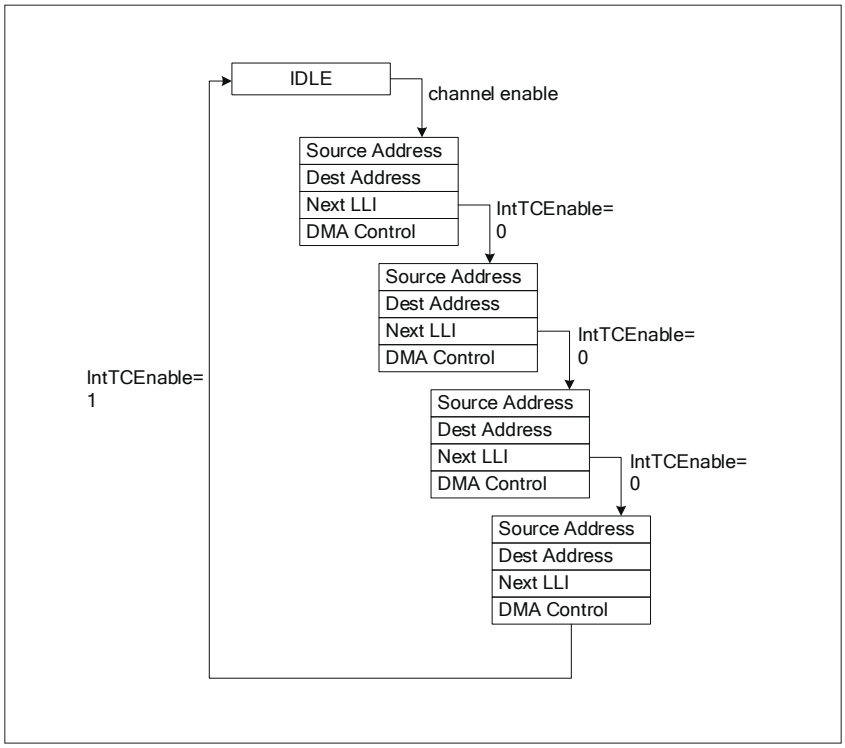


Fig. 7.3: LLI Framework

7.3.5 DMA Interrupt

- DMA_INT_TCOMPLETED
  - Data transfer complete interrupt: This interrupt will be generated when a data transfer is completed
- DMA_INT_ERR
  - Data transfer error interrupt: This interrupt will be generated when an error occurs during data transfer

7.4 Transfer Mode

7.4.1 Memory to Memory

After this mode is enabled, DMA will transfer the data from the source address to the destination address based on the preset TransferSize. Upon transfer completed, the DMA controller will automatically return to the idle state and wait for the next transfer.

Configuration process:

1. Set the DMA_C0SrcAddr value of the register to the source memory address
2. Set the DMA_C0DstAddr value of the register to the destination memory address
3. Select a transfer mode, and set the value of the FlowCntrl bit in the register DMA_C0Config to 0, namely selecting the memorytomemory mode
4. Set the values of the corresponding bits in the register DMA_C0Control: the SI bit is set to 1, the DI bit is set to 0, the source address automatic accumulation mode is enabled; set the transfer width of the source and destination through SWidth and DWidth respectively; and set the burst type of the source and the destination through SBSize and DBSize respectively
5. Select a proper channel, enable DMA, and complete data transfer

7.4.2 Memory to Peripherals

In this mode, DMA will transfer the data from the source to the internal cache according to the preset TransferSize. The transfer will automatically pause when the cache space is insufficient and continue when there is enough cache space until the preset TransferSize is met.

Moreover, when the destination peripheral request is triggered, the target configuration will be burst to the destination address, and it will automatically return to the idle state until the preset TransferSize is met and wait for the next transfer.

Configuration process:

1. Set the DMA_C0SrcAddr value of the register to the source memory address
2. Set the DMA_C0DstAddr value of the register to the destination peripheral address

3. Select a transfer mode, and set the value of the FlowCntrl bit in the register DMA_C0Config to 1, namely selecting the Memorytoperipheral mode
4. Set the values of the corresponding bits in the register DMA_C0Control: the SI bit is set to 1, the DI bit is set to 0, the source address automatic accumulation mode is enabled; set the transfer width of the source and destination through SWidth and DWidth respectively; and set the burst type of the source and the destination through SBSize and DBSize respectively
5. Select a proper channel, enable DMA, and complete data transfer

7.4.3 Peripheral to Memory

In this mode, when the source peripheral request is triggered, the source configuration will be burst into the cache until the preset TransferSize is met. Moreover, when the internal cache is enough for one burst to the destination, DMA will automatically transfer the cached content to the destination address, automatically return to the idle state until the preset TransferSize is met, and wait for the next transfer.

Configuration process:

1. Set the DMA_C0SrcAddr value of the register to the source peripheral address
2. Set the DMA_C0DstAddr value of the register to the destination memory address
3. Select a transfer mode, and set the value of the FlowCntrl bit in the register DMA_C0Config to 2, namely selecting the Peripheraltomemory mode
4. Set the values of the corresponding bits in the register DMA_C0Control: Set the DI and SI bits to 1 to enable the automatic address accumulation mode; set the transfer width of the source and destination through SWidth and DWidth respectively; and set the burst type of the source and the destination through SBSize and DBSize respectively
5. Select a proper channel, enable DMA, and complete data transfer

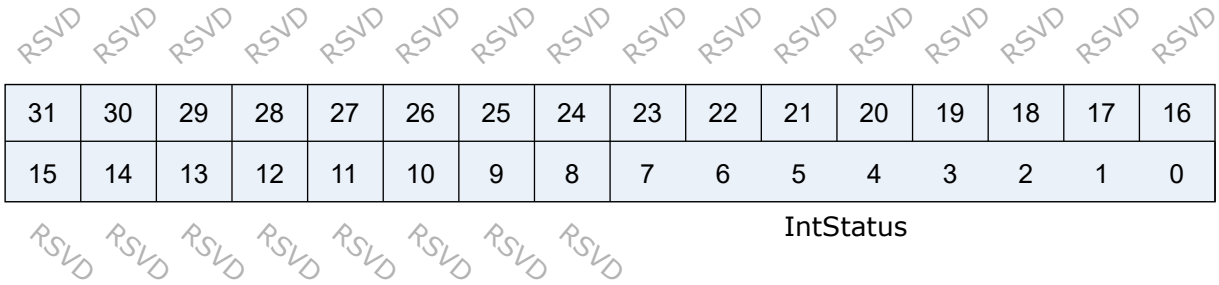
7.5 Register description

Name	Description
DMA_IntStatus	Interrupt status after masking
DMA_IntTCStatus	TC interrupt status
DMA_IntTCClear	TC interrupt clear
DMA_IntErrorStatus	Error interrupt status
DMA_IntErrClr	Error interrupt clear
DMA_RawIntTCStatus	TC interrupt status before masking

Name	Description
DMA_RawIntErrorStatus	Error interrupt status before masking
DMA_EnbldChns	Channel enable status
DMA_SoftBReq	Software burst request
DMA_SoftSReq	Software single request
DMA_SoftLBReq	Software last burst request
DMA_SoftLSReq	Software last single request
DMA_Config	Enable and endian
DMA_Sync	Synchronization logic
DMA_C0SrcAddr	Channel 0 source address
DMA_C0DstAddr	Channel 0 destination address
DMA_C0LLI	Channel 0 linked list address
DMA_C0Control	Channel 0 control
DMA_C0Config	Channel 0 configure
DMA_C1SrcAddr	Channel 1 source address
DMA_C1DstAddr	Channel 1 destination address
DMA_C1LLI	Channel 1 linked list address
DMA_C1Control	Channel 1 control
DMA_C1Config	Channel 1 configure
DMA_C2SrcAddr	Channel 2 source address
DMA_C2DstAddr	Channel 2 destination address
DMA_C2LLI	Channel 2 linked list address
DMA_C2Control	Channel 2 control
DMA_C2Config	Channel 2 configure
DMA_C3SrcAddr	Channel 3 source address
DMA_C3DstAddr	Channel 3 destination address
DMA_C3LLI	Channel 3 linked list address
DMA_C3Control	Channel 3 control
DMA_C3Config	Channel 3 configure

7.5.1 DMA_IntStatus

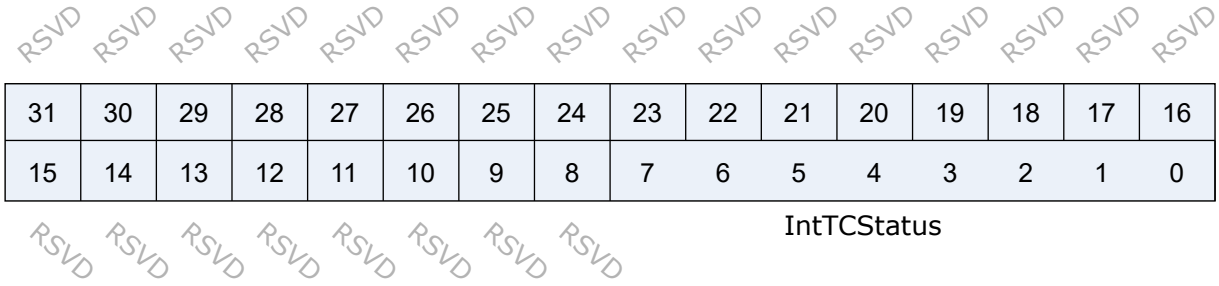
Address：0x2000c000



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntStatus	r	0	Status of the DMA interrupts after masking

7.5.2 DMA_IntTCStatus

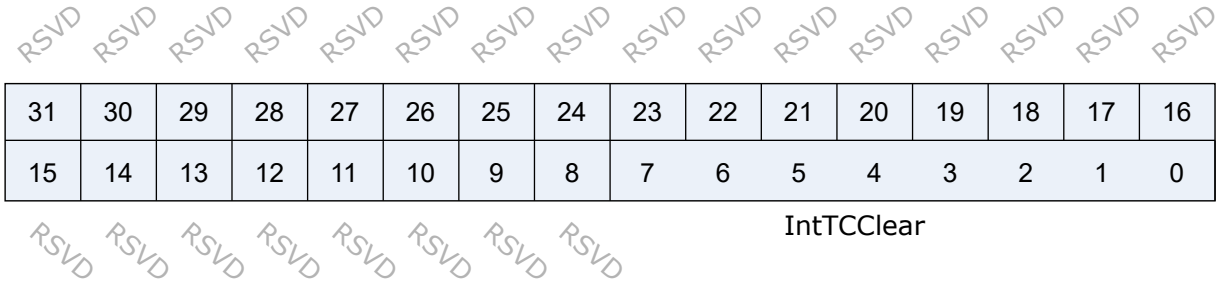
Address：0x2000c004



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntTCStatus	r	0	Interrupt terminal count request status

7.5.3 DMA_IntTCClear

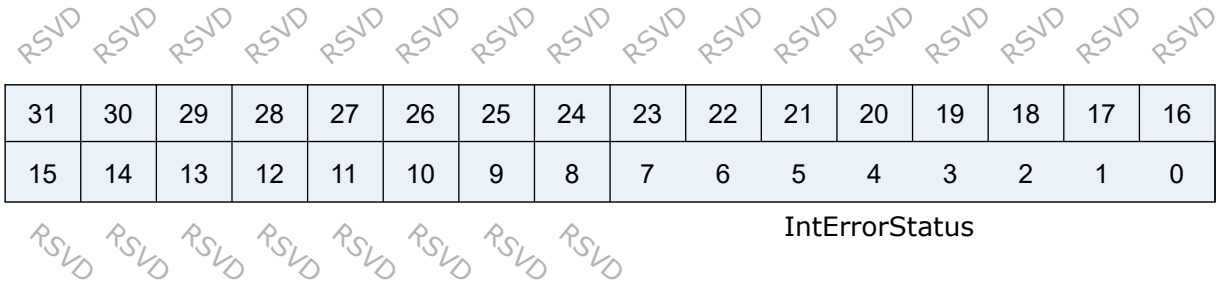
Address：0x2000c008



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntTCClear	w	0	Terminal count request clear

7.5.4 DMA_IntErrorStatus

Address：0x2000c00c



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntErrorStatus	r	0	Interrupt error status

7.5.5 DMA_IntErrClr

Address：0x2000c010

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD IntErrClr

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntErrClr	w	0	Interrupt error clear

7.5.6 DMA_RawIntTCStatus

Address：0x2000c014

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RawIntTCStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	RawIntTCStatus	r	0	Status of the terminal count interrupt prior to masking

7.5.7 DMA_RawIntErrorStatus

Address：0x2000c018

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

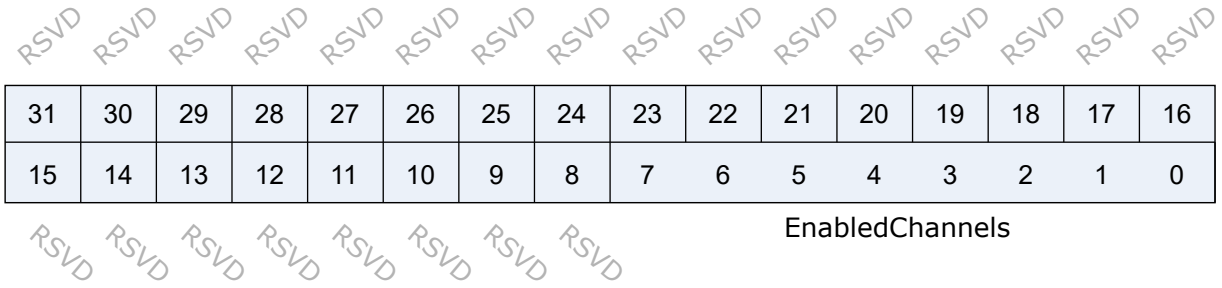
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RawIntErrorStatus

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	RawIntErrorStatus	r	0	Status of the error interrupt prior to masking

7.5.8 DMA_EnbldChns

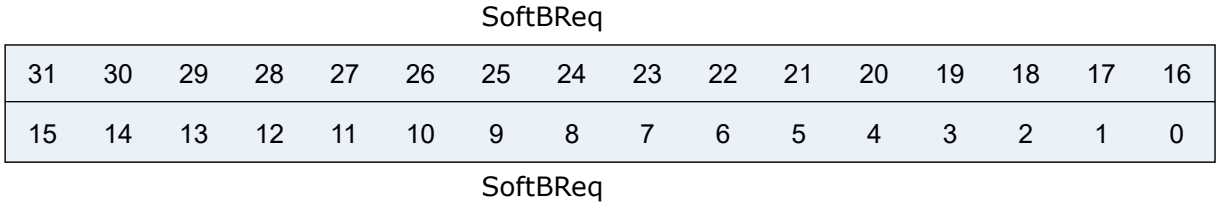
Address：0x2000c01c



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	EnabledChannels	r	0	Channel enable status

7.5.9 DMA_SoftBReq

Address：0x2000c020



Bits	Name	Type	Reset	Description
31:0	SoftBReq	r/w	0	Software burst request

7.5.10 DMA_SoftSReq

Address：0x2000c024

SoftSReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftSReq

Bits	Name	Type	Reset	Description
31:0	SoftSReq	r/w	0	Software single request

7.5.11 DMA_SoftLBReq

Address：0x2000c028

SoftLBReq

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftLBReq

Bits	Name	Type	Reset	Description
31:0	SoftLBReq	r/w	0	Software last burst request

7.5.12 DMA_SoftLSReq

Address：0x2000c02c

SoftLSReq

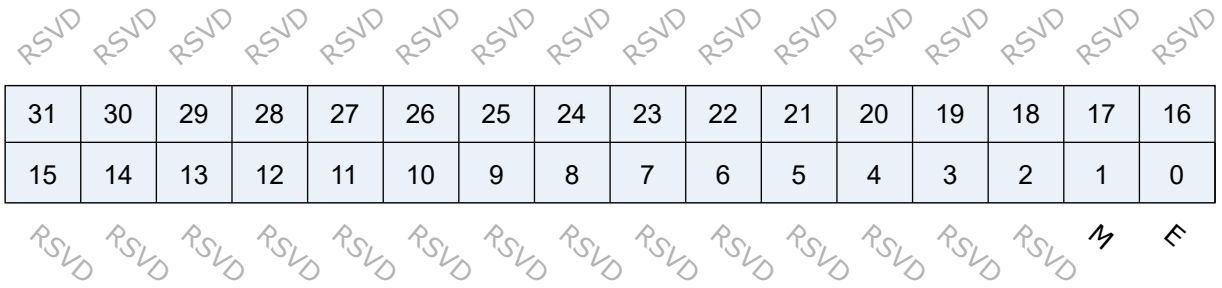
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SoftLSReq

Bits	Name	Type	Reset	Description
31:0	SoftLSReq	r/w	0	Software last single request

7.5.13 DMA_Config

Address：0x2000c030



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	M	r/w	0	AHB Master endianness configuration: 0 = little-endian, 1 = big-endian
0	E	r/w	0	SMDMA Enable.

7.5.14 DMA_Sync

Address：0x2000c034

DMA_Sync

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DMA_Sync

Bits	Name	Type	Reset	Description
31:0	DMA_Sync	r/w	0	DMA synchronization logic for DMA request signals: 0 = enable, 1 = disable

7.5.15 DMA_C0SrcAddr

Address：0x2000c100

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	DMA source address

7.5.16 DMA_C0DstAddr

Address：0x2000c104

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	DMA Destination address

7.5.17 DMA_C0LLI

Address：0x2000c108

LLI

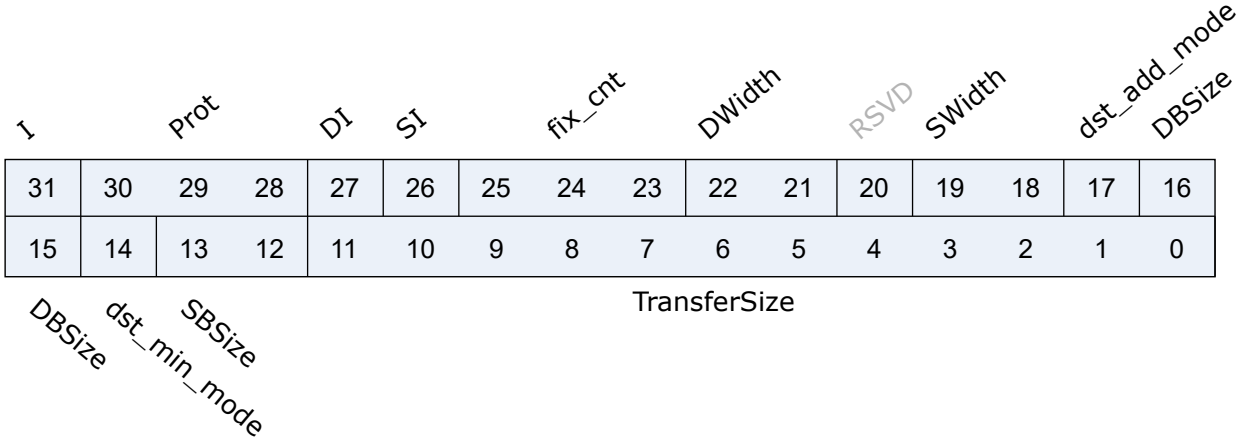
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	First linked list item. Bits [1:0] must be 0.

7.5.18 DMA_C0Control

Address：0x2000c10c

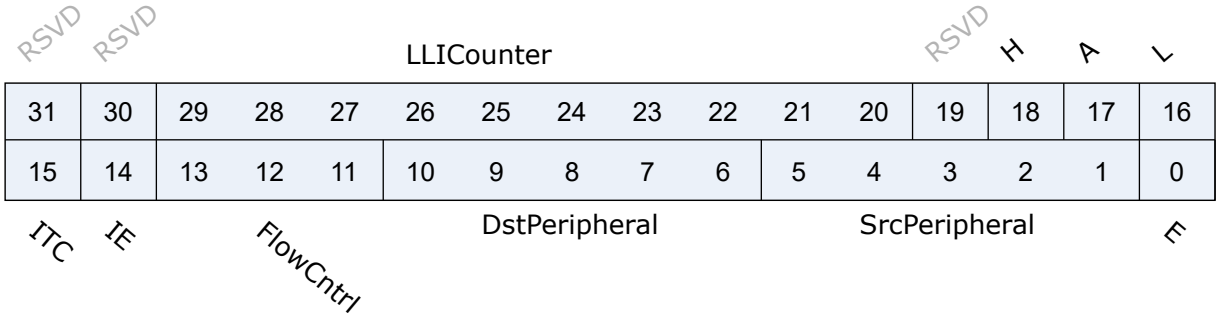


Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic

Bits	Name	Type	Reset	Description
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSize	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.19 DMA_C0Config

Address：0x2000c110



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:20	LLICounter	r	0	LLI counter. Increased 1 each LLI run. Cleared 0 when config Control.
19	RSVD			
18	H	r/w	0	Halt: 0 = enable DMA requests, 1 = ignore subsequent source DMA requests.
17	A	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.

Bits	Name	Type	Reset	Description
16	L	r/w	0	Lock.
15	ITC	r/w	0	Terminal count interrupt mask.
14	IE	r/w	0	Interrupt error mask.
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	E	r/w	0	Channel enable.
-1:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.20 DMA_C1SrcAddr

Address：0x2000c200

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	DMA source address

7.5.21 DMA_C1DstAddr

Address：0x2000c204

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	DMA Destination address

7.5.22 DMA_C1LLI

Address：0x2000c208

LLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	First linked list item. Bits [1:0] must be 0.

7.5.23 DMA_C1Control

Address：0x2000c20c

I Prot DI SI fix_cnt DWidth RSVD SWidth dst_add_mode DBSize

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

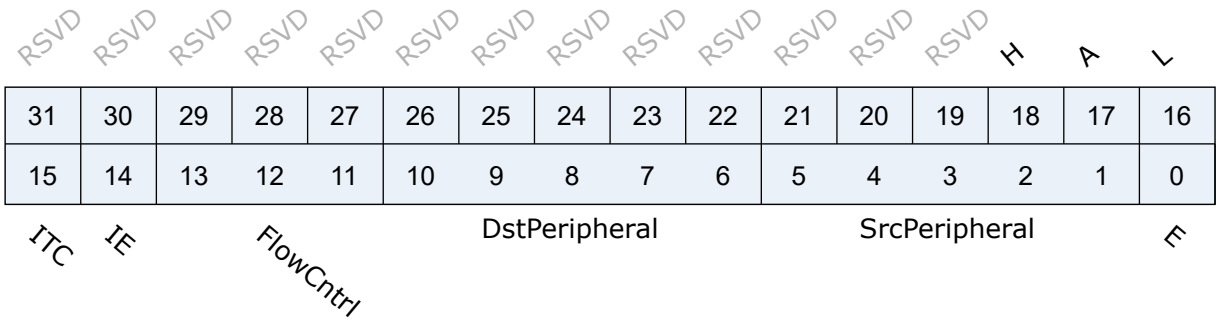
TransferSize

DBSize dst_min_mode SBSize

Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSize	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.24 DMA_C1Config

Address：0x2000c210



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	Halt: 0 = enable DMA requests, 1 = ignore subsequent source DMA requests.
17	A	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
16	L	r/w	0	Lock.
15	ITC	r/w	0	Terminal count interrupt mask.
14	IE	r/w	0	Interrupt error mask.
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	E	r/w	0	Channel enable.
-1:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.25 DMA_C2SrcAddr

Address：0x2000c300

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	DMA source address

7.5.26 DMA_C2DstAddr

Address：0x2000c304

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	DMA Destination address

7.5.27 DMA_C2LLI

Address：0x2000c308

LLI

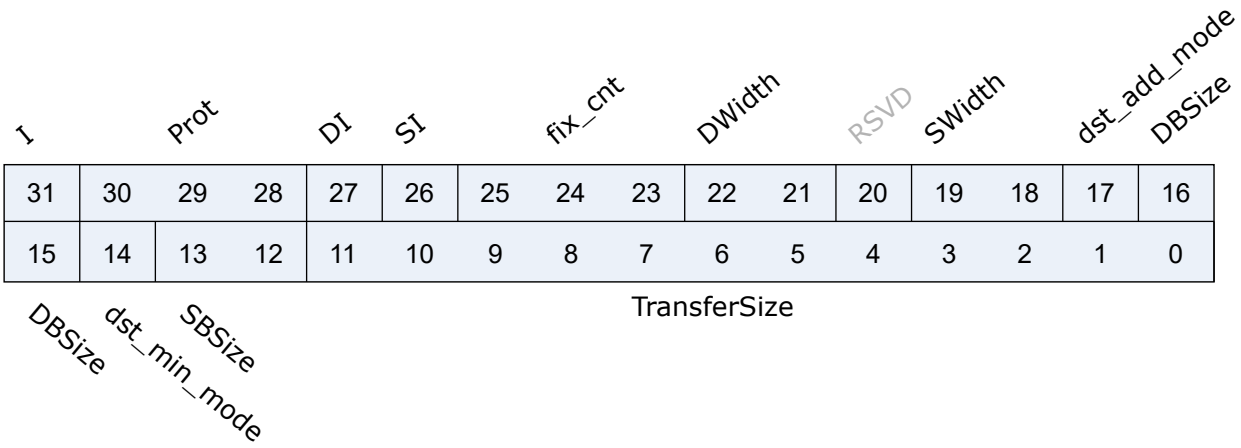
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	First linked list item. Bits [1:0] must be 0.

7.5.28 DMA_C2Control

Address：0x2000c30c

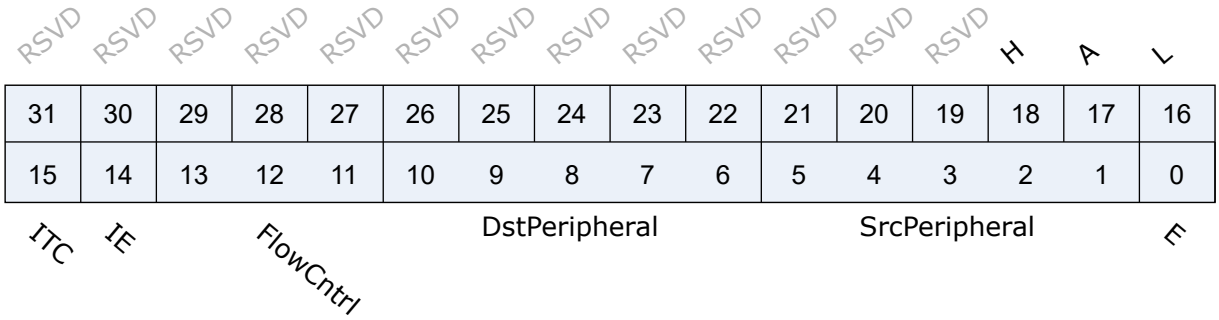


Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			

Bits	Name	Type	Reset	Description
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSize	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.29 DMA_C2Config

Address：0x2000c310



Bits	Name	Type	Reset	Description
31:19	RSVD			

Bits	Name	Type	Reset	Description
18	H	r/w	0	Halt: 0 = enable DMA requests, 1 = ignore subsequent source DMA requests.
17	A	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
16	L	r/w	0	Lock.
15	ITC	r/w	0	Terminal count interrupt mask.
14	IE	r/w	0	Interrupt error mask.
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	E	r/w	0	Channel enable.
-1:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.30 DMA_C3SrcAddr

Address：0x2000c400

SrcAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SrcAddr

Bits	Name	Type	Reset	Description
31:0	SrcAddr	r/w	0	DMA source address

7.5.31 DMA_C3DstAddr

Address：0x2000c404

DstAddr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DstAddr

Bits	Name	Type	Reset	Description
31:0	DstAddr	r/w	0	DMA Destination address

7.5.32 DMA_C3LLI

Address：0x2000c408

LLI

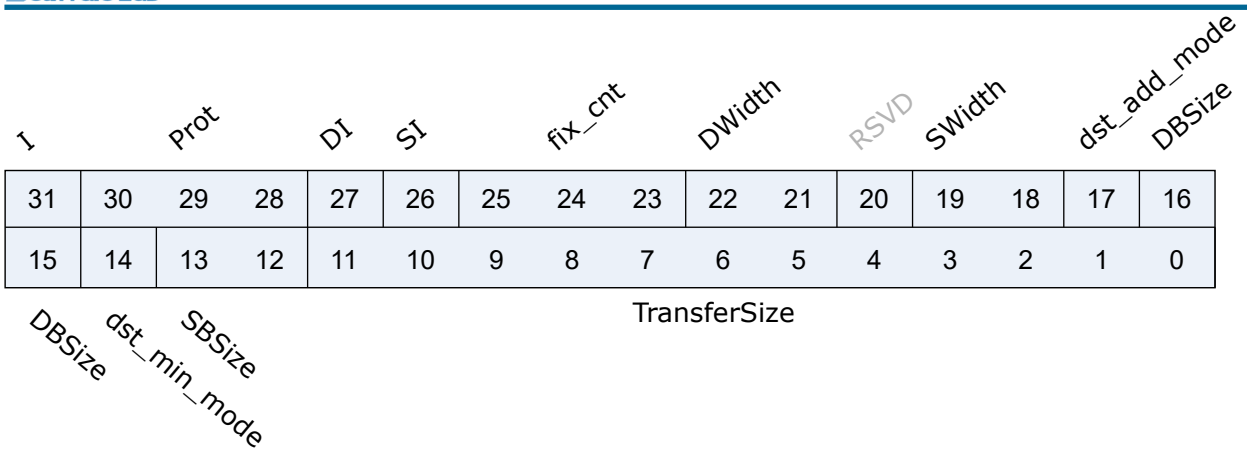
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	First linked list item. Bits [1:0] must be 0.

7.5.33 DMA_C3Control

Address：0x2000c40c

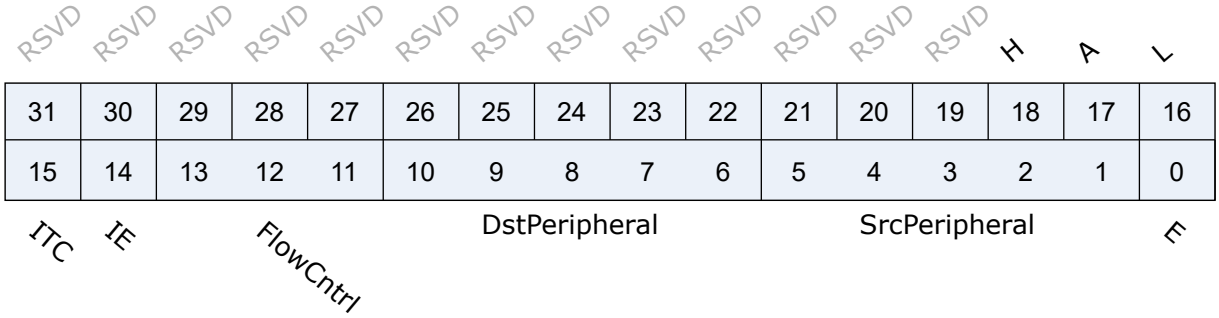


Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Source increment. When set, the source address is incremented after each transfer.
25:23	fix_cnt	r/w	3'd0	Only effect when dst_min_mode = 1 Destination transfer cnt = (total src byte cnt - (fix_cnt«DWidth))«DWidth
22:21	DWidth	r/w	2'b10	Destination transfer width: 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
20	RSVD			
19:18	SWidth	r/w	2'b10	Source transfer width 2'b00 : byte 2'b01 : half-word 2'b10 : word 2'b11 : double-word
17	dst_add_mode	r/w	1'b0	Add mode : issue remain destination traffic

Bits	Name	Type	Reset	Description
16:15	DBSize	r/w	2'b01	Destination burst size 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
14	dst_min_mode	r/w	1'b0	Minus mode : Not issue all destination traffic
13:12	SBSize	r/w	2'b01	Source burst size: 2'b00 : INCR1 2'b01 : INCR4 2'b10 : INCR8 2'b11 : INCR16 Note : SBSize*Swidth should <= CH FIFO Size
11:0	TransferSize	r/w	0	Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

7.5.34 DMA_C3Config

Address：0x2000c410



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	Halt: 0 = enable DMA requests, 1 = ignore subsequent source DMA requests.
17	A	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
16	L	r/w	0	Lock.
15	ITC	r/w	0	Terminal count interrupt mask.
14	IE	r/w	0	Interrupt error mask.

Bits	Name	Type	Reset	Description
13:11	FlowCntrl	r/w	0	000: Memory-to-memory (DMA) 001: Memory-to-peripheral (DMA) 010: Peripheral-to-memory (DMA) 011: Source peripheral-to-Destination peripheral (DMA) 100: Source peripheral-to-Destination peripheral (Destination peripheral) 101: Memory-to-peripheral (peripheral) 110: Peripheral-to-memory (peripheral) 111: Source peripheral-to-Destination peripheral (Source peripheral)
10:6	DstPeripheral	r/w	0	Destination peripheral.
5:1	SrcPeripheral	r/w	0	Source peripheral.
0	E	r/w	0	Channel enable.
-1:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

8 IR

8.1 Overview

Infrared Remote Control (IR) is a wireless and non-contact control technique that features strong anti-interference, reliable information transmission, low power consumption, and low cost. The IR radiating circuit uses the IR LED to emit the modulated infrared waves. The receiving circuit consists of infrared receiving diodes, triodes or silicon photocells, which convert the infrared light emitted by the infrared transmitter into corresponding electrical signals, and then send them to the post amplifier.

8.2 Features

- Receives data through the fixed NEC and RC5 protocols
- Receives data in any format by pulse width counting
- 64*2 bytes receive FIFO
- End of receiving interrupts

8.3 Functional Description

8.3.1 Fixed Protocol Based Receiving

IR receiving supports the fixed NEC and RC5 protocols.

- NEC Protocol

Logical ‘1’ and logical ‘0’ waveforms of the NEC protocol are shown as follows:

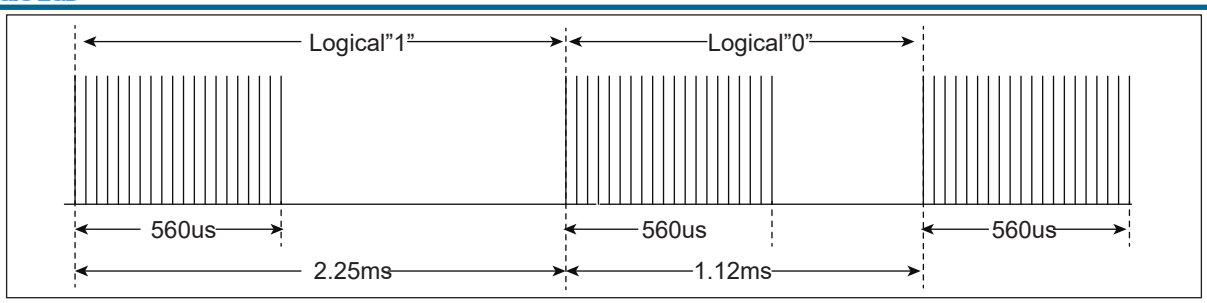


Fig. 8.1: NEC Logic Waveform

Logical ‘1’ is 2.25 ms and the pulse time is 560 us. Logical ‘0’ is 1.12 ms and the pulse time is 560 us. The format of the NEC protocol is shown as follows:

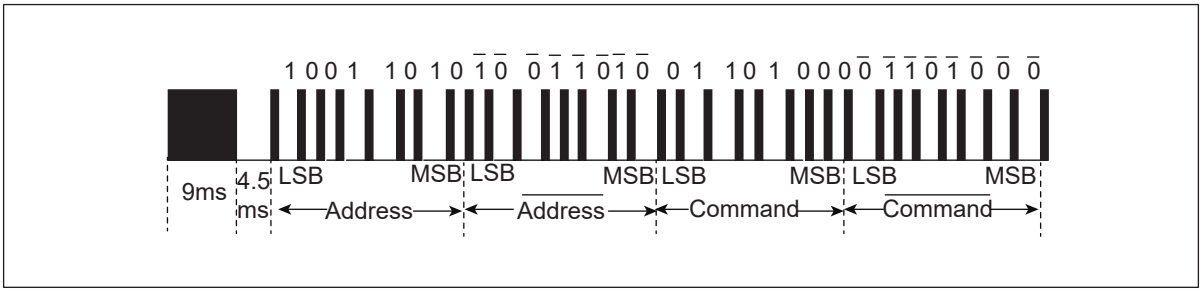


Fig. 8.2: NEC Protocol Waveform

The head pulse is a 9 ms high-level pulse and a 4.5 ms low-level pulse, followed by an 8-bit address code and its radix-minus-one complement (diminished radix complement), then an 8-bit command code and its radix complement, and the tail pulse is a 560 us high-level pulse and a 560 us low-level pulse.

- RC5 Protocol

Logical ‘1’ and logical ‘0’ waveforms of the RC5 protocol are shown as follows:

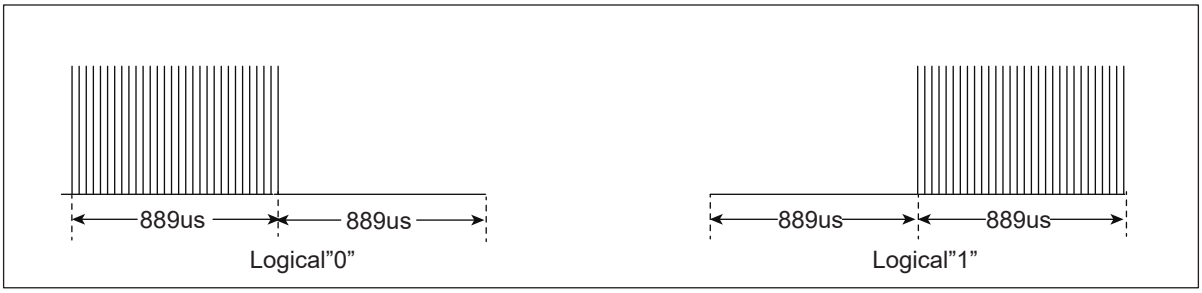


Fig. 8.3: RC5 Logic Waveform

The logical ‘1’ is 1.778 ms, first the 889 us low level and then the 889 us high level. Logical ‘0’ is opposite to the logical 1 waveform. The format of the RC5 protocol is shown as follows:

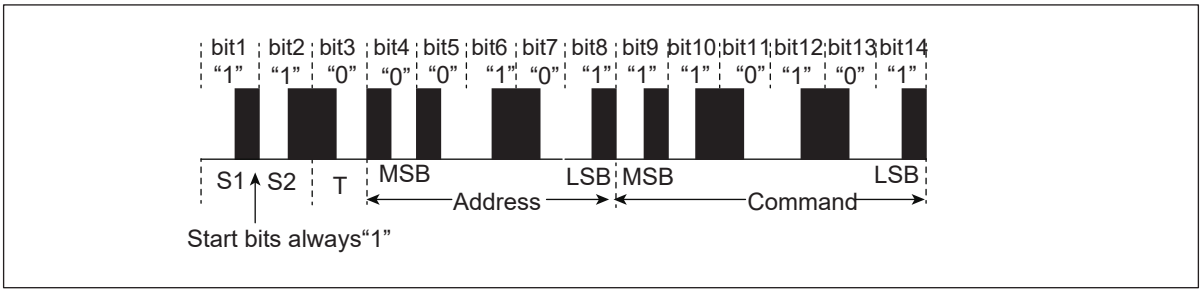


Fig. 8.4: RC5 Protocol Waveform

The first two bits are the Start bits, both logical ‘1’. The third bit is the toggle bit, which toggles each time a key is released and pressed again. The next 5 bits are address bits and the next 6 bits are command bits.

It is worth noting that to improve the receiving sensitivity, the common infrared integrated receiver outputs a low level after receiving a high level, so you need to enable the receiving toggle function when using the IR receiving function.

8.3.2 Pulse width receiving

For data in any format other than the NEC and RC5 protocols, IR will count the duration of each high or low level in turn by its clock, and then store the data into the RX FIFO with a depth of 64 and a width of 2 bytes.

8.3.3 IR Interrupt

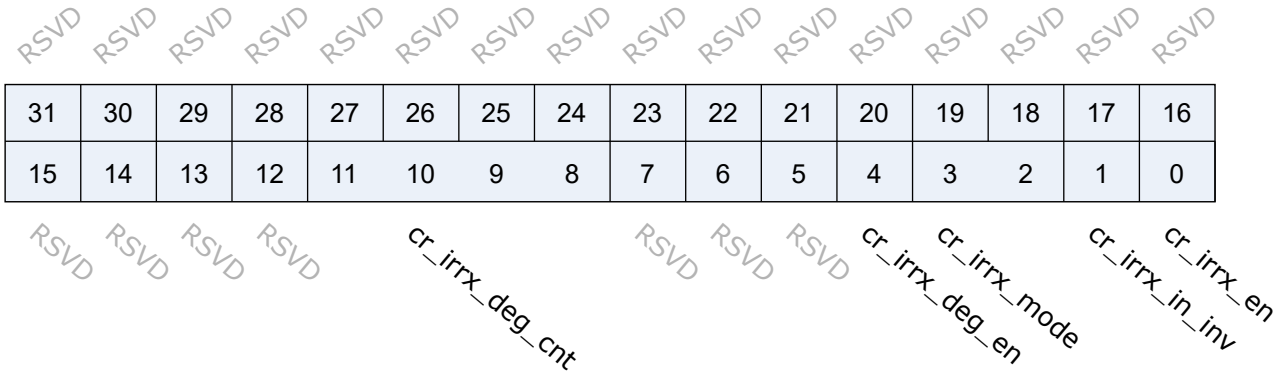
IR has a separate receiving end interrupt. During the receiving process, it will use the internal working clock to count the time that each level is maintained. Once the count reaches the set end threshold, a receiving end interrupt will be generated. The receive interrupt status and clear interrupt can be queried through the register IRRX_INT_STS.

8.4 Register description

Name	Description
irrx_config	RX configure
irrx_int_sts	RX interrupt status
irrx_pw_config	RX pulse width threshold
irrx_data_count	RX data bit count
irrx_data_word0	RX low 32-bit data
irrx_data_word1	RX high 32-bit data
ir_fifo_config_0	FIFO status
ir_fifo_config_1	FIFO threshold and available count
ir_fifo_rdata	RX FIFO

8.4.1 irrx_config

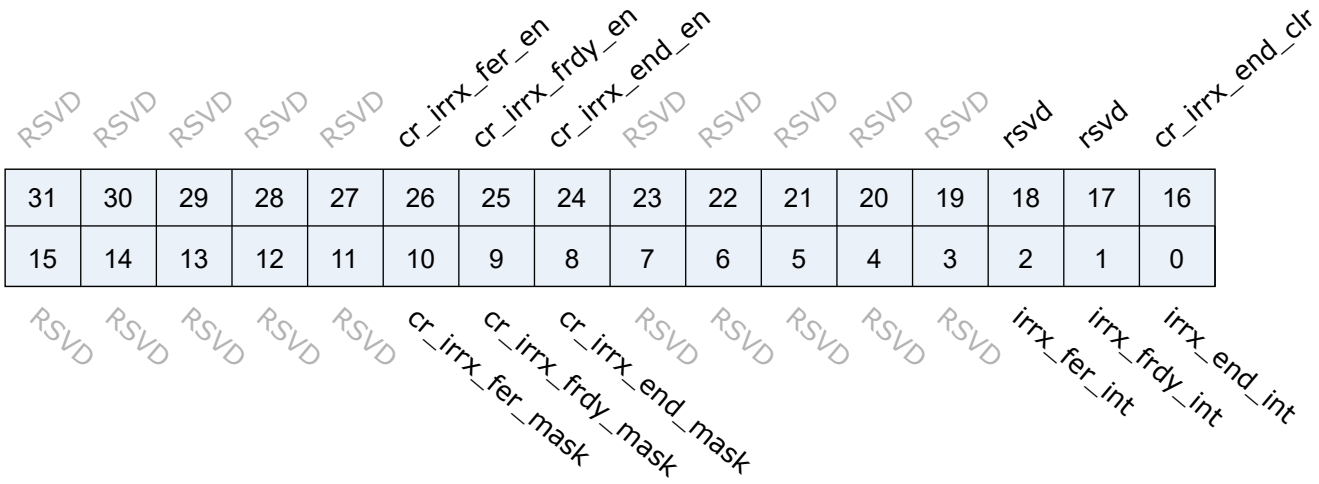
Address：0x2000a640



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	cr_irrx_deg_cnt	r/w	4'd0	De-glitch function cycle count
7:5	RSVD			
4	cr_irrx_deg_en	r/w	1'b0	Enable signal of IRRX input de-glitch function
3:2	cr_irrx_mode	r/w	2'd0	IRRX mode 0: NEC 1: RC5 2: SW pulse-width detection mode (SWM) 3: Reserved
1	cr_irrx_in_inv	r/w	1'b1	Input inverse signal
0	cr_irrx_en	r/w	1'b0	Enable signal of IRRX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

8.4.2 irrx_int_sts

Address：0x2000a644



Bits	Name	Type	Reset	Description
31:27	RSVD			
26	cr_irrx_fer_en	r/w	1'b1	Interrupt enable of irrx_fer_int
25	cr_irrx_frdy_en	r/w	1'b1	Interrupt enable of irrx_frdy_int
24	cr_irrx_end_en	r/w	1'b1	Interrupt enable of irrx_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_irrx_end_clr	w1c	1'b0	Interrupt clear of irrx_end_int
15:11	RSVD			
10	cr_irrx_fer_mask	r/w	1'b1	Interrupt mask of irrx_fer_int
9	cr_irrx_frdy_mask	r/w	1'b1	Interrupt mask of irrx_frdy_int
8	cr_irrx_end_mask	r/w	1'b1	Interrupt mask of irrx_end_int
7:3	RSVD			
2	irrx_fer_int	r	1'b0	IRRX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	irrx_frdy_int	r	1'b0	IRRX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
0	irrx_end_int	r	1'b0	IRRX transfer end interrupt

8.4.3 irrx_pw_config

Address：0x2000a648

cr_irrx_end_th

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_irrx_data_th

Bits	Name	Type	Reset	Description
31:16	cr_irrx_end_th	r/w	16'd8999	Pulse width threshold to trigger END condition
15:0	cr_irrx_data_th	r/w	16'd3399	Pulse width threshold for Logic0/1 detection (Don't care if SWM is enabled)

8.4.4 irrx_data_count

Address：0x2000a650

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD sts_irrx_data_cnt

Bits	Name	Type	Reset	Description
31:7	RSVD			
6:0	sts_irrx_data_cnt	r	7'd0	RX data bit count (pulse-width count for SWM)

8.4.5 irrx_data_word0

Address：0x2000a654

sts_irrx_data_word0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_irrx_data_word0

Bits	Name	Type	Reset	Description
31:0	sts_irrx_data_word0	r	32'h0	RX data word 0

8.4.6 irrx_data_word1

Address：0x2000a658

sts_irrx_data_word1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_irrx_data_word1

Bits	Name	Type	Reset	Description
31:0	sts_irrx_data_word1	r	32'h0	RX data word 1

8.4.7 ir_fifo_config_0

Address：0x2000a680

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	rx_fifo_underflow	rx_fifo_overflow	RSVD	RSVD	rx_fifo_clr	RSVD	RSVD	RSVD

Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5:4	RSVD			
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2:0	RSVD			

8.4.8 ir_fifo_config_1

Address：0x2000a684

RSVD RSVD rx_fifo_th RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD rx_fifo_cnt RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

Bits	Name	Type	Reset	Description
31:30	RSVD			
29:24	rx_fifo_th	r/w	6'd0	RX FIFO threshold, irrx_frdy_int will not be asserted if rx_-fifo_cnt is less than this value
23:15	RSVD			
14:8	rx_fifo_cnt	r	7'd0	RX FIFO available count
7:32	RSVD			
31:0	tx_fifo_wdata	w	32'h0	IRTX FIFO data Normal Mode: Each entry contains a 32-bit data word, LSB is sent first Software Mode: Each entry contains 4 pulse widths, [7:0] is the 1st pulse, [15:8] is the 2nd pulse, etc)

8.4.9 ir_fifo_rdata

Address：0x2000a68c

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

rx_fifo_rdata

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	rx_fifo_rdata	r	16'h0	IRRX FIFO pulse width data for Software Mode

9 SPI

9.1 Overview

Serial Peripheral Interface (SPI) Bus is a synchronous serial communication interface specification for short-range communication. Devices communicate in the full duplex mode, which is a master-slave mode where one master controls one or more slaves.

SPI completes full-duplex communication with 4 signal lines, namely CS (chip select), SCLK (clock), MOSI (master output slave input), and MISO (master input slave output).

9.2 Features

- Can be used as SPI master or slave
- Both master and slave support 4 operating modes (CPOL, CPHA)
- Both master and slave support 1/2/3/4-byte transfer mode
- The sending and receiving channels each have a FIFO with a depth of 32 bytes
- The adaptive FIFO depth variation characteristic suits high-performance applications
  - When the Frame is 32Bits, the depth of FIFO is 8
  - When the Frame is 24Bits, the depth of FIFO is 8
  - When the Frame is 16Bits, the depth of FIFO is 16
  - When the Frame is 8Bits, the depth of FIFO is 32
- Adjustable byte transfer sequence
- Flexible clock configuration, which supports up to 80M clock
- Configurable MSB/LSB transfer priority
- Receive ignore function: You can set to ignore the reception of data from the specified location

- Supports timeout mechanism in the slave mode
- Supports DMA transfer mode

9.3 Functional Description

9.3.1 Clock Control

Due to different clock phases and polarity settings, the SPI clock has four modes, which can be set by cr_spi_sclk_pol (CPOL) and cr_spi_sclk_ph (CPHA) in the register spi_config. CPOL determines the level of SCK clock signal when it is idle. If CPOL=0(cr_spi_sclk_pol=0), the idle level is low, and if CPOL=1(cr_spi_sclk_pol=1), the idle level is high. CPHA determines the sampling time. If CPHA=0(cr_spi_sclk_ph=1), sampling is made at the first lock edge of each cycle, and if CPHA=1(cr_spi_sclk_ph=0), that is made at the second clock edge of each cycle.

By setting the registers spi_prd_0 and spi_prd_1, you can also adjust the duration of the start and end levels of the clock, time of phase 0/1, and interval between frames of data. The settings of the four modes are shown as follows:

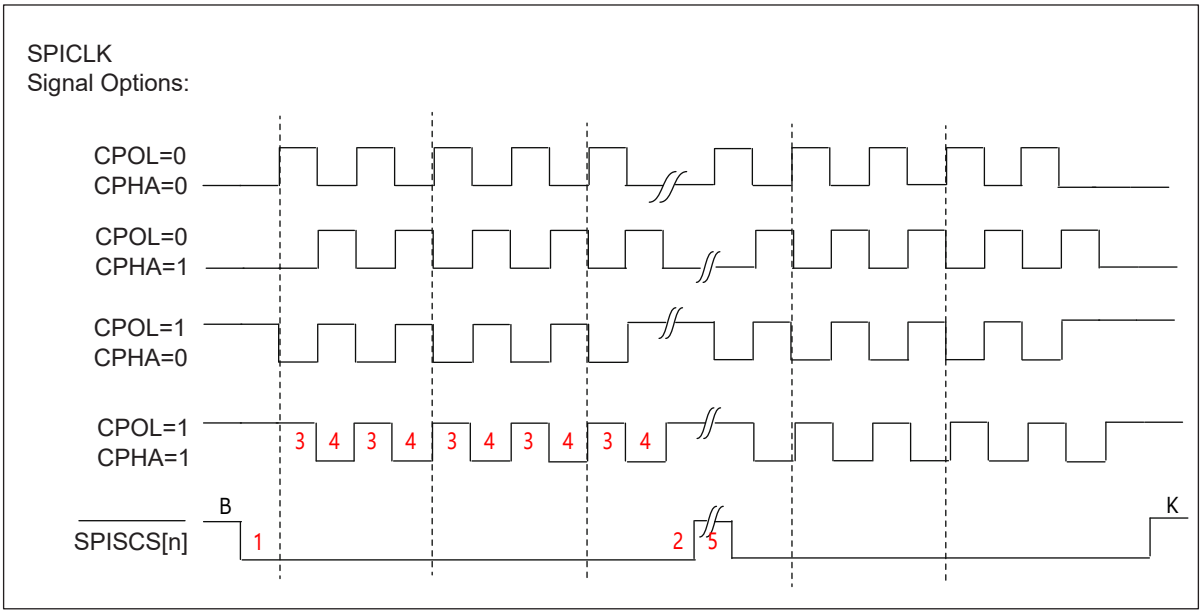


Fig. 9.1: SPI Timing

The meanings of numbers are as follows:

- “1” denotes the length of the START condition, which is configured by the cr_spi_prd_s in the register spi_prd_0.
- “2” denotes the length of the STOP condition, which is configured by the cr_spi_prd_p in the register spi_prd_0.
- “3” denotes the length of phase 0, which is configured by the cr_spi_prd_d_ph_0 in the register spi_prd_0.
- “4” denotes the length of phase 1, which is configured by the cr_spi_prd_d_ph_1 in the register spi_prd_0.
- “5” denotes the interval between frames of data, which is configured by the cr_spi_prd_i in the register spi_prd_1.

9.3.2 Master Continuous Transfer Mode

After this mode is enabled, the CS signal will not be released when the current data is sent and there are still available data in FIFO.

9.3.3 Master-Slave: Transfer and Receive Data

The same framesize shall be set for the master and slave for transferring and receiving data by configuring the cr_spi_frame_size in the register spi_config. When the master and slave agree to communicate at a 32 bits framesize, if the clk of the master does not meet 32 bits due to an exception in a frame of data, the following symptoms occur.

- The data sent by the master cannot be transferred to the RX FIFO of the slave. The slave cannot receive data from the master.
- When the slave sends data, it will skip this frame of data and continue to send the next frame of data when the master’s clk is normal again.

9.3.4 Receive Ignore Function

When the start and end bits to be filtered out are set, SPI will discard the corresponding data segments in the received data, as shown below:

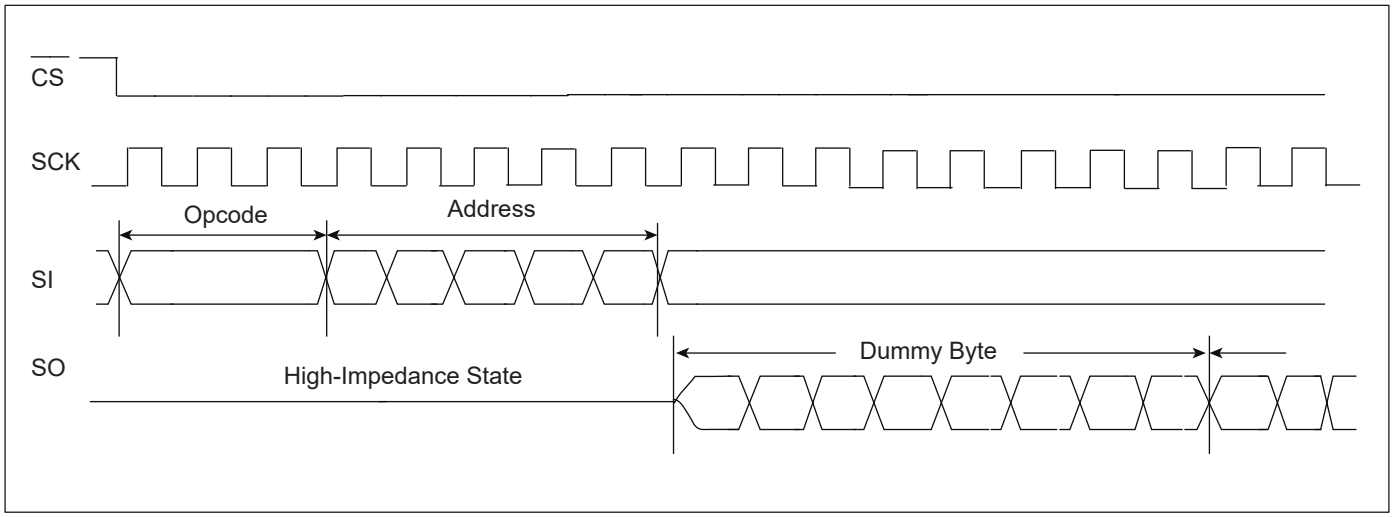


Fig. 9.2: SPI Ignore Waveform

You can enable this function by configuring the cr_spi_rxd_ignr_en in the register spi_config. The start bit of this function is set by configuring the cr_spi_rxd_ignr_s in the register spi_rxd_ignr. The end bit of this function is set by configuring the cr_spi_rxd_ignr_p in the register spi_rxd_ignr.

In the above figure, the start bit to be filtered is set to 0, and if the end bit is set to 7, Dummy Byte will be received; if the end bit is set to 15, Dummy Byte will be discarded.

9.3.5 Filtering Function

When this function is enabled and a threshold is set, SPI will filter the data less than or equal to the width threshold.Assuming that the SPI top clock is 160MHz and the threshold is set to 4, any data with a width below (4/160MHz = 25ns) will be filtered out.

When this function is enabled by setting the cr_spi_deg_en in the register spi_config and the threshold is set by configuring the cr_spi_deg_cnt, SPI will filter out the data that cannot reach the width threshold. As shown in the figure below, the data width is 4, “Input” is the initial data and “output” is the filtered data.

Filtering logic process:

- “Tgl” is the exclusive OR result of input and output.
- “Deg_cnt” counts from 0, and the counting condition is that “tgl” is at the high level and “reached” is at the low level.
- “Reached” means whether the current deg_cnt count reaches the set cr_spi_deg_cnt, and it is at a high level once reached.
- When “reached” is at a high level, “input” is output to “output”.
- Note: user-defined condition for deg_cnt: “tgl” is at a high level and “reached” is at a low level. In other cases, “deg_cnt” will be cleared to 0.

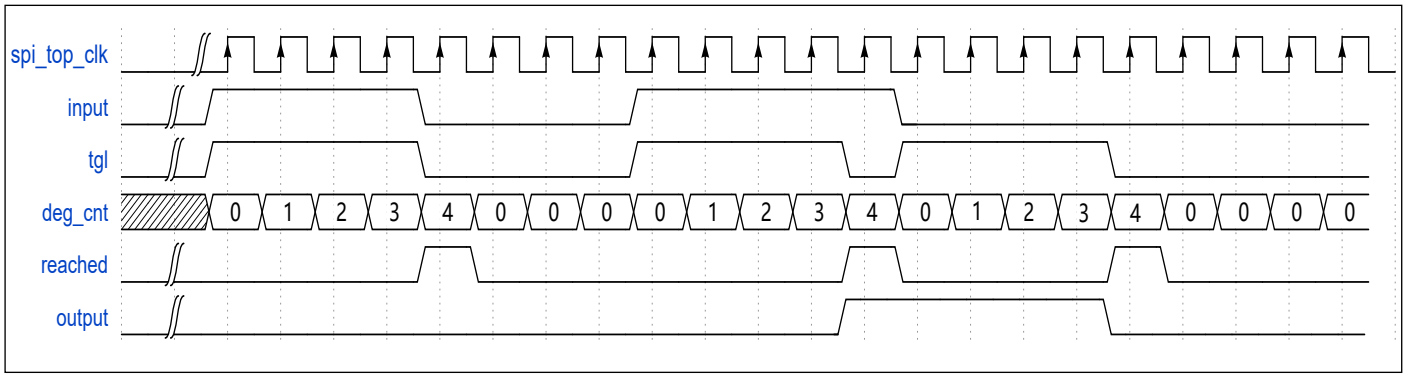


Fig. 9.3: SPI Filter Waveform

9.3.6 Configurable MSB/LSB Transfer

The configurable MSB/LSB transfer mode is limited to the priority transfer sequence of 8 bits in one byte, and the transfer sequence of bits in one byte is set by configuring the cr_spi_bit_inv bit in the register spi_config. 0 indicates MSB and 1 indicates LSB.

For example, for data transfer where the frame size is 24 bits, the data format is Data[23:0]=0×123456.

When MSB transfer is set, the transfer sequence is: 01010110 (binary, 1st byte: 0×56); 00110100 (binary, 2nd byte: 0×34); 00010010 (binary, 3rd byte: 0×12). When LSB transfer is set, the transfer sequence is: 01101010 (binary, 1st byte: 0×56); 00101100 (binary, 2nd byte: 0×34); 01001000 (binary, 3rd byte: 0×12).

9.3.7 Adjustable Byte Transfer Sequence

The adjustable byte transfer sequence is limited to the priority transfer sequence between different bytes in FIFO. The transfer sequence of bytes in FIFO is set by configuring the cr_spi_byte_inv bit in the register spi_config. 0 means sending LSB first, and 1 means sending MSB first.

For example, for data transfer where the frame size is 24 bits, the data format is Data[23:0]=0×123456.

When LSB transfer priority is set, the transfer sequence is 0×56 (1st byte: LSB); 0×34 (2nd byte: intermediate byte); 0×12 (3rd byte: MSB). When MSB transfer priority is set, the transfer sequence is 0×12 (3rd byte: MSB); 0×34 (2nd byte: intermediate byte); 0×56 (1st byte: LSB).

Adjustable byte transfer can be used in conjunction with configurable MSB/LSB transfer.

9.3.8 Slave Mode Timeout Mechanism

When a timeout threshold is set, an interrupt will be triggered when SPI in the slave mode receives no clock signal after the threshold exceeds.

9.3.9 I/O Transfer Mode

The chip communication processor can perform FIFO padding and clearing operations in response to the interrupt from the FIFO. Each FIFO has a programmable FIFO trigger threshold to trigger an interrupt. When rx_fifo_cnt in the register spi_fifo_config_1 is greater than the trigger threshold of rx_fifo_th, an interrupt will be generated to send a signal to the chip communication processor to clear the RX FIFO. When rx_fifo_cnt in the register spi_fifo_config_1 is greater than rx_fifo_th, an interrupt will be generated to send a signal to the chip communication processor to re-pad the TX FIFO.

You can query the SPI status register to determine the sampled value in the FIFO and the FIFO status. The software must provide correct trigger thresholds for RX FIFO and TX FIFO, and prevent the overflow of RX FIFO and the underflow of TX FIFO.

9.3.10 DMA Transfer Mode

SPI supports the DMA transfer mode. To enable this mode, you must set the thresholds of TX FIFO and RX FIFO respectively. Setting spi_dma_tx_en in the register spi_fifo_config_0 to 1 can enable the DMA sending mode. Setting spi_dma_rx_en in the register spi_fifo_config_0 to 1 can enable the DMA receiving mode. When this mode is enabled, UART will check the TX/RX FIFO. Once the tx_fifo_cnt/rx_fifo_cnt in the register spi_fifo_config_1 is greater than tx_-fifo_th/rx_fifo_th, a DMA request will be initiated, and DMA will transfer data into TX FIFO or remove data from RX FIFO as configured.

9.3.11 SPI Interrupt

SPI supports the following interrupt control modes:

- SPI end of transfer interrupt
  - In the master mode, the SPI end of transfer interrupt will be triggered when the transfer of each frame of data ends.
  - In the slave mode, that interrupt is triggered when the CS signal is released.
- TX FIFO request interrupt
  - The TX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold, and the interrupt flag will be cleared automatically when the condition is unmet.
- RX FIFO request interrupt
  - The RX FIFO request interrupt will be triggered when the FIFO available count value is greater than the preset threshold, and the interrupt flag will be cleared automatically when the condition is unmet.
- Slave mode transfer timeout interrupt
  - The slave mode transfer timeout interrupt will be triggered when no clock signal is received in the slave mode after the threshold exceeds. If the TX/RX FIFO overflows or underflows, it will trigger the TX/RX FIFO overflow interrupt.
- Slave mode TX overload interrupt
  - Slave mode TX overload interrupt will trigger when the TX is not ready to transmit in slave mode.
- TX/RX FIFO overflow interrupt
  - TX/RX FIFO overflow interrupt is triggered if an overflow or underflow occurs in the TX/RX FIFO. When the tx_fifo_clr/rx_fifo_clr bit in the FIFO clear register spi_fifo_config_0 is set to 1, the corresponding FIFO will be cleared and the overflow interrupt flag will be cleared automatically.

You can query the interrupt status through the register SPI_INT_STS and write 1 to the corresponding bit to clear the interrupt.

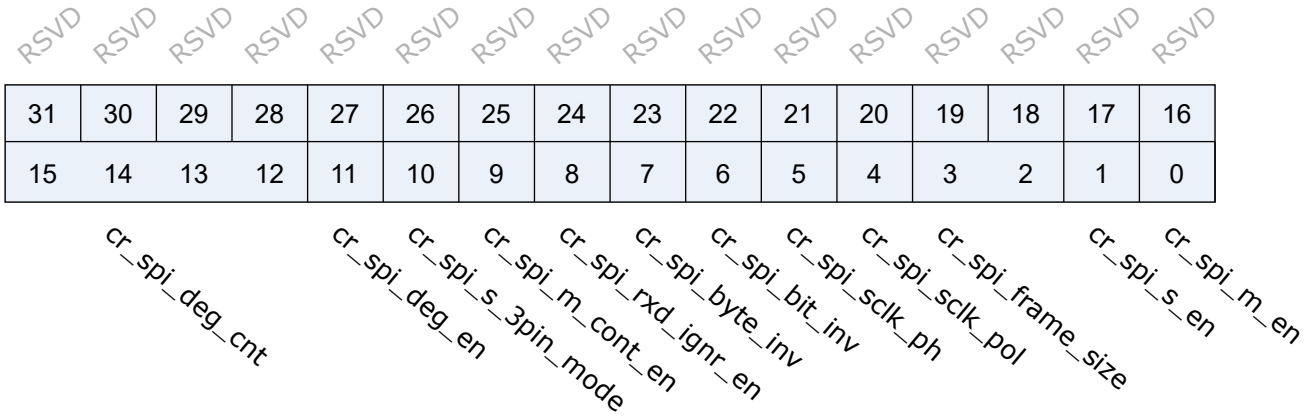
9.4 Register description

Name	Description
spi_config	Master and slave configure
spi_int_sts	Interrupt configure and status
spi_bus_busy	Bus busy status

Name	Description
spi_prd_0	Period configure 0
spi_prd_1	Period configure 1
spi_rxd_ignr	RX ignore function
spi_sto_value	Timer-out value setting
spi_fifo_config_0	FIFO status and DMA mode
spi_fifo_config_1	FIFO threshold and available count
spi_fifo_wdata	TX FIFO
spi_fifo_rdata	RX FIFO
backup_io_en	IO backup

9.4.1 spi_config

Address：0x40019000

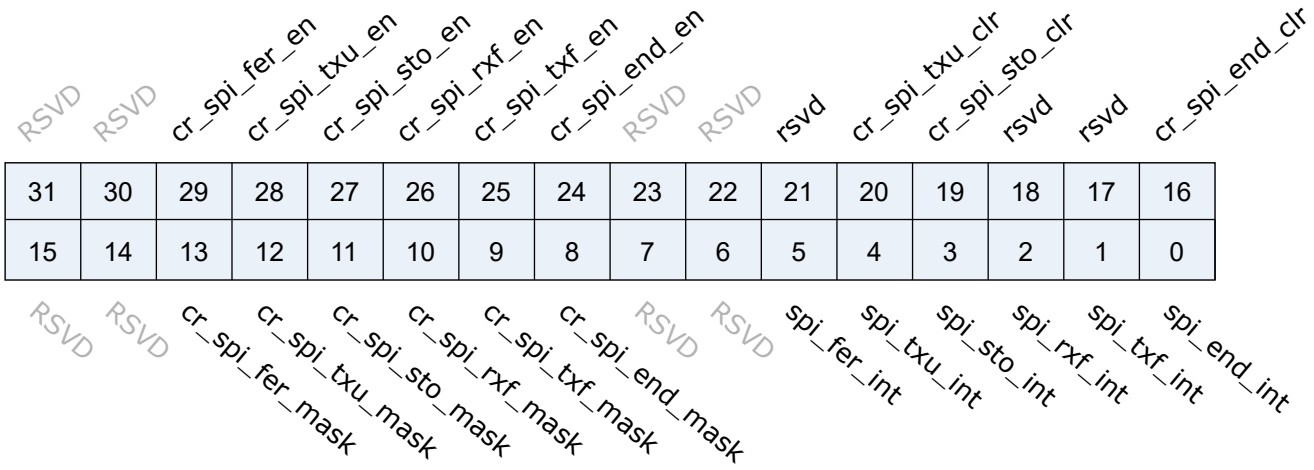


Bits	Name	Type	Reset	Description
31:16	RSVD			
15:12	cr_spi_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_spi_deg_en	r/w	1'b0	Enable signal of all input de-glitch function
10	cr_spi_s_3pin_mode	r/w	1'b0	SPI slave 3-pin mode 1'b0: 4-pin mode (SS_n is enabled) 1'b1: 3-pin mode (SS_n is disabled / don't care)

Bits	Name	Type	Reset	Description
9	cr_spi_m_cont_en	r/w	1'b0	Enable signal of master continuous transfer mode 1'b0: Disabled, SS_n will de-assert between each data frame 1'b1: Enabled, SS_n will stay asserted between each consecutive data frame if the next data is valid in the FIFO
8	cr_spi_rxd_ignr_en	r/w	1'b0	Enable signal of RX data ignore function
7	cr_spi_byte_inv	r/w	1'b0	Byte-inverse signal for each FIFO entry data 0: Byte[0] is sent out first 1: Byte[3] is sent out first(32-bit frame size) / byte[2] is send out first(24-bit frame size) / byte[1] is send out first(16-bit frame size)
6	cr_spi_bit_inv	r/w	1'b0	Bit-inverse signal for each data byte 0: Each byte is sent out MSB-first 1: Each byte is sent out LSB-first
5	cr_spi_sclk_ph	r/w	1'b0	SCLK clock phase inverse signal 0: Data is sampled on the second edge of SCLK 1: Data is sampled on the first edge of SCLK
4	cr_spi_sclk_pol	r/w	1'b0	SCLK polarity 0: SCLK output LOW at IDLE state 1: SCLK output HIGH at IDLE state
3:2	cr_spi_frame_size	r/w	2'd0	SPI frame size (also the valid width for each FIFO entry) 2'd0: 8-bit, FIFO space is 1*32 = 32 byte 2'd1: 16-bit, FIFO space is 2*16 = 32 byte 2'd2: 24-bit, FIFO space is 3*8 = 24 byte 2'd3: 32-bit, FIFO space is 4*8 = 32 byte
1	cr_spi_s_en	r/w	1'b0	Enable signal of SPI Slave function, Master and Slave should not be both enabled at the same time (This bit becomes don't-care if cr_spi_m_en is enabled)
0	cr_spi_m_en	r/w	1'b0	Enable signal of SPI Master function Asserting this bit will trigger the transaction, and should be de-asserted after finish

9.4.2 spi_int_sts

Address：0x40019004

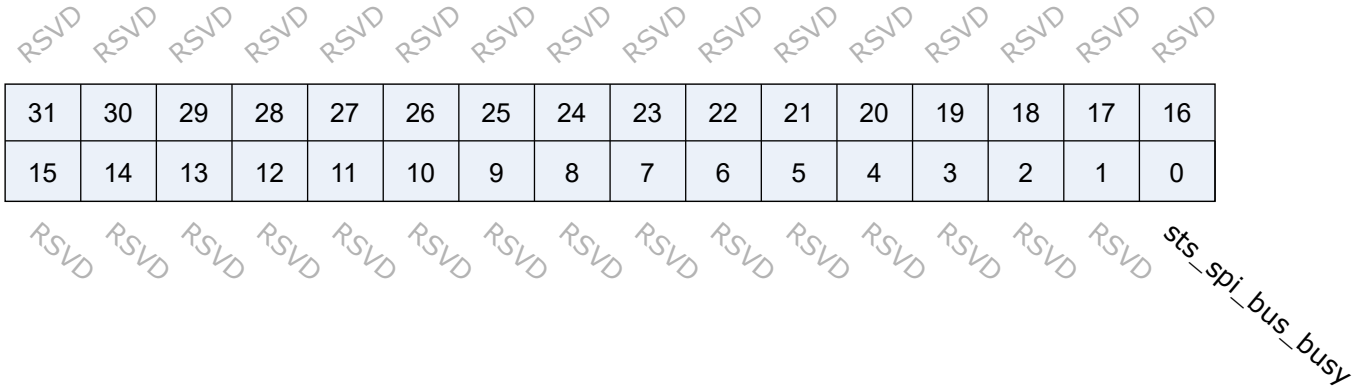


Bits	Name	Type	Reset	Description
31:30	RSVD			
29	cr_spi_fer_en	r/w	1'b1	Interrupt enable of spi_fer_int
28	cr_spi_txu_en	r/w	1'b1	Interrupt enable of spi_txu_int
27	cr_spi_sto_en	r/w	1'b1	Interrupt enable of spi_sto_int
26	cr_spi_rxf_en	r/w	1'b1	Interrupt enable of spi_rxv_int
25	cr_spi_txf_en	r/w	1'b1	Interrupt enable of spi_txe_int
24	cr_spi_end_en	r/w	1'b1	Interrupt enable of spi_end_int
23:22	RSVD			
21	rsvd	rsvd	1'b0	
20	cr_spi_txu_clr	w1c	1'b0	Interrupt clear of spi_txu_int
19	cr_spi_sto_clr	w1c	1'b0	Interrupt clear of spi_sto_int
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_spi_end_clr	w1c	1'b0	Interrupt clear of spi_end_int
15:14	RSVD			
13	cr_spi_fer_mask	r/w	1'b1	Interrupt mask of spi_fer_int
12	cr_spi_txu_mask	r/w	1'b1	Interrupt mask of spi_txu_int
11	cr_spi_sto_mask	r/w	1'b1	Interrupt mask of spi_sto_int
10	cr_spi_rxf_mask	r/w	1'b1	Interrupt mask of spi_rxv_int
9	cr_spi_txf_mask	r/w	1'b1	Interrupt mask of spi_txe_int

Bits	Name	Type	Reset	Description
8	cr_spi_end_mask	r/w	1'b1	Interrupt mask of spi_end_int
7:6	RSVD			
5	spi_fer_int	r	1'b0	SPI TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
4	spi_txu_int	r	1'b0	SPI slave mode TX underrun error flag, triggered when TXD is not ready during transfer in slave mode
3	spi_sto_int	r	1'b0	SPI slave mode transfer time-out interrupt, triggered when SPI bus is idle for a given value
2	spi_rxf_int	r	1'b0	SPI RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
1	spi_txf_int	r	1'b1	SPI TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	spi_end_int	r	1'b0	SPI transfer end interrupt, shared by both master and slave mode Master mode: Triggered when the final frame is transferred Slave mode: Triggered when CS_n is de-asserted

9.4.3 spi_bus_busy

Address：0x40019008



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	sts_spi_bus_busy	r	1'b0	Indicator of SPI bus busy 0: Idle 1: Busy

9.4.4 spi_prd_0

Address：0x40019010

cr_spi_prd_d_ph_1								cr_spi_prd_d_ph_0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_spi_prd_p								cr_spi_prd_s							

Bits	Name	Type	Reset	Description
31:24	cr_spi_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 (unit: SPI source clock period)
23:16	cr_spi_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (unit: SPI source clock period)
15:8	cr_spi_prd_p	r/w	8'd15	Length of STOP condition (unit: SPI source clock period)
7:0	cr_spi_prd_s	r/w	8'd15	Length of START condition (unit: SPI source clock period)

9.4.5 spi_prd_1

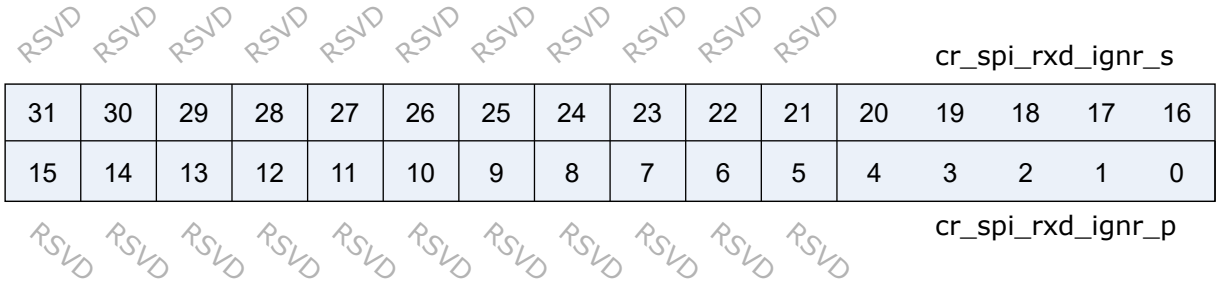
Address：0x40019014

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	cr_spi_prd_i							

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_spi_prd_i	r/w	8'd15	Length of INTERVAL between frame (unit: SPI source clock period)

9.4.6 spi_rxd_ignr

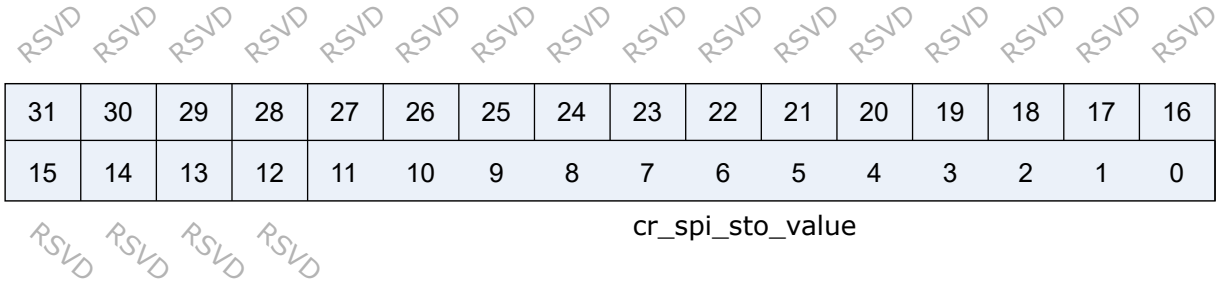
Address：0x40019018



Bits	Name	Type	Reset	Description
31:21	RSVD			
20:16	cr_spi_rxd_ignr_s	r/w	5'd0	Starting point of RX data ignore function (unit: bit)
15:5	RSVD			
4:0	cr_spi_rxd_ignr_p	r/w	5'd0	Stopping point of RX data ignore function (unit: bit)

9.4.7 spi_sto_value

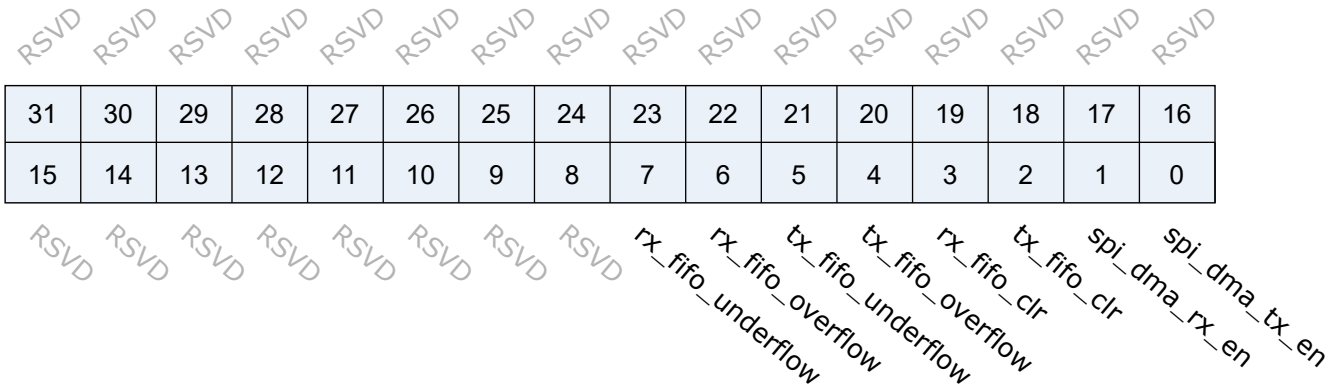
Address：0x4001901c



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:0	cr_spi_sto_value	r/w	12'hFFF	Time-out value for spi_sto_int triggering

9.4.8 spi_fifo_config_0

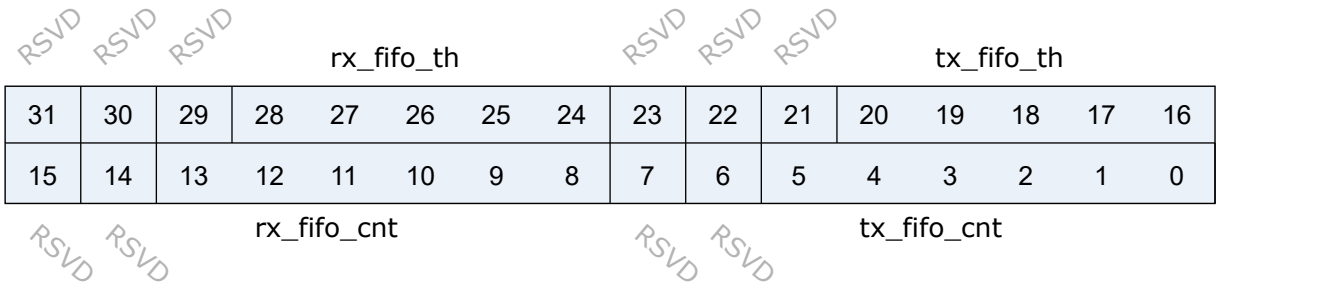
Address：0x40019080



Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO, RX FIFO will be empty when write 1 to this bit
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO, TX FIFO will be empty when write 1 to this bit
1	spi_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	spi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

9.4.9 spi_fifo_config_1

Address：0x40019084



Bits	Name	Type	Reset	Description
31:29	RSVD			
28:24	rx_fifo_th	r/w	5'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_-fifo_cnt is less than this value
23:21	RSVD			
20:16	tx_fifo_th	r/w	5'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_-fifo_cnt is less than this value
15:14	RSVD			
13:8	rx_fifo_cnt	r	6'd0	RX FIFO available count, means byte count of data received in RX FIFO (unit: byte)
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count, means empty space remained in TX FIFO (unit: byte)

9.4.10 spi_fifo_wdata

Address：0x40019088

spi_fifo_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

spi_fifo_wdata

Bits	Name	Type	Reset	Description
31:0	spi_fifo_wdata	w	x	TX FIFO write data port Note: Partial valid if cr_spi_frame_size is set to different value: 2'd0 (8-bit frame): Only [7:0] are valid and [31:8] are don't-care 2'd1 (16-bit frame): Only [15:0] are valid and [31:16] are don't-care 2'd2 (24-bit frame): Only [23:0] are valid and [31:24] are don't-care 2'd3 (32-bit frame): Entire [31:0] are valid

9.4.11 spi_fifo_rdata

Address：0x4001908c

spi_fifo_rdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

spi_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	spi_fifo_rdata	r	32'h0	RX FIFO read data port Note: Partial valid if cr_spi_frame_size is set to different value: 2'd0 (8-bit frame): Only [7:0] are valid and [31:8] are all 0s 2'd1 (16-bit frame): Only [15:0] are valid and [31:16] are all 0s 2'd2 (24-bit frame): Only [23:0] are valid and [31:24] are all 0s 2'd3 (32-bit frame): Entire [31:0] is valid

9.4.12 backup_io_en

Address：0x400190fc

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD backup_io_en

Bits	Name	Type	Reset	Description
31:1	RSVD			
0	backup_io_en	r/w	1'b0	Enable IO backup function

10 UART

10.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) provides a flexible way to exchange full-duplex data with external devices. BL616/BL618 is provided with 2 UARTs, which can be used together with DMA to achieve efficient data communication.

10.2 Features

- Full-duplex asynchronous communication
- Optional data bit length: 5/6/7/8-bit
- Optional stop bit length: 0.5/1/1.5/2-bit
- Supports odd/even/none check bit
- Error-detectable start bit
- Abundant interrupt control modes
- Hardware flow control (RTS/CTS)
- Convenient baud rate programming
- Configurable MSB/LSB transfer priority
- Automatic baud rate detection of ordinary/fixed characters
- 32-byte TX/RX FIFO
- Supports DMA transfer mode
- Supports baud rate of 10 Mbps and below
- Supports the LIN bus protocol

- Supports the RS485 mode
- Optional clock sources: 160M/BCLK/XCLk
- Support filter function

10.3 Functional Description

10.3.1 Data Formats

The normal UART communication data consists of start bits, data bits, parity check bits, and stop bits. The UART of xx supports configurable data bits, parity check bits, and stop bits, which are set in registers utx_config and urx_config. The waveform of a frame of data is shown as follows:

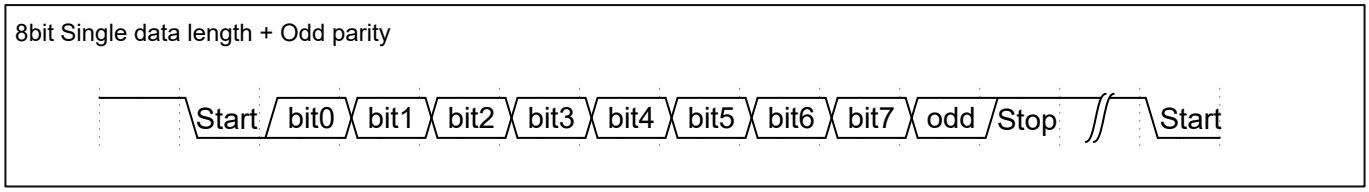


Fig. 10.1: UART Data Format

The start bit of the data frame occupies 1 bit, and the stop bit can be 0.5/1/1.5/2-bit wide by configuring the cr_utx_bit_cnt_p bit in the register utx_config. The start bit is at a low level and the stop bit is at a high level. The data bit width can be set to 5/6/7/8-bit by the cr_utx_bit_cnt_d bit in the register utx_config.

When the cr_utx_prt_en bit in the register utx_config and the cr_urx_prt_en bit in the register urx_config are set, the data frame adds a parity check bit after the data. The cr_utx_prt_sel bit in the register utx_config and the cr_urx_prt_sel bit in the register urx_config are used to select odd or even parity check. When the receiver detects the check bit error of the input data, it will generate the check error interrupt. However, the received data will still be stored into the FIFO.

Calculation method of odd parity check: If there is an odd number of “1” in the current data bit, the odd parity check bit is set to 0. Otherwise, it is set to 1.

Calculation method of even parity check: If there is an odd number of “1” in the current data bit, the even parity check bit is set to 1. Otherwise, it is set to 0.

10.3.2 Basic Architecture

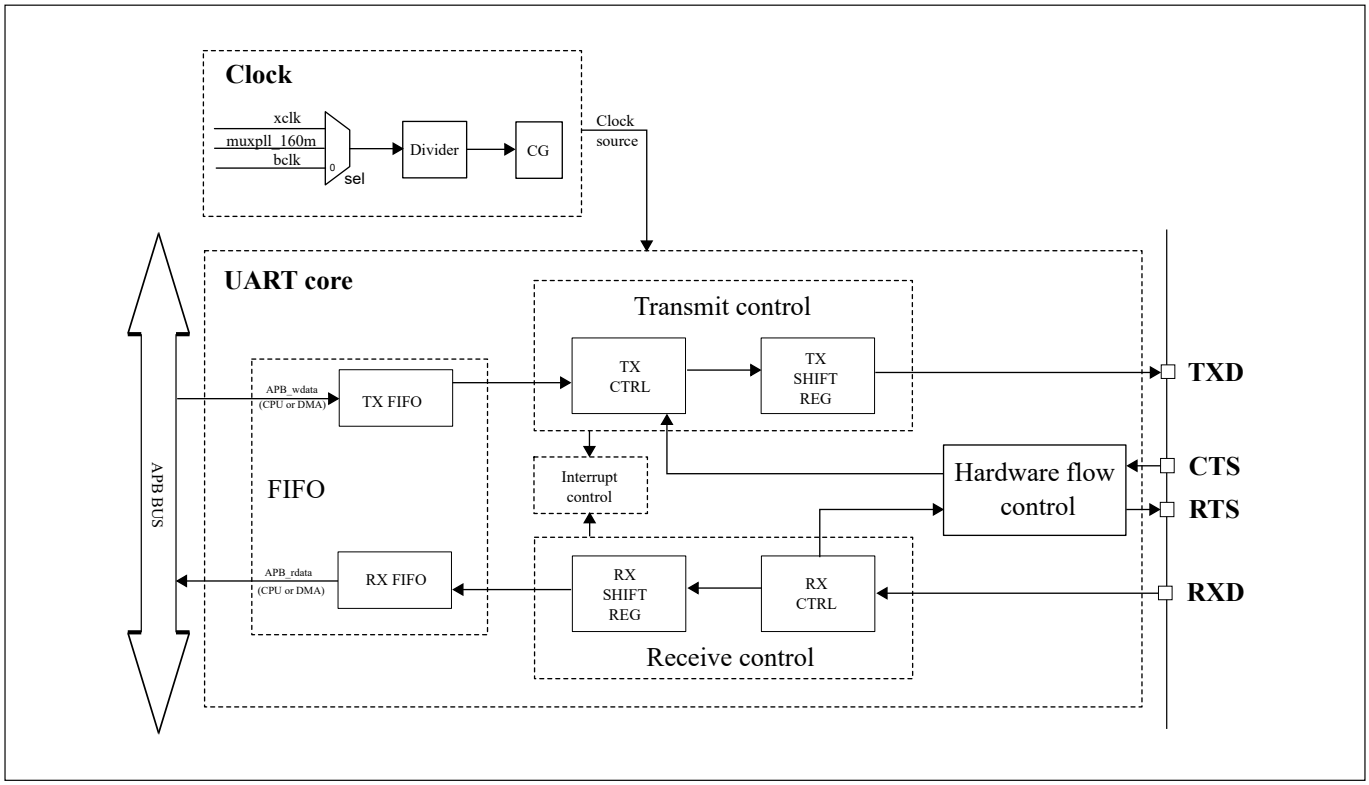


Fig. 10.2: UART Architecture

The UART has 3 clock sources: XCLK, 160MHz CLK and BCLK. The frequency divider in the clock is used to divide the frequency of the clock source and then generate the clock signal to drive the UART module.

The UART controller is divided into two functional blocks: transmitter and receiver.

10.3.3 Transmitter

The transmitter contains a 32-byte TX FIFO to store the data to be sent. When the transmission enable bit is set, the data stored in FIFO will be output from the TX pin. Software can transfer data into TX FIFO through DMA or APB bus. Software can check the status of transmitter by querying the remaining free space count value of TX FIFO through tx_fifo_cnt in the register uart_fifo_config_1.

FreeRun mode of transmitter:

- If the FreeRun mode is disabled, transmission will be terminated and an interrupt will be generated when the sent bytes reach the specified length. Before next transmission, you need to re-disable and enable the TxE bit.
- If the FreeRun mode is enabled, the transmitter will send when there is data in the TX FIFO, and will not stop working because the sent bytes reach the specified length.

10.3.4 Receiver

The receiver contains a 32-byte RX FIFO to store the received data. Software can check the status of receiver by querying the available data count value of RX FIFO through rx_fifo_cnt in the register uart_fifo_config_1. The low 8 bits of the register URX_RTO_TIMER are used to set a receiving timeout threshold, which will trigger an interrupt when the receiver fails to receive data beyond the threshold. The cr_urx_deg_en and cr_urx_deg_cnt in the register urx_config are used to enable the deburring function and set the threshold, which control the filtering part before sampling by UART. UART will filter out the burrs whose width is lower than the threshold in the waveform and then send it to sampling.

10.3.5 Baud Rate Setting

$$Baudrate = \frac{UART\_clk}{uart\_prd + 1}$$

The user can set the baud rate of RX and TX separately. Take TX as an example: the value of uart_prd is the value of the lower 16 bits cr_utx_bit_prd of the register UART_BIT_PRD. Since the maximum value of the 16-bit bit width coefficient is 65535, the minimum baud rate supported by UART is : UART_clk/65536.

Before sampling the data, UART will filter the data to remove the burrs in the waveform. Then, the data will be sampled at the intermediate value of the 16-bit width factor, so that the sampling time can be adjusted based on baud rates, to ensure that the intermediate value is always sampled, providing much higher flexibility and accuracy. The sampling process is shown as follows:

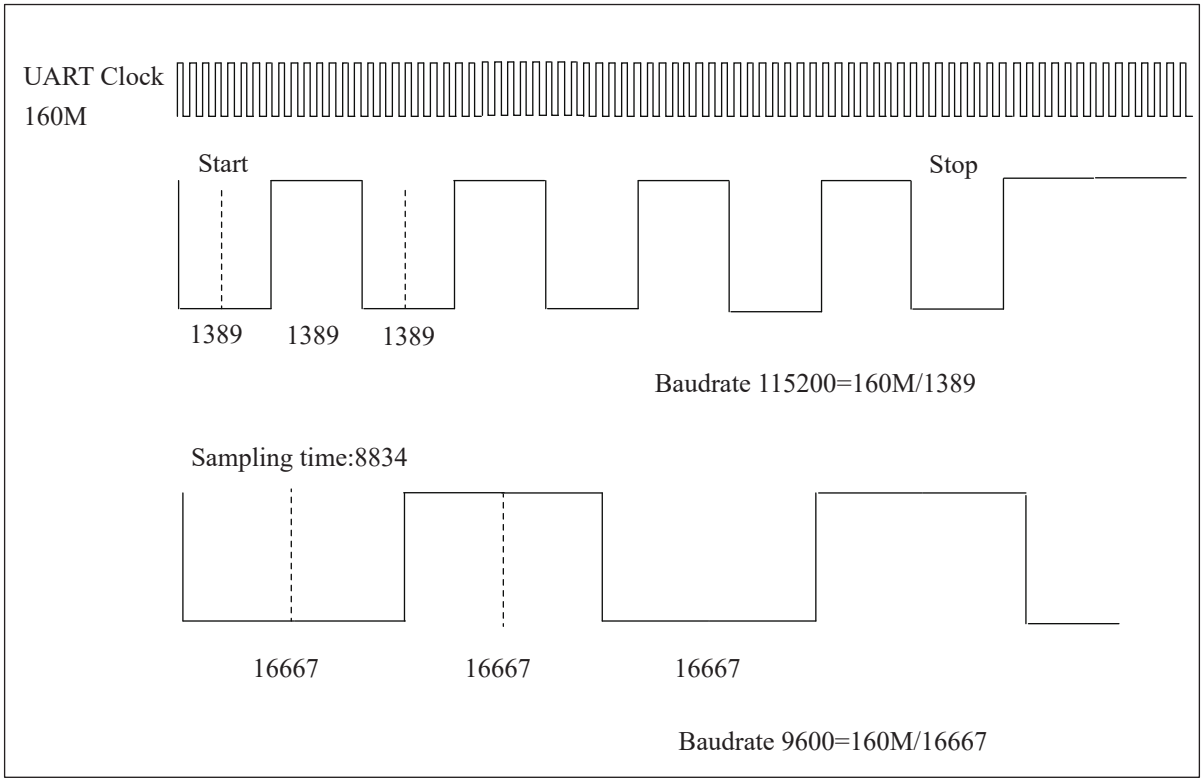


Fig. 10.3: UART Sampling Waveform

10.3.6 Filtering

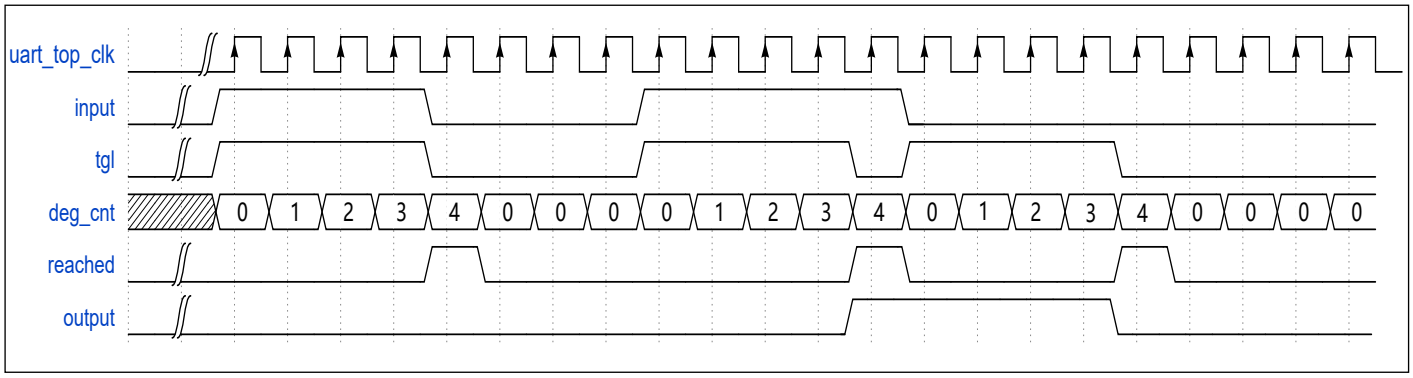


Fig. 10.4: UART Filter Waveform

When this function is enabled by configuring cr_urx_deg_en and the threshold is set by configuring cr_urx_deg_cnt in the register urx_config, UART will filter out the data that cannot meet the width threshold. As shown in the figure below, when the data width is 4, setting cr_urx_deg_cnt to 4 can meet this condition. “Input” is the initial data and “output” is the filtered data.

Filtering logic process:

- “Tgl” is the exclusive OR result of input and output.
- “Deg_cnt” counts from 0, and the counting condition is that “tgl” is at the high level and “reached” is at the low level.
- If the count value of deg_cnt reaches the value set by cr_urx_deg_cnt, reached is high level.
- When “reached” is at a high level, “input” is output to “output”.
- Note: user-defined condition for deg_cnt: “tgl” is at a high level and “reached” is at a low level. In other cases, “deg_cnt” will be cleared to 0.

10.3.7 Automatic Baud Rate Detection

The UART module supports automatic baud rate detection, which is divided into two modes, a generic mode and a fixed character (square wave) mode. The cr_urx_abr_en in the urx_config register enables auto baud rate detection, and when it is turned on, both detection modes are enabled.

Generic mode

For any character data received, UART will count the number of clocks in the start bit width, which will then be written into the low 16 bits sts_urx_abr_prd_start in the register STS_URX_ABR_PRD and used to calculate the baud rate. So the correct baud rate can be obtained when the first received data bit is 1, such as ‘0x01’ under LSBFIRST.

Fixed character (square wave) mode

In this mode, after the UART module counts the number of clocks in the bit width, it will continue to count the number of clocks in the subsequent data bits and compare it with the start bit. The check is passed, otherwise the count value is discarded. The allowable error can be set by setting the cr_urx_abr_pw_tol bit in the register urx_abr_pw_tol, and the unit is the clock source of the UART.

Therefore, only when the fixed character ‘0x55’/’0xD5’ under LSB-FIRST or ‘0xAA’/’0xAB’ under MSB-FIRST is received, the UART module will write the clock count value in the starting bit width into the high 16-bit sts_urx_abr_prd_0x55 of the register STS_URX_ABR_PRD. As shown below:

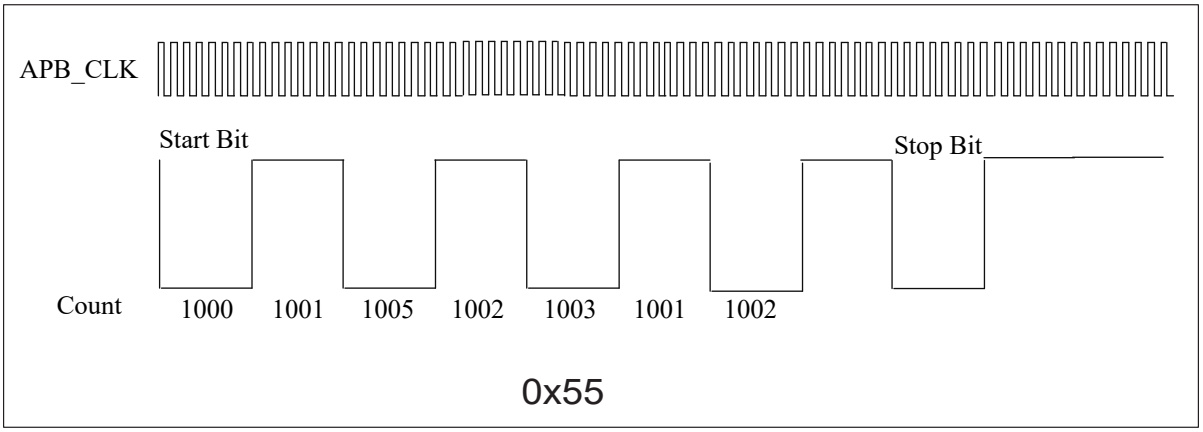


Fig. 10.5: Waveform of UART in fixed character mode

As shown above, assuming the maximum allowable error set is 4, for a received data with unknown baud rate, the UART uses UART_CLK to count the bit width of the starting bit as 1000, the bit width of the second bit as 1001, which is not more than 4 UART_CLK up or down from the previous bit width, then the UART will continue to count the third bit. The third bit is 1005, the difference with the starting bit is more than 4, the detection is not passed and the data is discarded. UART compares the first 6 bit widths of the data bits with the starting bit in turn.

Formula for calculating the detected baud rate:

$$Baudrate = \frac{UART\_clk}{Count + 1}$$

10.3.8 Hardware Flow Control

UART supports hardware flow control in CTS/RTS mode to prevent data in FIFO from being lost due to too late processing. Hardware flow control connection is shown as follows:

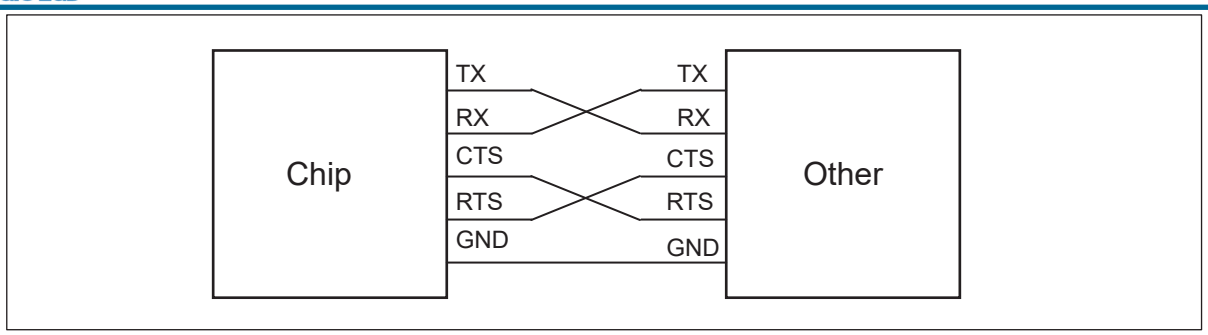


Fig. 10.6: UART hardware flow control

Require To Send (RTS) is an output signal, which indicates whether the chip is ready to receive data from the other side. This is valid at a low level denoting that the chip can receive data.

Clear To Send (CTS) is an input signal, which determines whether the chip can send data to the other side. This is valid at a low level denoting that the chip can send data to the other side.

When the hardware flow control function is enabled, the low level of chip’ s RTS indicates requesting the other side to send data, and the high level of that indicates informing the other side to stop sending data. When the chip detects that CTS goes high, TX will stop sending data, and continue sending until CTS goes low. If CTS goes high or low at any time during communication, it does not affect the continuity of data sent by TX, and the other side also can receive continuous data.

Two ways for hardware flow control of the transmitter:

- Hardware control (the cr_urx_rts_sw_mode in the register uart_sw_mode is 0): RTS goes high when cr_urx_en in the register urx_config is not turned on or the RX FIFO is almost full (one byte left).
- Software control(the cr_urx_rts_sw_mode in the register uart_sw_mode is 1): The level of RTS can be changed by configuring cr_urx_rts_sw_val in the register uart_sw_mode.

10.3.9 DMA Transfer

UART supports DMA transfer. Using DMA transfer, the TX and RX FIFO thresholds need to be set respectively by tx_fifo_th and rx_fifo_th in register uart_fifo_config_1. When this mode is enabled, if tx_fifo_cnt in uart_fifo_config_1 is greater than tx_fifo_th, a DMA TX request will be triggered. After the DMA is configured, when the DMA receives the request, it will move the data from the memory to the TX FIFO according to the settings. If the rx_fifo_cnt in uart_-fifo_config_1 is greater than rx_fifo_th, the DMA RX request will be triggered. After the DMA is configured, when the DMA receives the request, it will transfer the data of the RX FIFO to the memory according to the settings.

In order to ensure the correctness of the data transferred by the chip DMA TX Channel, the following conditions need to be met in the Channel configuration: (transferWidth * burstSize) ≤ (tx_fifo_th + 1).

In order to ensure the integrity of the data transferred by the chip DMA RX Channel, the following conditions need to be met in the Channel configuration: (transferWidth * burstSize) = (rx_fifo_th + 1).

10.3.10 Support for LIN Bus

The protocol for the Local Interconnect Network (LIN) is based on the Volcano-Lite technology developed by the Volvo spin-out company—Volcano Communications Technology (VCT). LIN is a complementary protocol to CAN and SAE J1850, suitable for applications that have low requirement for time or require no precise fault tolerance (as LIN is not as reliable as CAN). LIN aims to be easy to use as a low-cost alternative to CAN. The vehicle parts where LIN can be used include window regulator, rearview mirror, wiper, and rain sensor.

UART supports the LIN bus mode.The LIN bus is under the master-slave mode, and data is always initiated by the master node. The frame (header) sent by the master node contains synchronization interval field, synchronization byte field, and identifier field. A typical LIN data transfer is shown as follows.

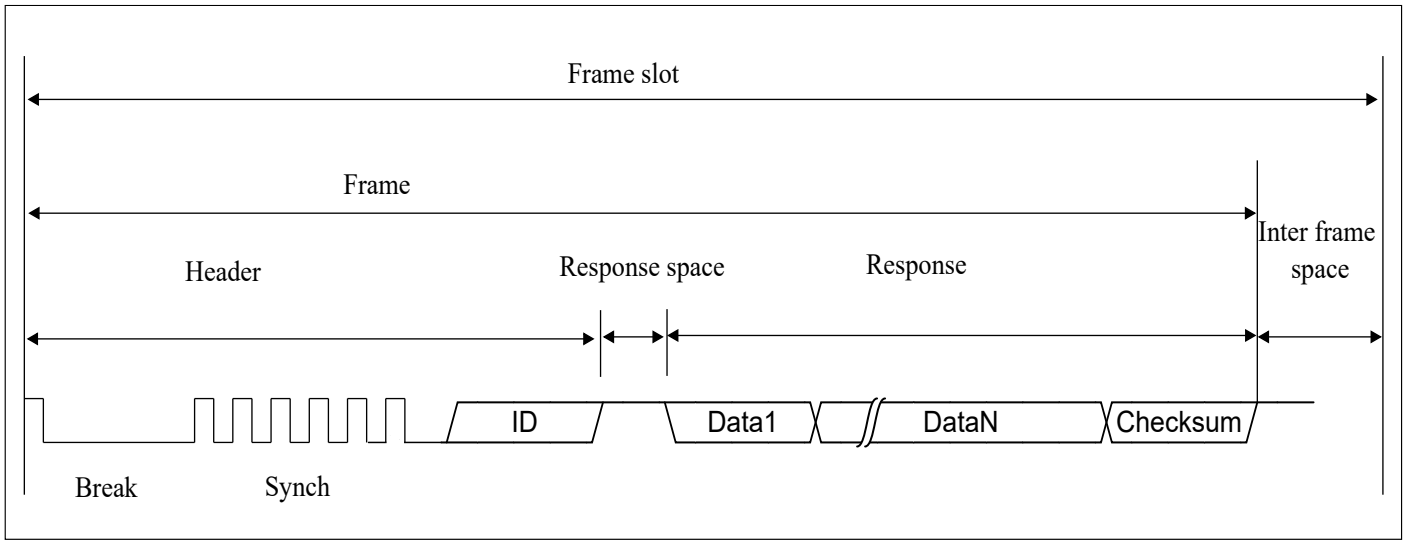


Fig. 10.7: A typical LIN frame

1. LIN Break Field

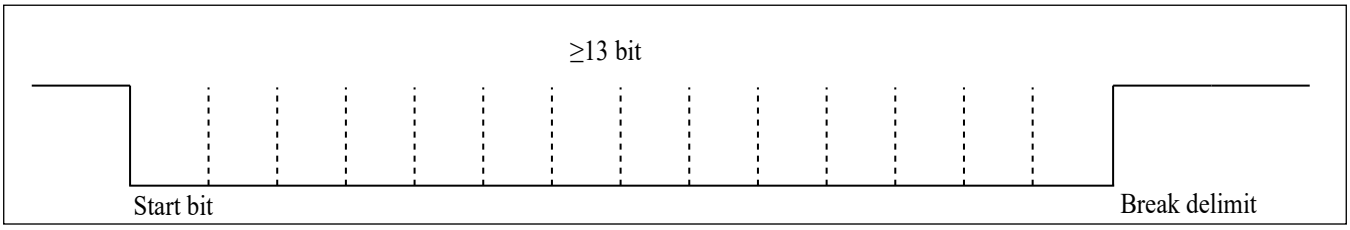


Fig. 10.8: Break Field of LIN

The synchronization interval field indicates the start of the message, with at least 13 dominant bits (including the start bit). The synchronization interval ends with an “interval separator”, which contains at least one recessive bit.

The length of the Break in the LIN frame can be set by cr_utx_bit_cnt_b in utx_config.

2. LIN Sync Field

A synchronization byte field is sent to determine the time between two falling edges, to determine the transmission

rate used by the master node. The bit pattern is 0x55 (01010101, maximum number of falling edges).

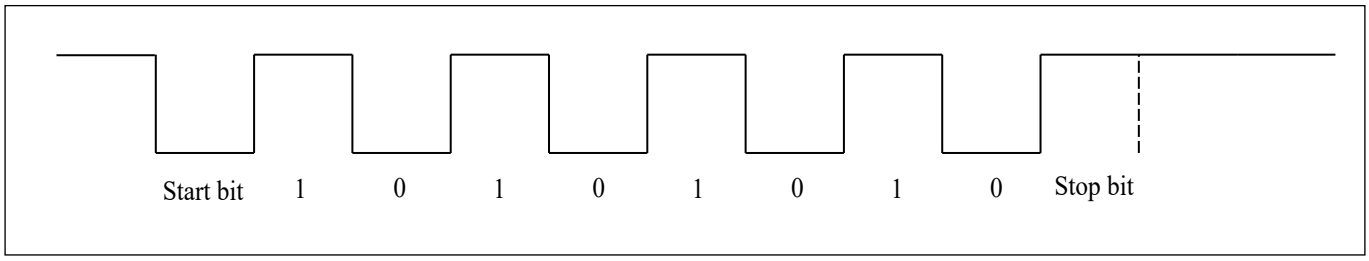


Fig. 10.9: Sync Field of LIN

3. LIN ID Field

The identifier field contains a 6-bit identifier and two parity check bits. The 6-bit identifier contains information about the sender and receiver, and the number of bytes required in the response. The parity check bit is calculated as follows: The check bit P0 is the result of logical OR operation among ID0, ID1, ID2, and ID4. The check bit P1 is the result of inversion after logical OR operation among ID1, ID3, ID4, and ID5.

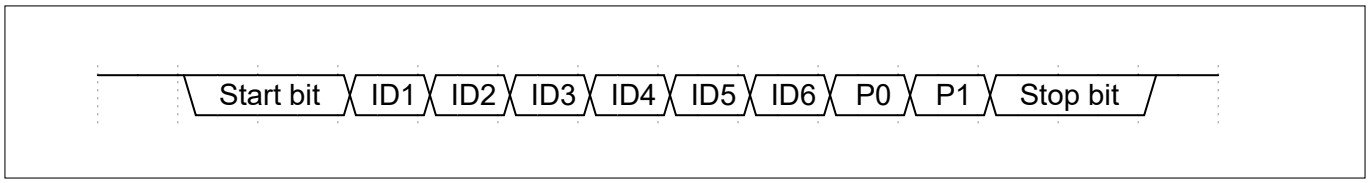


Fig. 10.10: ID Field of LIN

The slave node waits for the synchronization interval field, and then starts to synchronize between master and slave nodes through the synchronization byte field. Depending on the identifier sent by the master node, the slave node will receive, send or do not respond. The slave node that should send data sends the number of bytes requested by the master node, and then ends the transmission with a checksum field.

UART supports the LIN transfer mode. To enable this mode, you need to configure the cr_utx_bit_cnt_b by setting cr_utx_lin_en in the register utx_config so that the synchronization interval field consists of at least a 13-bit dominant level.

10.3.11 RS485 mode

UART supports the RS485 mode. After the cr_utx_rs485_en in the register UTX_RS485_CFG is set, UART can work in the RS485 mode. Then, UART can be connected to the RS485 bus through an external RS485 transceiver. In this mode, the RTS pin in the module performs the Dir function of transceiver. When UART has data to send, it will automatically control the RTS pin at a high level, so that the transceiver can send data to the bus. Contrarily, when UART has no data to send, it will automatically control the RTS at a low level, to keep the transceiver in the RX state.

UART supports the RS485 transfer mode. To enable this mode, you need to set cr_utx_rs485_pol and cr_utx_rs485_-en in the register UTX_RS485_CFG.

10.3.12 UART Interrupt

UART supports the following interrupt control modes:

- TX end of transfer interrupt
- RX end interrupt
- TX FIFO request interrupt
- RX FIFO request interrupt
- RX timeout interrupt
- RX parity check error interrupt
- TX FIFO overflow interrupt
- RX FIFO overflow interrupt
- RX BCR interrupt
- LIN synchronization error interrupt
- Auto baud rate detection (universal mode) interrupt
- Auto baud rate detection (fixed characters mode) interrupt

10.3.13 TX/RX end of transfer interrupt

You can set a transfer length for TX and RX respectively by configuring the high 16 bits of the registers utx_config and urx_config. When the number of transferred bytes reaches this value, the corresponding TX/RX end of transfer interrupt will be triggered. While this interrupt is generated, TX stops working. To continue to use TX, you must re-initialize this module. Then, RX resumes to work. If the preset transfer length of TX is less than the data volume actually sent by TX, the other side can only receive the data equal to the transfer length, and the remaining data will be stored in TX FIFO. After this module is re-initialized, the data in TX FIFO can be sent out.

For TX, if the data continuously filled into the TX FIFO is greater than the set transmission length value, only the data of the set length value will be transmitted on the TX pin, and the excess data will be kept in the TX FIFO, and the TX will be re-enabled. After the function, the remaining data in the TX FIFO will be sent out.

For example: set the TX transmission length value to 64, enable the TX function, first fill 63 bytes into the TX FIFO, these 63 bytes will be transmitted on the pins, but no TX transmission completion interrupt is generated, and then the TX FIFO is sent to the TX FIFO. Fill in 1 byte, at this time, the transmission length reaches the transmission length value set by TX, a transmission completion interrupt will be generated, and the TX function will stop. Continue to fill 1 byte into the TX FIFO, you will find that there is no data transmission on the pin, the byte is still retained in the TX FIFO, and the TX function is turned off and re-enabled, and the byte is sent out on the pin.

For RX, if the data length sent by the other party exceeds the set transmission length, RX can continue to receive data after the RX transmission completion interrupt is generated.

For example: set the RX transmission length value to 16, the other party sends 32 bytes of data, RX will generate an RX transmission completion interrupt when it receives 16 bytes of data, and continue to receive the remaining 16 bytes of data, all saved in the RX FIFO.

10.3.14 TX/RX FIFO request interrupt

A TX FIFO request interrupt will be generated when tx_fifo_cnt in uart_fifo_config_1 is greater than tx_fifo_th. When the condition is not met, the interrupt flag will be cleared automatically.

A RX FIFO request interrupt will be generated when rx_fifo_cnt in uart_fifo_config_1 is greater than rx_fifo_th. When the condition is not met, the interrupt flag will be cleared automatically.

TX/RX supports multiple rounds of transmission/receiving, instead of reaching the value set by tx_fifo_th/rx_fifo_th at a time.

E.g:

1. Set tx_fifo_th/rx_fifo_th in register uart_fifo_config_1 to 16.
2. Set cr_utx_frm_en in register utx_config to enable free run mode.
3. Set cr_utx_frdy_mask/cr_urx_frdy_mask in register uart_int_mask to 0, and enable FIFO interrupt of TX/RX.
4. Set cr_utx_en/cr_urx_en in register utx_config/urx_config to enable TX/RX.
5. TX FIFO interrupt: TX will always enter the FIFO interrupt, when the chip sends 128 bytes, set cr_utx_frdy_mask to 1 to shield the interrupt. If you want to enter the TX FIFO interrupt again, set cr_utx_frdy_mask to 0.
6. RX FIFO interrupt: the other party first sends 15 bytes, no interrupt is generated, at this time, the value of rx_fifo_cnt is 15, and an interrupt is generated when 1 byte is sent again to reach the value set by rx_fifo_th. After the transmission is interrupted, the other party sends the data again, and the chip can receive the data.

10.3.15 RX timeout interrupt

The RX timeout interrupt generation condition is: after receiving data last time, the receiver will start timing, and the interrupt will be triggered when the timing value exceeds the timeout threshold and the next data has not been received. The time-out threshold value is in the unit of communication bit.

When the other party sends data to the chip, a timeout interrupt will be generated after the set timeout period is reached.

10.3.16 RX parity check error interrupt

The RX parity check error interrupt will be generated when a parity check error occurs. But it does not affect the RX, which still can correctly receive and analyze the data sent by the other side. When receiving data, RX takes the first 8 bits as data bits and ignores parity check bits, ensuring data consistency.

For example, you can enable parity check by setting cr_utx_prt_en/cr_urx_prt_en in the register utx_config/urx_config, and select the parity check type by setting cr_utx_prt_sel/cr_urx_prt_sel in the register utx_config/urx_config. When the other side sends data to the chip through odd/even parity check, the parity check of RX is disabled, but RX can receive correct data.

10.3.17 TX/RX FIFO overflow interrupt

If the TX/RX FIFO overflows or underflows, it will trigger the corresponding overflow interrupt. When the tx_fifo_clr and rx_fifo_clr bits in the FIFO clear bit register UART_FIFO_CONFIG_0 are set to 1, the corresponding FIFO will be cleared and the overflow interrupt flag will be cleared automatically. You can query the interrupt status through the register UART_INT_STS, and clear the interrupt by writing 1 to the corresponding bit in the register UART_INT_CLEAR.

10.3.18 RX BCR interrupt

A BCR interrupt will be generated when the data received by RX reaches the value set by cr_urx_bcr_value in the register urx_bcr_int_cfg.

The difference from RX END interrupt is that END interrupt is suitable for receiving data of known length, while BCR interrupt can be used to receive interrupts of unknown length. The trigger position of the END interrupt is controlled by cr_urx_len, and the counter will be cleared to 0 when an interrupt is triggered. The trigger position of BCR interrupt is controlled by cr_urx_bcr_value. When the interrupt is triggered, the counter will accumulate instead of being cleared to 0, but it can be cleared by software (cr_urx_bcr_clr). When the BCR interrupt is used together with the chained DMA, if you do not know how much data has been transferred by DMA, you can check it through “count”.

10.3.19 LIN synchronization error interrupt

When cr_utx_lin_en in utx_config is enabled, the LIN mode is enabled. Then, if the synchronization field of the LIN bus is not detected when data is received in this mode, the LIN synchronization error interrupt will be generated.

10.3.20 Auto baud rate detection (universal/fixed characters mode) interrupt

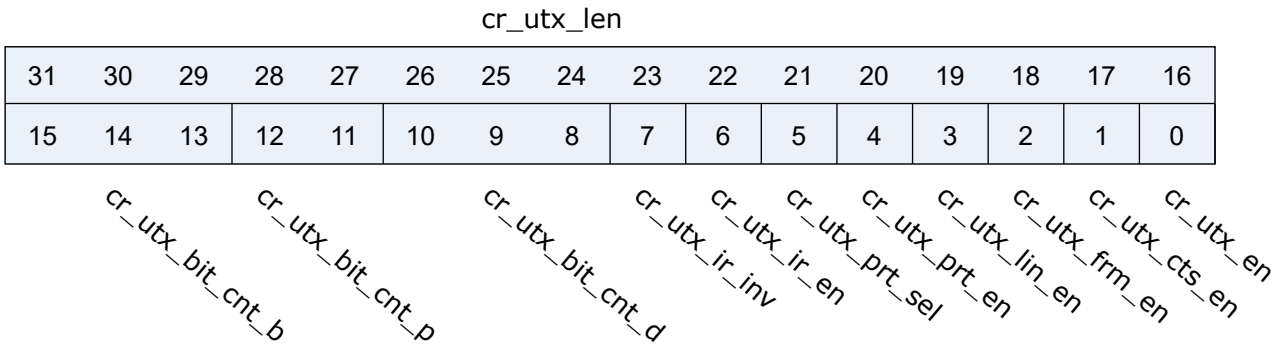
In the auto baud rate detection mode, when a baud rate is detected, the auto baud rate detection (universal/fixed characters mode) interrupt will be generated as configured.

10.4 Register description

Name	Description
utx_config	TX configure
urx_config	RX configure
uart_bit_prd	Baud rate setting
data_config	LSB/MSB-first select
utx_ir_position	IR mode TX setting
urx_ir_position	IR mode RX setting
urx_rto_timer	Time-out value setting
uart_sw_mode	software mode
uart_int_sts	UART interrupt status
uart_int_mask	UART interrupt mask
uart_int_clear	UART interrupt clear
uart_int_en	UART interrupt enable
uart_status	Bus busy status
sts_urx_abr_prd	Aute baud rate detection
urx_abr_prd_b01	ABR detection bit 0/1
urx_abr_prd_b23	ABR detection bit 2/3
urx_abr_prd_b45	ABR detection bit 4/5
urx_abr_prd_b67	ABR detection bit 6/7
urx_abr_pw_tol	0x55 ABR max allowable error
urx_bcr_int_cfg	BCR interrupt counter
utx_rs485_cfg	RS-485 configure
uart_fifo_config_0	FIFO status and DMA mode
uart_fifo_config_1	FIFO threshold and available count
uart_fifo_wdata	TX FIFO
uart_fifo_rdata	RX FIFO

10.4.1 utx_config

Address：0x40010000

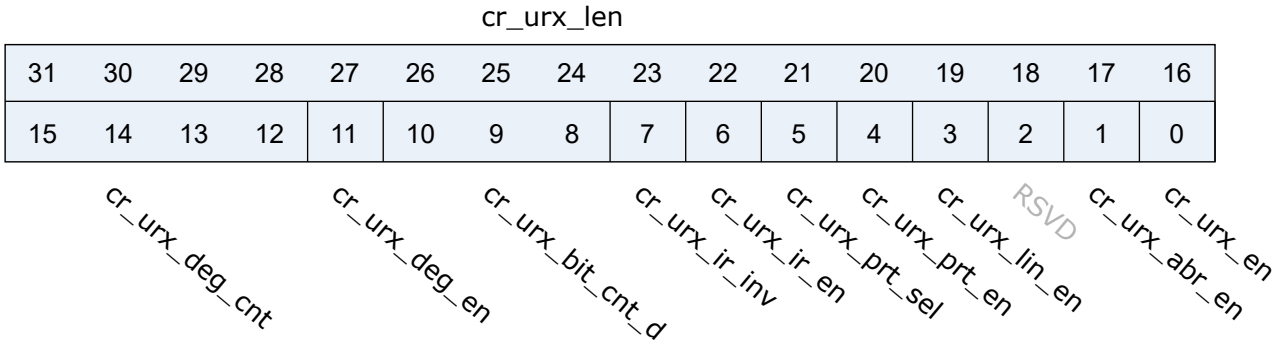


Bits	Name	Type	Reset	Description
31:16	cr_utx_len	r/w	16'd0	Length of UART TX data transfer (Unit: character/byte), TX end interrupt will be triggered when TX send cr_utx_len+1 bytes data (Don't-care if cr_utx_frm_en is enabled)
15:13	cr_utx_bit_cnt_b	r/w	3'd4	UART TX BREAK bit count (for LIN protocol) Note: Additional 8 bit times will be added since LIN Break field requires at least 13 bit times 4: 4+1+8=13 bit 5: 5+1+8=14 bit •••
12:11	cr_utx_bit_cnt_p	r/w	2'd1	UART TX STOP bit count (unit: 0.5 bit) 0: 0.5 bit 1: 1 bit 2: 1.5 bit 3: 2 bit
10:8	cr_utx_bit_cnt_d	r/w	3'd7	UART TX DATA bit count for each character 4: 5 bit 5: 6 bit 6: 7 bit 7: 8 bit
7	cr_utx_ir_inv	r/w	1'b0	Inverse signal of UART TX output in IR mode
6	cr_utx_ir_en	r/w	1'b0	Enable signal of UART TX IR mode
5	cr_utx_prt_sel	r/w	1'b0	Select signal of UART TX parity bit 1: Odd parity 0: Even parity
4	cr_utx_prt_en	r/w	1'b0	Enable signal of UART TX parity bit
3	cr_utx_lin_en	r/w	1'b0	Enable signal of UART TX LIN mode (LIN header will be sent before sending data)

Bits	Name	Type	Reset	Description
2	cr_utx_frm_en	r/w	1'b0	Enable signal of UART TX freerun mode (utx_end_int will be disabled)
1	cr_utx_cts_en	r/w	1'b0	Enable signal of UART TX CTS flow control function
0	cr_utx_en	r/w	1'b0	Enable signal of UART TX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

10.4.2 urx_config

Address：0x40010004



Bits	Name	Type	Reset	Description
31:16	cr_urx_len	r/w	16'd0	Length of UART RX data transfer (Unit: character/byte) urx_end_int will assert when this length is reached
15:12	cr_urx_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_urx_deg_en	r/w	1'b0	Enable signal of RXD input de-glitch function
10:8	cr_urx_bit_cnt_d	r/w	3'd7	UART RX DATA bit count for each character, like cr_utx_-bit_cnt_d
7	cr_urx_ir_inv	r/w	1'b0	Inverse signal of UART RX input in IR mode
6	cr_urx_ir_en	r/w	1'b0	Enable signal of UART RX IR mode
5	cr_urx_prt_sel	r/w	1'b0	Select signal of UART RX parity bit 1: Odd parity 0: Even parity
4	cr_urx_prt_en	r/w	1'b0	Enable signal of UART RX parity bit
3	cr_urx_lin_en	r/w	1'b0	Enable signal of UART RX LIN mode (LIN header will be required and checked before receiving data)
2	RSVD			

Bits	Name	Type	Reset	Description
1	cr_urx_abr_en	r/w	1'b0	Enable signal of UART RX Auto Baud Rate detection function
0	cr_urx_en	r/w	1'b0	Enable signal of UART RX function

10.4.3 uart_bit_prd

Address：0x40010008

cr_urx_bit_prd

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_utx_bit_prd

Bits	Name	Type	Reset	Description
31:16	cr_urx_bit_prd	r/w	16'd255	Period of each UART RX bit, RX baud rate = uart clock / (cr_urx_bit_prd + 1)
15:0	cr_utx_bit_prd	r/w	16'd255	Period of each UART TX bit, TX baud rate = uart clock / (cr_utx_bit_prd + 1)

10.4.4 data_config

Address：0x4001000c

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD cr_uart_bit_inv

Bits	Name	Type	Reset	Description
31:1	RSVD			

Bits	Name	Type	Reset	Description
0	cr_uart_bit_inv	r/w	1'b0	Bit-inverse signal for each data byte 0: Each byte is sent out LSB-first 1: Each byte is sent out MSB-first

10.4.5 utx_ir_position

Address：0x40010010

cr_utx_ir_pos_p

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_utx_ir_pos_s

Bits	Name	Type	Reset	Description
31:16	cr_utx_ir_pos_p	r/w	16'd159	STOP position of UART TX IR pulse
15:0	cr_utx_ir_pos_s	r/w	16'd112	START position of UART TX IR pulse

10.4.6 urx_ir_position

Address：0x40010014

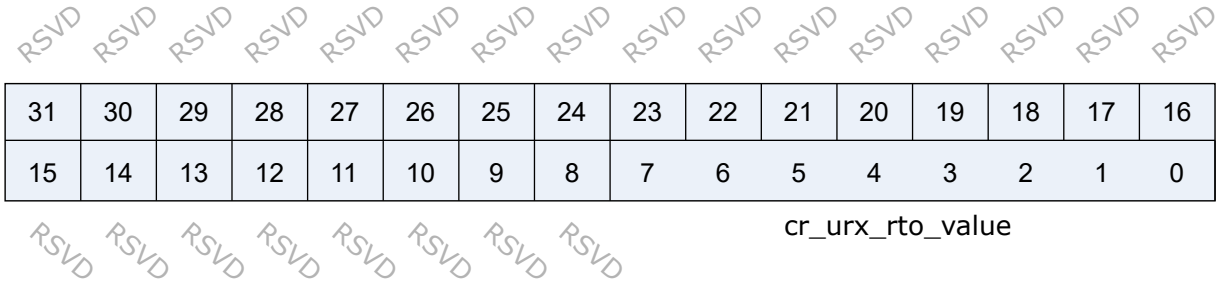
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_urx_ir_pos_s

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	cr_urx_ir_pos_s	r/w	16'd111	START position of UART RXD pulse recovered from IR signal

10.4.7 urx_rto_timer

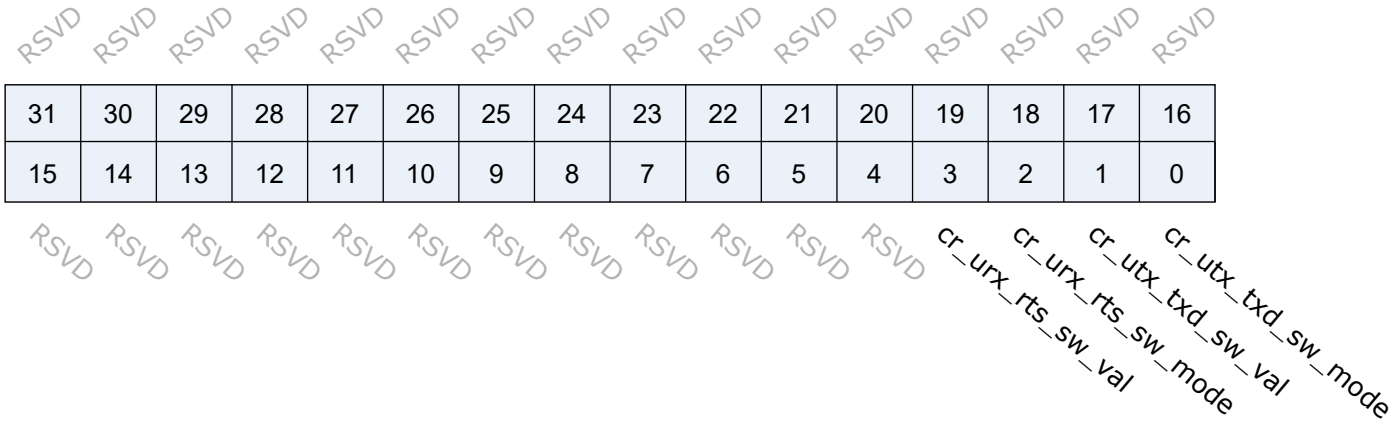
Address：0x40010018



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_urx_rto_value	r/w	8'd15	Time-out value for triggering RTO interrupt (unit: bit time)

10.4.8 uart_sw_mode

Address：0x4001001c

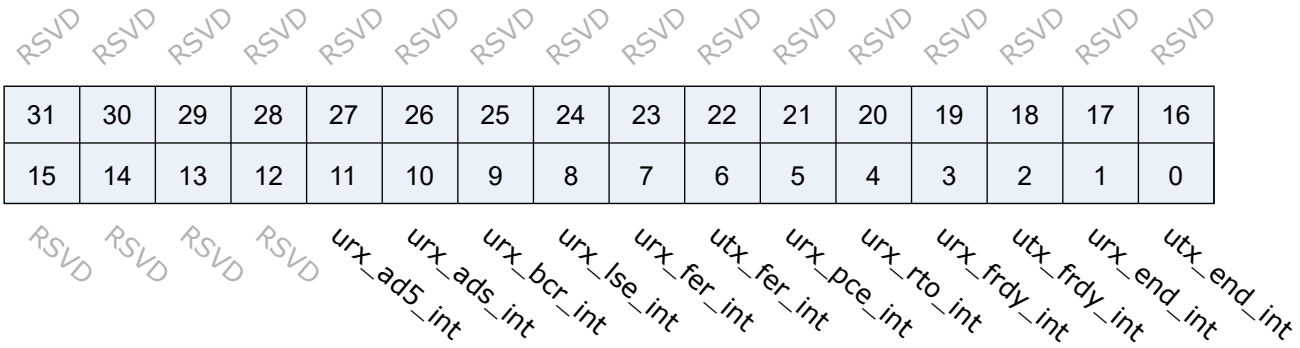


Bits	Name	Type	Reset	Description
31:4	RSVD			
3	cr_urx_rts_sw_val	r/w	1'b0	UART RX RTS output SW control value 0: Low level 1: High level
2	cr_urx_rts_sw_mode	r/w	1'b0	UART RX RTS output SW control mode enable
1	cr_utx_txd_sw_val	r/w	1'b0	UART TX TXD output SW control value 0: Low level 1: High level

Bits	Name	Type	Reset	Description
0	cr_utx_txd_sw_mode	r/w	1'b0	UART TX TXD output SW control mode enable

10.4.9 uart_int_sts

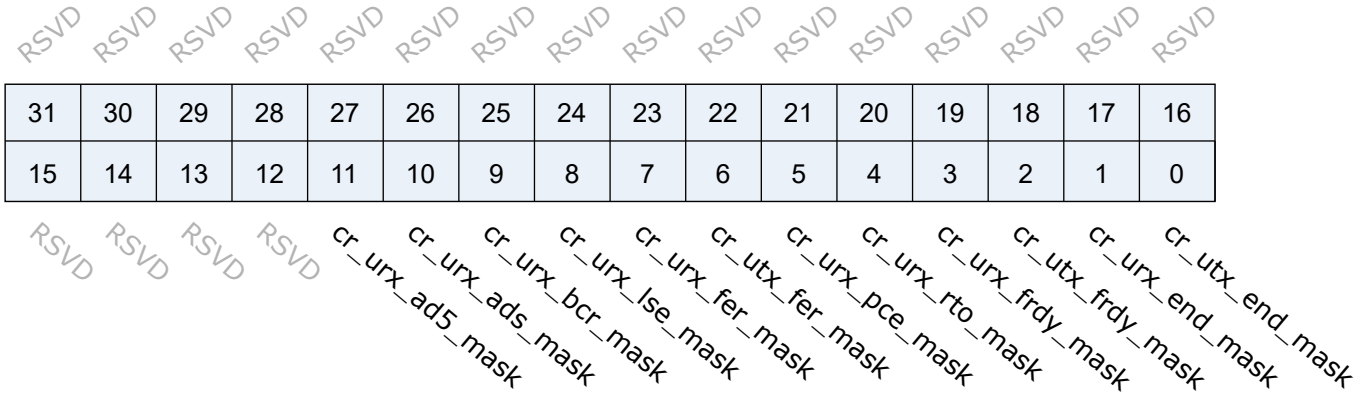
Address：0x40010020



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	urx_ad5_int	r	1'b0	UART RX ABR Detection finish interrupt using codeword 0x55
10	urx_ads_int	r	1'b0	UART RX ABR Detection finish interrupt using START bit
9	urx_bcr_int	r	1'b0	UART RX byte count reached interrupt
8	urx_lse_int	r	1'b0	UART RX LIN mode sync field error interrupt
7	urx_fer_int	r	1'b0	UART RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
6	utx_fer_int	r	1'b0	UART TX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
5	urx_pce_int	r	1'b0	UART RX parity check error interrupt
4	urx_rto_int	r	1'b0	UART RX Time-out interrupt
3	urx_frdy_int	r	1'b0	UART RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
2	utx_frdy_int	r	1'b1	UART TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
1	urx_end_int	r	1'b0	UART RX transfer end interrupt (set according to cr_urx_-len)
0	utx_end_int	r	1'b0	UART TX transfer end interrupt (set according to cr_utx_-len)

10.4.10 uart_int_mask

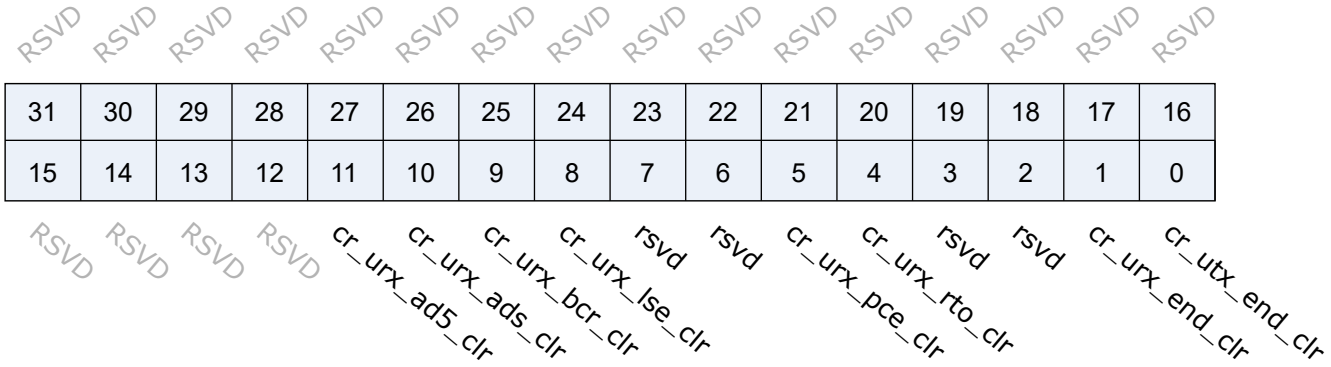
Address：0x40010024



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_mask	r/w	1'b1	Interrupt mask of urx_ad5_int
10	cr_urx_ads_mask	r/w	1'b1	Interrupt mask of urx_ads_int
9	cr_urx_bcr_mask	r/w	1'b1	Interrupt mask of urx_bcr_int
8	cr_urx_lse_mask	r/w	1'b1	Interrupt mask of urx_lse_int
7	cr_urx_fer_mask	r/w	1'b1	Interrupt mask of urx_fer_int
6	cr_utx_fer_mask	r/w	1'b1	Interrupt mask of utx_fer_int
5	cr_urx_pce_mask	r/w	1'b1	Interrupt mask of urx_pce_int
4	cr_urx_rto_mask	r/w	1'b1	Interrupt mask of urx_rto_int
3	cr_urx_frdy_mask	r/w	1'b1	Interrupt mask of urx_frdy_int
2	cr_utx_frdy_mask	r/w	1'b1	Interrupt mask of utx_frdy_int
1	cr_urx_end_mask	r/w	1'b1	Interrupt mask of urx_end_int
0	cr_utx_end_mask	r/w	1'b1	Interrupt mask of utx_end_int

10.4.11 uart_int_clear

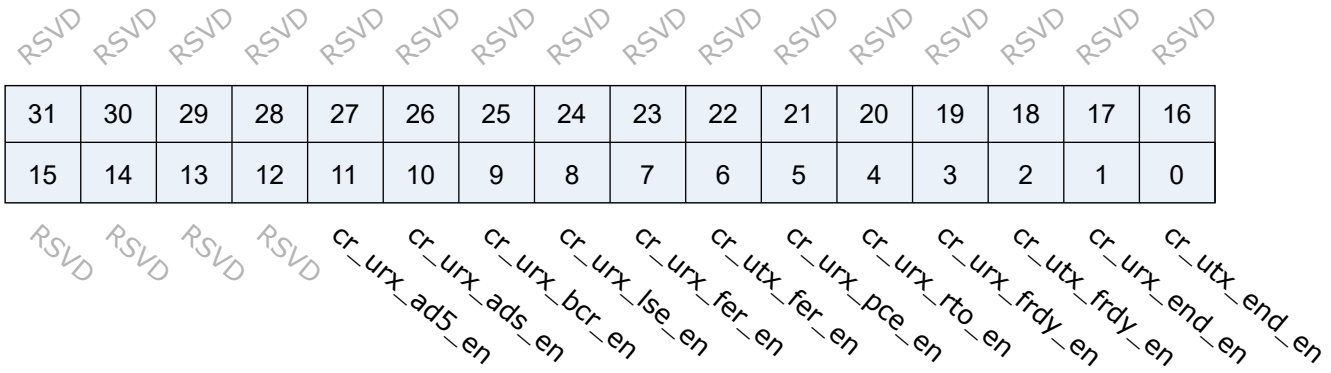
Address：0x40010028



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_clr	w1c	1'b0	Interrupt clear of urx_ad5_int
10	cr_urx_ads_clr	w1c	1'b0	Interrupt clear of urx_ads_int
9	cr_urx_bcr_clr	w1c	1'b0	Interrupt clear of urx_bcr_int
8	cr_urx_lse_clr	w1c	1'b0	Interrupt clear of urx_lse_int
7	rsvd	rsvd	1'b0	
6	rsvd	rsvd	1'b0	
5	cr_urx_pce_clr	w1c	1'b0	Interrupt clear of urx_pce_int
4	cr_urx_rto_clr	w1c	1'b0	Interrupt clear of urx_rto_int
3	rsvd	rsvd	1'b0	
2	rsvd	rsvd	1'b0	
1	cr_urx_end_clr	w1c	1'b0	Interrupt clear of urx_end_int
0	cr_utx_end_clr	w1c	1'b0	Interrupt clear of utx_end_int

10.4.12 uart_int_en

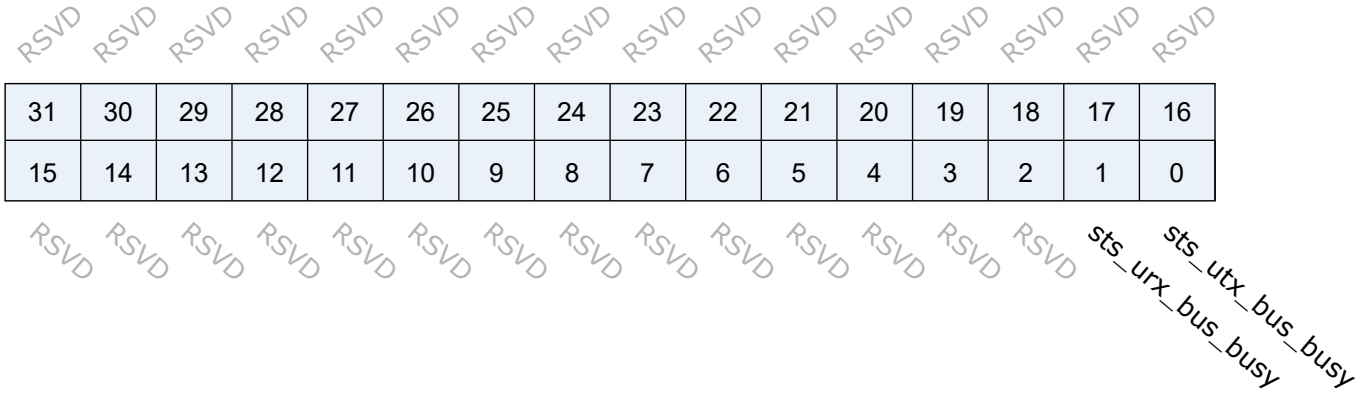
Address：0x4001002c



Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_en	r/w	1'b1	Interrupt enable of urx_ad5_int
10	cr_urx_ads_en	r/w	1'b1	Interrupt enable of urx_ads_int
9	cr_urx_bcr_en	r/w	1'b1	Interrupt enable of urx_bcr_int
8	cr_urx_lse_en	r/w	1'b1	Interrupt enable of urx_lse_int
7	cr_urx_fer_en	r/w	1'b1	Interrupt enable of urx_fer_int
6	cr_utx_fer_en	r/w	1'b1	Interrupt enable of utx_fer_int
5	cr_urx_pce_en	r/w	1'b1	Interrupt enable of urx_pce_int
4	cr_urx_rto_en	r/w	1'b1	Interrupt enable of urx_rto_int
3	cr_urx_frdy_en	r/w	1'b1	Interrupt enable of urx_frdy_int
2	cr_utx_frdy_en	r/w	1'b1	Interrupt enable of utx_frdy_int
1	cr_urx_end_en	r/w	1'b1	Interrupt enable of urx_end_int
0	cr_utx_end_en	r/w	1'b1	Interrupt enable of utx_end_int

10.4.13 uart_status

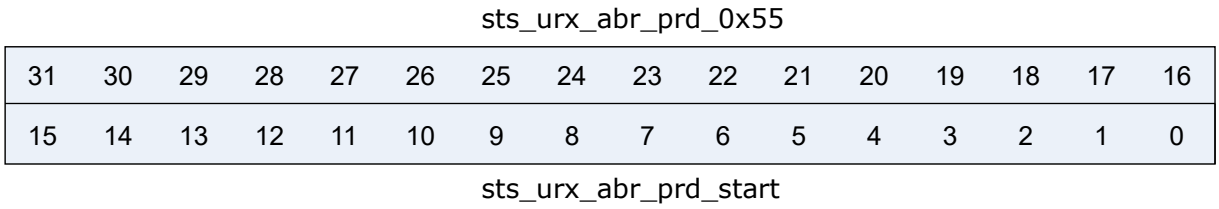
Address：0x40010030



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	sts_urx_bus_busy	r	1'b0	Indicator of UART RX bus busy 0: Idle 1: Busy
0	sts_utx_bus_busy	r	1'b0	Indicator of UART TX bus busy 0: Idle 1: Busy

10.4.14 sts_urx_abr_prd

Address：0x40010034



Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_0x55	r	16'd0	Bit period of Auto Baud Rate detection using codeword 0x55, baud rate = uart clock / (sts_urx_abr_prd_0x55 + 1)
15:0	sts_urx_abr_prd_start	r	16'd0	Bit period of Auto Baud Rate detection using START bit, baud rate = uart clock / (sts_urx_abr_prd_start + 1)

10.4.15 urx_abr_prd_b01

Address：0x40010038

sts_urx_abr_prd_bit1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_urx_abr_prd_bit0

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit1	r	16'd0	Bit period of Auto Baud Rate detection - bit[1]
15:0	sts_urx_abr_prd_bit0	r	16'd0	Bit period of Auto Baud Rate detection - bit[0]

10.4.16 urx_abr_prd_b23

Address：0x4001003c

sts_urx_abr_prd_bit3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_urx_abr_prd_bit2

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit3	r	16'd0	Bit period of Auto Baud Rate detection - bit[3]
15:0	sts_urx_abr_prd_bit2	r	16'd0	Bit period of Auto Baud Rate detection - bit[2]

10.4.17 urx_abr_prd_b45

Address：0x40010040

sts_urx_abr_prd_bit5

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_urx_abr_prd_bit4

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit5	r	16'd0	Bit period of Auto Baud Rate detection - bit[5]
15:0	sts_urx_abr_prd_bit4	r	16'd0	Bit period of Auto Baud Rate detection - bit[4]

10.4.18 urx_abr_prd_b67

Address：0x40010044

sts_urx_abr_prd_bit7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_urx_abr_prd_bit6

Bits	Name	Type	Reset	Description
31:16	sts_urx_abr_prd_bit7	r	16'd0	Bit period of Auto Baud Rate detection - bit[7]
15:0	sts_urx_abr_prd_bit6	r	16'd0	Bit period of Auto Baud Rate detection - bit[6]

10.4.19 urx_abr_pw_tol

Address：0x40010048

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD cr_urx_abr_pw_tol

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_urx_abr_pw_tol	r/w	8'd3	Auto Baud Rate detection pulse-width tolerance for using codeword 0x55

10.4.20 urx_bcr_int_cfg

Address：0x40010050

sts_urx_bcr_count

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_urx_bcr_value

Bits	Name	Type	Reset	Description
31:16	sts_urx_bcr_count	r	16'd0	Current byte count of urx_bcr_int counter, auto-cleared bt cr_urx_bcr_clr
15:0	cr_urx_bcr_value	r/w	16'hFFFF	Byte count setting for urx_bcr_int counter

10.4.21 utx_rs485_cfg

Address：0x40010054

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

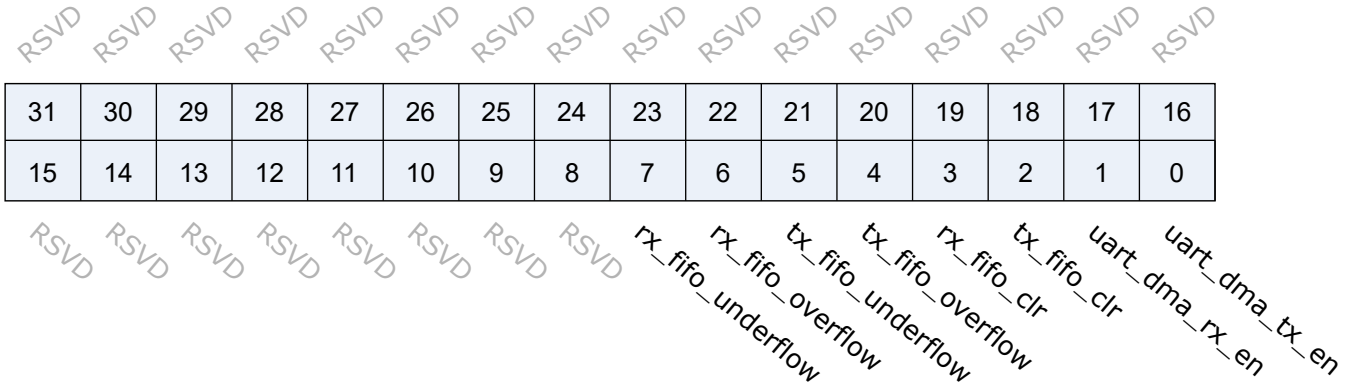
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD cr_utx_rs485_pol cr_utx_rs485_en

Bits	Name	Type	Reset	Description
31:2	RSVD			
1	cr_utx_rs485_pol	r/w	1'b1	DE pin polarity in RS-485 transceiver mode 1'b0: DE is active-low 1'b1: DE is active-high
0	cr_utx_rs485_en	r/w	1'b0	Enable signal for RS-485 transceiver mode 1'b0: Disabled, normal UART 1'b1: Enabled, IO is connected to RS-485 transceiver and RTS_O becomes DE function

10.4.22 uart_fifo_config_0

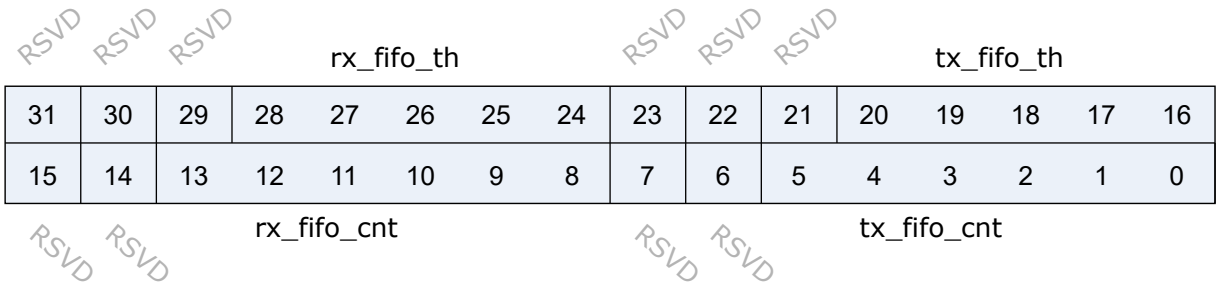
Address：0x40010080



Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO, RX FIFO will be empty when write 1 to this bit
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO, TX FIFO will be empty when write 1 to this bit
1	uart_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	uart_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

10.4.23 uart_fifo_config_1

Address：0x40010084



Bits	Name	Type	Reset	Description
31:29	RSVD			
28:24	rx_fifo_th	r/w	5'd0	RX FIFO threshold, dma_rx_req and rx_fifo_int will not be asserted if rx_fifo_cnt is less than this value
23:21	RSVD			
20:16	tx_fifo_th	r/w	5'd0	TX FIFO threshold, dma_tx_req and tx_fifo_int will not be asserted if tx_fifo_cnt is less than this value
15:14	RSVD			
13:8	rx_fifo_cnt	r	6'd0	RX FIFO available count, means byte count of data received in RX FIFO
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count, means empty space remained in TX FIFO

10.4.24 uart_fifo_wdata

Address：0x40010088

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

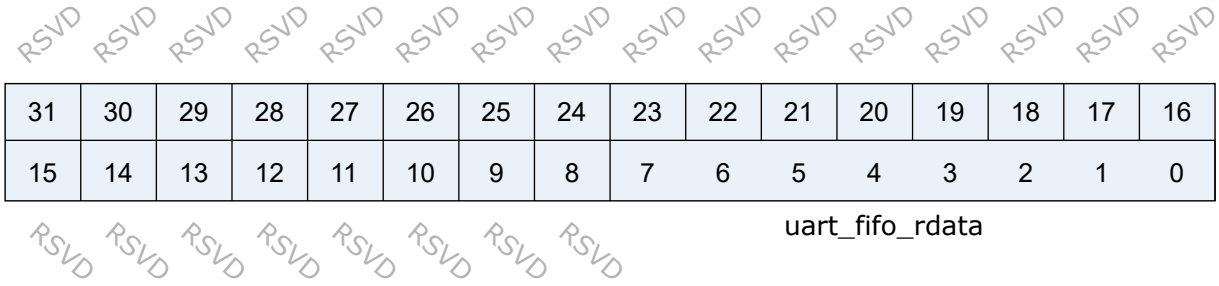
RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

uart_fifo_wdata

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	uart_fifo_wdata	w	x	TX FIFO, size is 32*1=32-byte

10.4.25 uart_fifo_rdata

Address：0x4001008c



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	uart_fifo_rdata	r	8'h0	RX FIFO, size is 32*1=32-byte

11 I2C

11.1 Overview

Inter-Integrated Circuit (I2C) is a serial communication bus, which uses a multi-slave and multi-master architecture and is connected to low-speed peripherals. Each device has a unique address identifier and can be used as a transmitter or receiver. The address of each device connected to the bus can be set by software through a unique address and the existing master or slave relation. The master can work as a master transmitter or a master receiver. If two or more masters are initialized at the same time, data can be prevented from being damaged through conflict detection and arbitration during transmission.

BL616/BL618 has two I2C controller masters, whose slaveAddr, subAddr, and data to be transferred can be flexibly configured, to facilitate communication with slaves. With the FIFO of 2-word depth and interrupt function, it can be used with DMA to improve efficiency and supports flexible adjustment of clock frequency.

11.2 Features

- Master mode
- Multi-master mode and arbitration function
- Flexible control of the level duration of the start, end and data transmission phases in segments
- Supports 7-bit address mode and 10-bit address mode
- Supports DMA transfer mode
- Supports multiple interrupt mechanisms

11.3 Functional Description

Table 11.1: I2C pin list

Name	Type	Description
I2Cx_SCL	Input/output	I2C serial clock signal
I2Cx_SDA	Input/output	I2C serial data signal

11.3.1 Start and stop conditions

All transmissions start with a START condition and end with a STOP condition. The START and STOP conditions are generally generated by the master. The bus is considered to be busy after the START condition and to be idle for a certain period of time after the STOP condition.

START condition: SDA produces a high-to-low level transition when SCL is high;

STOP condition: SDA produces a low-to-high level transition when SCL is high.

Waveform diagram:

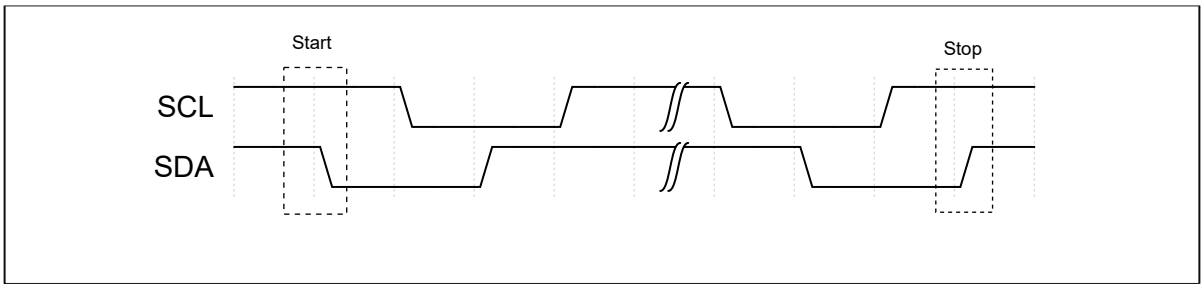


Fig. 11.1: Start and stop conditions of I2C

11.3.2 Data Transfer Format

1. 7-bit address mode:

The first 8 bits transferred are addressing bytes, including a 7-bit slave address and a 1-bit direction bit. Sending or receiving data by the master is controlled by the 8th bit in the first byte sent by the master. If it is 0, it means that the data is sent by the master, while “1” indicates that data is received by the master. After the direction bit is the answer bit (ACK), which is sent by the slave to answer (pull the signal low) and the host starts transmitting the specified length of data after receiving the answer. Upon data transfer completed, the master sends out a STOP signal, with waveform shown below:

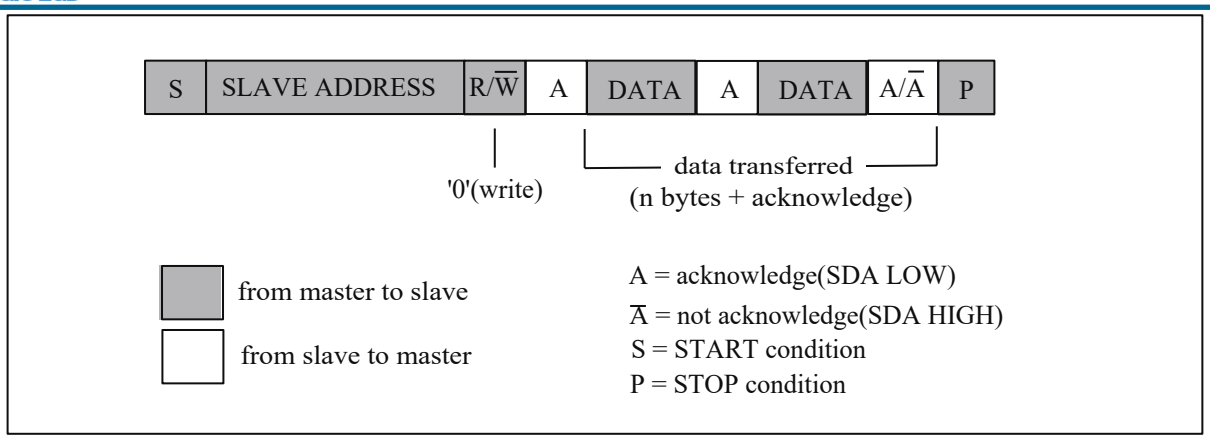


Fig. 11.2: Master transmit and slave receive data formats

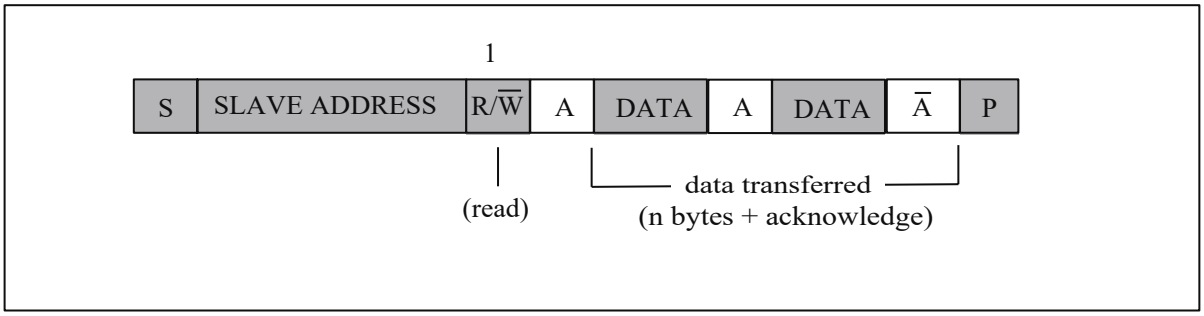


Fig. 11.3: Master receive and slave transmit data formats

Master Transmit and Slave Receive Timing

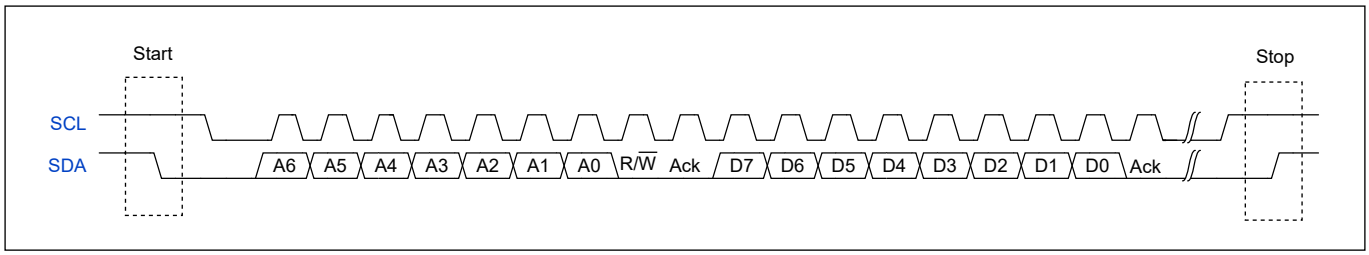


Fig. 11.4: Timing of master transmitter and slave receiver

Master Receive and Slave Transmit Timing

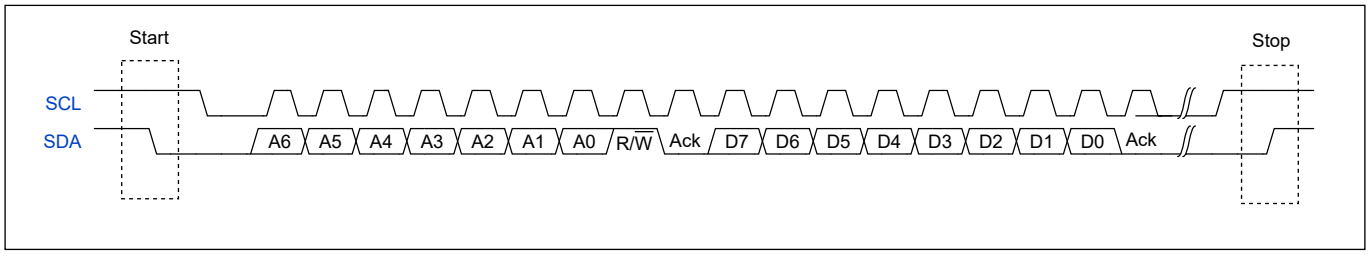


Fig. 11.5: Timing of master receiver and slave transmitter

2. 10-bit address mode

The cr_i2c_10b_addr_en in the register i2c_config must be set to 1 before use.

The 10-bit slave address consists of the two bytes after the START condition (S) or the repeated START condition (Sr). The first 7 bits of the first byte are 1111 0XX, where XX are the first two bits of MSB of the 10-bit address. The 8th bit of the first byte is the read/write bit that determines the transfer direction. The second byte is the remaining low 8 bits of the 10 bit address. The data transfer format is as follows.

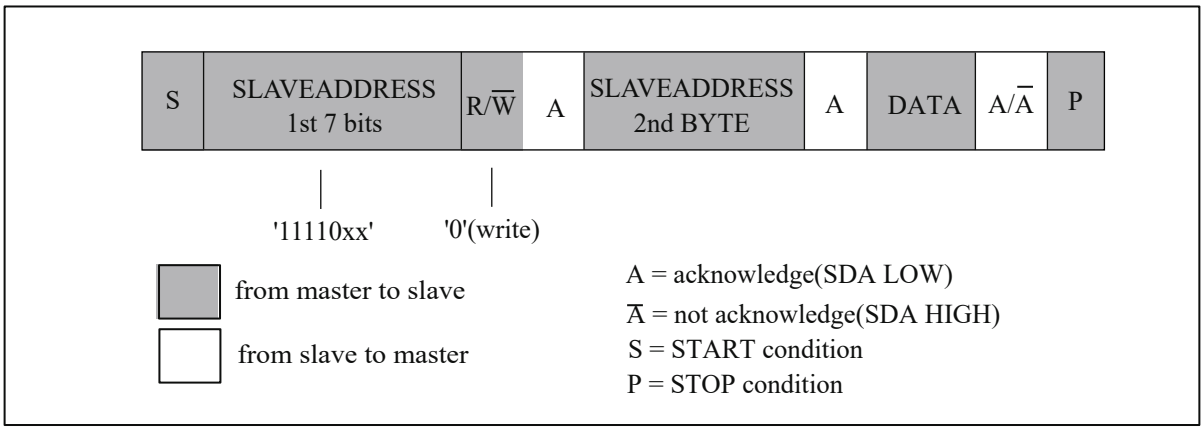


Fig. 11.6: Master transmit and slave receive data format (10bit slave address)

When receiving the 10-bit address following the START condition, the slave compares the first byte (1111 0XX) of the slave address with its own address, and checks whether the eighth bit (read/write bit) is 0. If the value of XX in the first byte is the same as the top two bits of the slave's 10 bit address, the first byte match passes and the slave will give answer A. If there are multiple slave devices connected to the bus, more than one device may match and generate answer A. Next, all slaves start to match the second byte (XXXX XXXX), where only one slave will have the exact same lower eight bits of the 10 bit address as the second byte, and that slave will give answer A. The slave that is addressed by the master will remain addressed until it receives a termination condition or a repeat start condition.

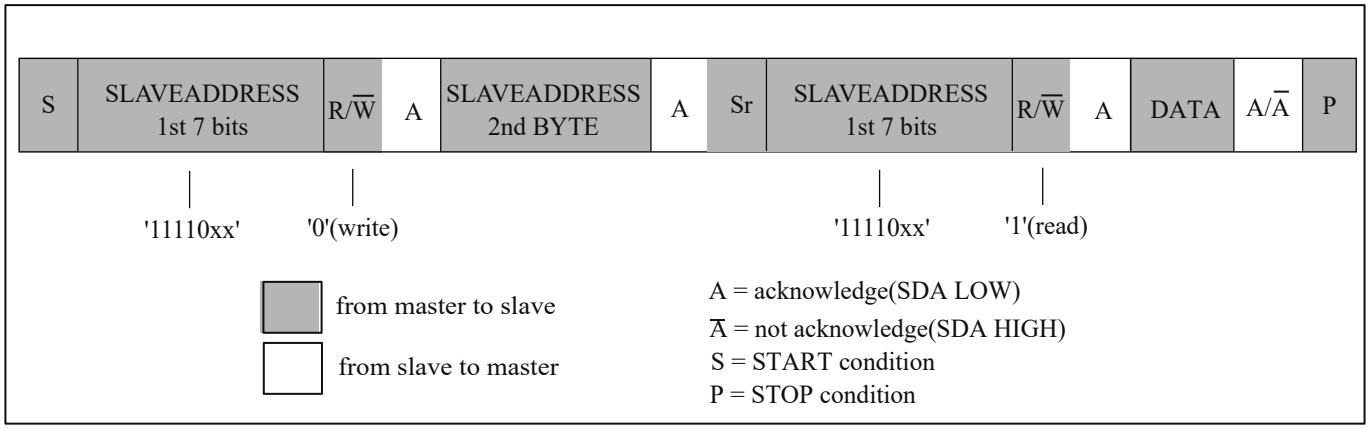


Fig. 11.7: Master receive and slave transmit data format (10bit slave address)

Before the second acknowledgement A, the process is the same as that of the master-transmitter addressing the

slave-receiver. After the repeated START condition (Sr), the matched slave will remain in the addressed state. This slave will check whether the first 7 bits of the first byte after Sr are 1111 0XX, and then test whether the 8th bit is 1 (read). If this also matches, the slave considers that it is addressed as a transmitter and generates an acknowledgement (A). The slave-transmitter will remain in the addressed state until it receives the STOP condition (P) or the repeated START condition (Sr) followed by a different slave address. Then, under Sr, all the slaves will compare their addresses with 11110XX and test the eighth bit (read/write bit). However, they will not be addressed, because for 10-bit devices, the read/write bit is 1, or for 7-bit devices, the slave addresses of 1111 0XX do not match.

11.3.3 Arbitration

When there are multiple masters on I2C bus, it may happen that multiple masters start data transfer at the same time. At this time, the arbitration mechanism will decide which master has the right to transfer data, while other masters have to give up the control of the bus and wait until the bus is idle before transferring data again.

During data transfer, all masters must check whether the SDA is consistent with the data they want to send when SCL stays high. When the SDA level is different from the expected one, it means that other masters are transferring data at the same time. The masters with different SDA levels will lose the arbitration and other masters will complete the data transfer.

The waveform of two masters transferring data and initiating the arbitration mechanism at the same time is as follows:

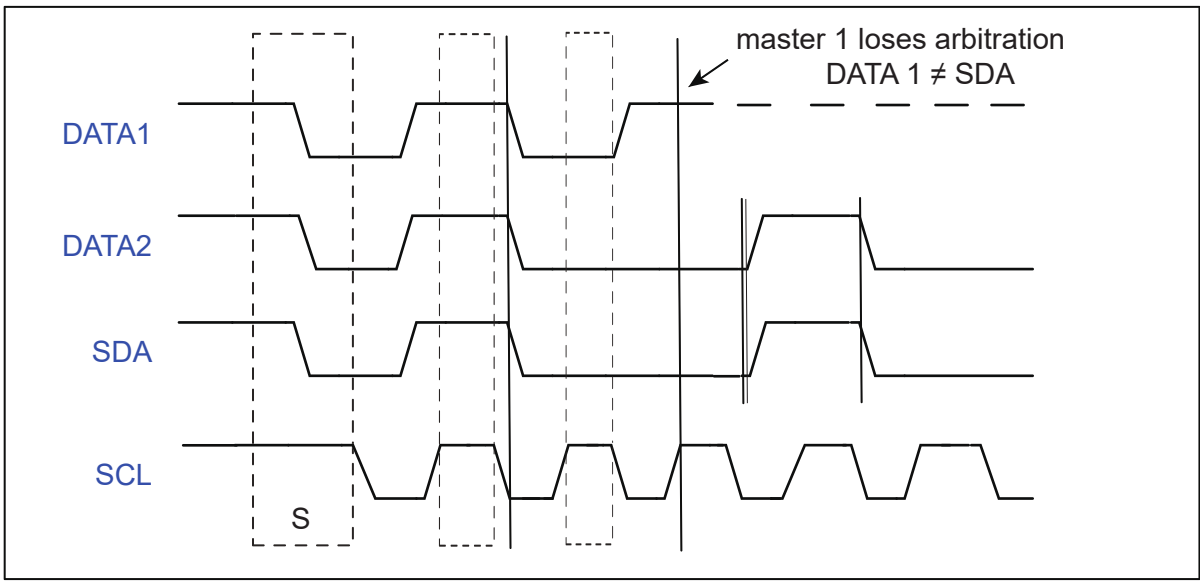


Fig. 11.8: Waveform of simultaneous data transfer

11.4 I2C Clock Setting

I2C clock can be derived from bclk (bus clock) and xclk, and frequency division can be done on this basis. The duration of the start condition, each bit of data and the end condition are set by registers i2c_prd_start, i2c_prd_data and i2c_prd_stop respectively. Each of these durations can be subdivided into 4 phases, and the number of samples in each phase is controlled by a separate byte in the register (the actual value is the register value plus 1). The 4 phase settings in the data section together determine the frequency division factor of the i2c clock. As shown in the figure below, suppose the I2C clock source is selected as 80M bclk and the register i2c_prd_data is set to 0x09070b09, then the second 0 in the figure is 0x09+1=0x0a, the second 1 is 0x07+1=0x08, the second 2 is 0x0b+1=0x0c, and the second 3 is 0x09+1=0x0a. Then the clock frequency of I2C is 80MHz/(0x0a+0x08+0x0c+0x0a) = 2MHz. Similarly, the first 0, 1, 2 and 3 are set by register i2c_prd_start, which determines the duration of the start condition, and the third 0, 1, 2 and 3 are set by register i2c_prd_stop, which determines the duration of the end condition.

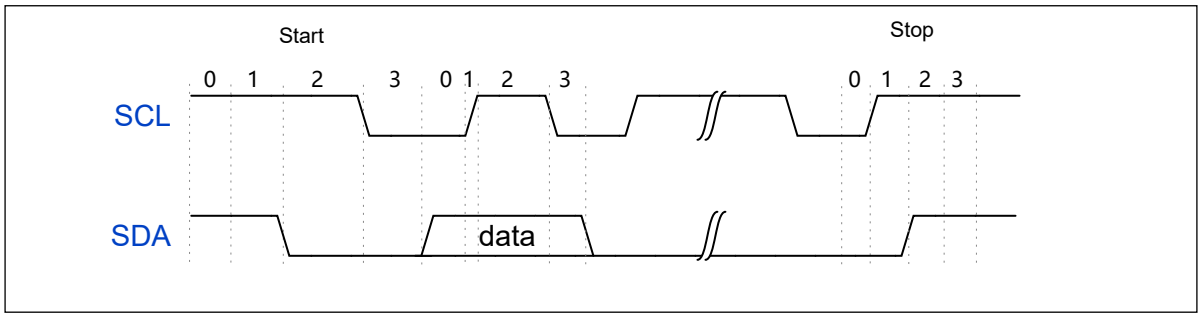


Fig. 11.9: I2C clock setting

11.5 I2C Configuration Flow

11.5.1 Configuration Items

- Read/write flag bit
- Slave address
- Slave register address
- Slave register address length
- Data (TX: configure the sent data; RX: store the received data)
- Data length
- Enable signal

11.5.2 Read/Write Flag Bit

I2C supports TX and RX working statuses. The cr_i2c_pkt_dir in the register i2c_config represents the TX/RX status, “0” for TX status and “1” for RX status.

11.5.3 Slave Address

Each slave connected to I2C will have a unique device address, which is usually 7 bits long. This address will be written into the cr_i2c_slv_addr in the register i2c_config. I2C will automatically shift to the left by 1 bit before sending the address, and the TX/RX direction bit will be added to the LSB.

11.5.4 Slave Register Address

The slave register address represents the register address where I2C needs to read and write a slave register. The slave register address is written to the register i2c_sub_addr, and the cr_i2c_sub_addr_en in the register i2c_config must be set to 1. If cr_i2c_sub_addr_en in the register i2c_config is set to 0, the I2C master will skip the slave register address field when sending.

11.5.5 Slave Register Address Length

The slave register address length is subtracted by 1 and then written to cr_i2c_sub_addr_bc in the register i2c_config.

11.5.6 Data

It refers to the data that needs to be sent to or received from the slave. When sending data, I2C must write the data (in word) into the register i2c_fifo_wdata. When receiving data, I2C must read out the data (in word) from the register i2c_fifo_rdata.

11.5.7 Data Length

The cr_i2c_pkt_len in the i2c_config register sets the send data length (the value written to the register + 1 is the send data length), and the maximum send length is 256 bytes.

11.5.8 Enable Signal

After the above items are configured, when cr_i2c_m_en in the enable signal register i2c_config is set to 1, the I2C sending process will be started automatically.

When the read/write flag bit is configured as 0, I2C sends data. Take sending 2 bytes as an example, the master’s transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1 bit + 0) + ACK
3. Slave register address + ACK

4. 1-byte data + ACK
5. 1-byte data + ACK
6. Stop bit

When the read/write flag bit is configured as 1, I2C receives data. Take receiving 2 bytes as an example, the master's transmission flow is as follows:

1. Start bit
2. (The slave address shifts to the left by 1 bit + 0) + ACK
3. Slave register address + ACK
4. Start bit
5. (The slave address shifts to the left by 1 bit + 1) + ACK
6. 1-byte data + ACK
7. 1-byte data + ACK
8. Stop bit

11.6 FIFO Management

I2C FIFO has a 2-word depth, and I2C includes RX FIFO and TX FIFO. The rx_fifo_cnt in the register i2c_fifo_config_1 represents how much data (in word) in RX FIFO needs to be read. The tx_fifo_cnt in the register i2c_fifo_config_1 represents how much free space (in word) in TX FIFO for writing.

I2C FIFO status:

- RX FIFO underflow: When the data in RX FIFO is completely read out or empty, if I2C continues to read data from RX FIFO, the rx_fifo_underflow in the register i2c_fifo_config_0 will be set to 1;
- RX FIFO overflow: When I2C receives data until the two words of RX FIFO are filled, without reading RX FIFO, if I2C receives data again, the rx_fifo_overflow in the register i2c_fifo_config_0 will be set to 1;
- TX FIFO underflow: When the data size filled into TX FIFO does not meet the configured I2C data length: cr_i2c_pkt_len in i2c_config, and no new data is filled into TX FIFO, the tx_fifo_underflow in the register i2c_fifo_config_0 will be set to 1;
- TX FIFO overflow: After the two words of TX FIFO are filled, before the data in TX FIFO is sent out, if data is filled into TX FIFO again, the tx_fifo_overflow in the register i2c_fifo_config_0 will be set to 1.

11.7 Use with DMA

I2C can send and receive data through DMA. Setting i2c_dma_tx_en in the register i2c_fifo_config_0 to 1 will enable the DMA TX mode. After the channel for I2C is allocated, DMA will transfer data from memory to the i2c_fifo_wdata register. Setting i2c_dma_rx_en in the register i2c_fifo_config_0 to 1 will enable the DMA RX mode. After the channel for I2C is allocated, DMA will transfer the data in the register i2c_fifo_rdata to memory. When I2C is used with DMA, DMA will automatically transfer data, so it is unnecessary for CPU to write data into I2C TX FIFO or read data from I2C RX FIFO.

11.7.1 DMA Sending Flow

1. Set read/write flag bit to 0
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data Length
6. Set enable signal register to 1
7. Configure DMA transfer size
8. Configure the transfer width of DMA source address
9. Configure the transfer width of DMA destination address (when I2C is used with DMA, the transfer width of destination address must be set to 32 bits, which is word-aligned)
10. Configure the DMA source address as the memory address for storing sent data
11. Configure the DMA destination address to I2C TX FIFO address, i2c_fifo_wdata
12. Enable DMA

11.7.2 DMA Receiving Flow

1. Set read/write flag bit to 1
2. Set slave address
3. Set slave register address
4. Set slave register address length
5. Data Length
6. Set enable signal register to 1
7. Configure DMA transfer size

8. Configure the transfer width of DMA source address (when I2C is used with DMA, the transfer width of source address must be set to 32 bits, which is word-aligned)
9. Configure the transfer width of DMA destination address
10. Configure the DMA source address to I2C RX FIFO address, i2c_fifo_rdata
11. Configure the DMA destination address as the memory address for storing received data
12. Enable DMA

11.8 Interrupt

I2C includes the following interrupts:

- I2C_TRANS_END_INT * I2C transfer end interrupt, which is generated when I2C completes a transfer
- I2C_TX_FIFO_READY_INT * When tx_fifo_cnt in i2c_fifo_config_1 is greater than tx_fifo_th, a TX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied
- I2C_RX_FIFO_READY_INT * When rx_fifo_cnt in i2c_fifo_config_1 is greater than rx_fifo_th, an RX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not satisfied
- I2C_NACK_RECV_INT * When the I2C module detects a NACK state, a NACK interrupt is generated
- I2C_ARB_LOST_INT * I2C arbitration lost interrupt
- I2C_FIFO_ERR_INT * FIFO ERROR interrupt is generated when TX/RX FIFO overflows or underflows

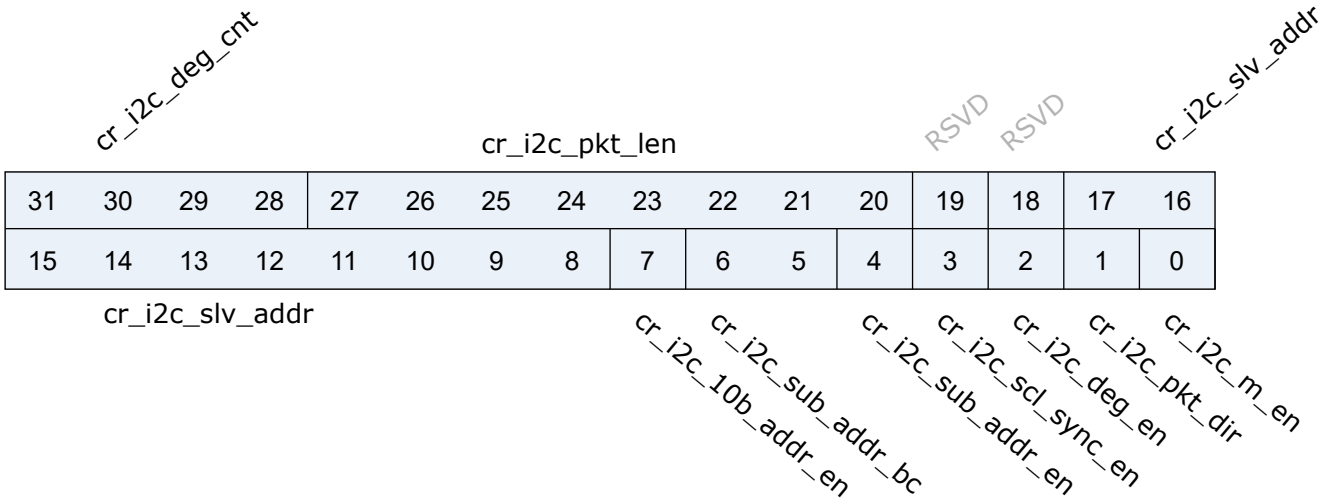
11.9 Register description

Name	Description
i2c_config	Master configure
i2c_int_sts	Interrupt configure and status
i2c_sub_addr	Sub-address setting
i2c_bus_busy	Bus busy status
i2c_prd_start	Start period setting
i2c_prd_stop	Stop period setting
i2c_prd_data	Data period setting
i2c_fifo_config_0	FIFO status and DMA mode
i2c_fifo_config_1	FIFO threshold and available count
i2c_fifo_wdata	TX FIFO

Name	Description
i2c_fifo_rdata	RX FIFO

11.9.1 i2c_config

Address：0x40018000

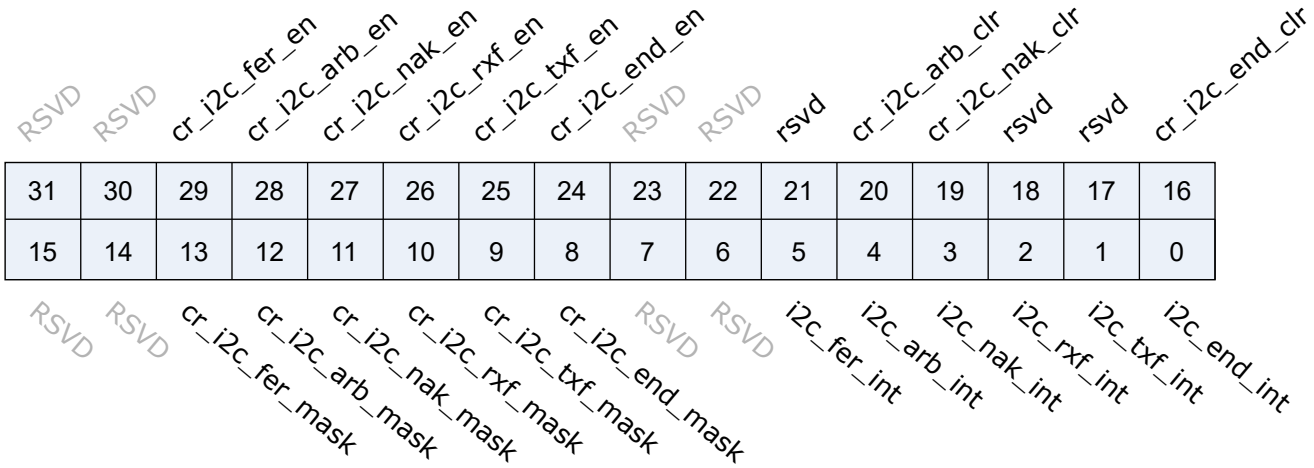


Bits	Name	Type	Reset	Description
31:28	cr_i2c_deg_cnt	r/w	4'd0	De-glitch function cycle count
27:20	cr_i2c_pkt_len	r/w	8'd0	Packet length (unit: byte)
19:18	RSVD			
17:8	cr_i2c_slv_addr	r/w	10'h0	Slave address for I2C transaction (target address)
7	cr_i2c_10b_addr_en	r/w	1'b0	Slave address 10-bit mode enable
6:5	cr_i2c_sub_addr_bc	r/w	2'd0	Sub-address field byte count 2'd0: 1-byte, 2'd1: 2-byte, 2'd2: 3-byte, 2'd3: 4-byte
4	cr_i2c_sub_addr_en	r/w	1'b0	Enable signal of I2C sub-address field
3	cr_i2c_scl_sync_en	r/w	1'b1	Enable signal of I2C SCL synchronization, should be enabled to support Multi-Master and Clock-Stretching (Normally should not be turned-off)
2	cr_i2c_deg_en	r/w	1'b0	Enable signal of I2C input de-glitch function (for all input pins)
1	cr_i2c_pkt_dir	r/w	1'b1	Transfer direction of the packet 1'b0: Write; 1'b1: Read

Bits	Name	Type	Reset	Description
0	cr_i2c_m_en	r/w	1'b0	Enable signal of I2C Master function Asserting this bit will trigger the transaction, and should be de-asserted after finish

11.9.2 i2c_int_sts

Address：0x40018004



Bits	Name	Type	Reset	Description
31:30	RSVD			
29	cr_i2c_fer_en	r/w	1'b1	Interrupt enable of i2c_fer_int
28	cr_i2c_arb_en	r/w	1'b1	Interrupt enable of i2c_arb_int
27	cr_i2c_nak_en	r/w	1'b1	Interrupt enable of i2c_nak_int
26	cr_i2c_rxf_en	r/w	1'b1	Interrupt enable of i2c_rxf_int
25	cr_i2c_txf_en	r/w	1'b1	Interrupt enable of i2c_txf_int
24	cr_i2c_end_en	r/w	1'b1	Interrupt enable of i2c_end_int
23:22	RSVD			
21	rsvd	rsvd	1'b0	
20	cr_i2c_arb_clr	w1c	1'b0	Interrupt clear of i2c_arb_int
19	cr_i2c_nak_clr	w1c	1'b0	Interrupt clear of i2c_nak_int
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_i2c_end_clr	w1c	1'b0	Interrupt clear of i2c_end_int

Bits	Name	Type	Reset	Description
15:14	RSVD			
13	cr_i2c_fer_mask	r/w	1'b1	Interrupt mask of i2c_fer_int
12	cr_i2c_arb_mask	r/w	1'b1	Interrupt mask of i2c_arb_int
11	cr_i2c_nak_mask	r/w	1'b1	Interrupt mask of i2c_nak_int
10	cr_i2c_rxf_mask	r/w	1'b1	Interrupt mask of i2c_rxf_int
9	cr_i2c_txf_mask	r/w	1'b1	Interrupt mask of i2c_txf_int
8	cr_i2c_end_mask	r/w	1'b1	Interrupt mask of i2c_end_int
7:6	RSVD			
5	i2c_fer_int	r	1'b0	I2C TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
4	i2c_arb_int	r	1'b0	I2C arbitration lost interrupt
3	i2c_nak_int	r	1'b0	I2C NACK-received interrupt
2	i2c_rxf_int	r	1'b0	I2C RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
1	i2c_txf_int	r	1'b1	I2C TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	i2c_end_int	r	1'b0	I2C transfer end interrupt

11.9.3 i2c_sub_addr

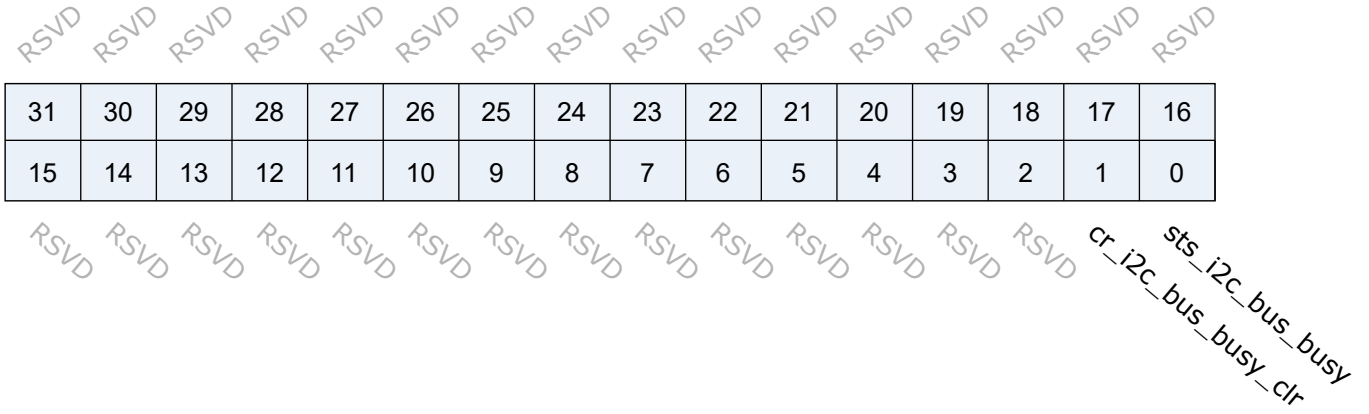
Address：0x40018008

cr_i2c_sub_addr_b3								cr_i2c_sub_addr_b2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_i2c_sub_addr_b1								cr_i2c_sub_addr_b0							

Bits	Name	Type	Reset	Description
31:24	cr_i2c_sub_addr_b3	r/w	8'd0	I2C sub-address field - byte[3]
23:16	cr_i2c_sub_addr_b2	r/w	8'd0	I2C sub-address field - byte[2]
15:8	cr_i2c_sub_addr_b1	r/w	8'd0	I2C sub-address field - byte[1]
7:0	cr_i2c_sub_addr_b0	r/w	8'd0	I2C sub-address field - byte[0] (sub-address starts from this byte)

11.9.4 i2c_bus_busy

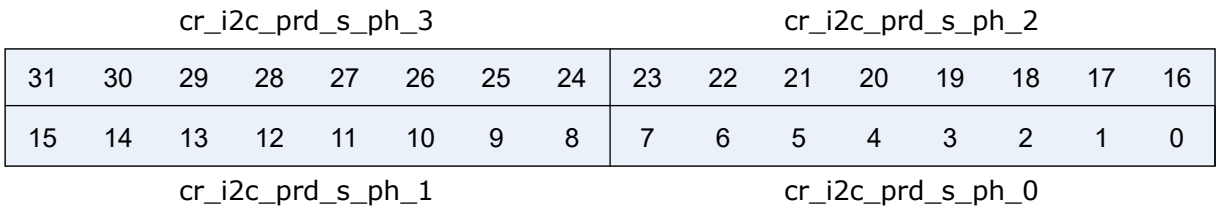
Address：0x4001800c



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	cr_i2c_bus_busy_clr	w1c	1'b0	Clear signal of bus_busy status, not for normal usage (in case I2C bus hangs)
0	sts_i2c_bus_busy	r	1'b0	Indicator of I2C bus busy 0: Idle 1: Busy

11.9.5 i2c_prd_start

Address：0x40018010



Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_s_ph_3	r/w	8'd15	Length of START condition phase 3 (unit: I2C source clock period)
23:16	cr_i2c_prd_s_ph_2	r/w	8'd15	Length of START condition phase 2 (unit: I2C source clock period)
15:8	cr_i2c_prd_s_ph_1	r/w	8'd15	Length of START condition phase 1 (unit: I2C source clock period)

Bits	Name	Type	Reset	Description
7:0	cr_i2c_prd_s_ph_0	r/w	8'd15	Length of START condition phase 0 (unit: I2C source clock period)

11.9.6 i2c_prd_stop

Address：0x40018014

cr_i2c_prd_p_ph_3								cr_i2c_prd_p_ph_2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_i2c_prd_p_ph_1								cr_i2c_prd_p_ph_0							

Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_p_ph_3	r/w	8'd15	Length of STOP condition phase 3 (unit: I2C source clock period)
23:16	cr_i2c_prd_p_ph_2	r/w	8'd15	Length of STOP condition phase 2 (unit: I2C source clock period)
15:8	cr_i2c_prd_p_ph_1	r/w	8'd15	Length of STOP condition phase 1 (unit: I2C source clock period)
7:0	cr_i2c_prd_p_ph_0	r/w	8'd15	Length of STOP condition phase 0 (unit: I2C source clock period)

11.9.7 i2c_prd_data

Address：0x40018018

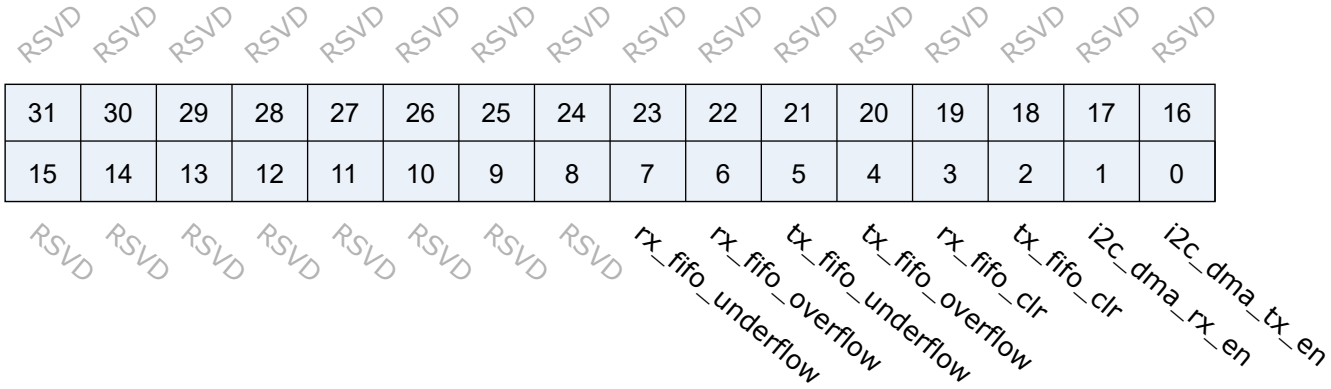
cr_i2c_prd_d_ph_3								cr_i2c_prd_d_ph_2							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_i2c_prd_d_ph_1								cr_i2c_prd_d_ph_0							

Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_d_ph_3	r/w	8'd15	Length of DATA phase 3 (unit: I2C source clock period)
23:16	cr_i2c_prd_d_ph_2	r/w	8'd15	Length of DATA phase 2 (unit: I2C source clock period)

Bits	Name	Type	Reset	Description
15:8	cr_i2c_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 (unit: I2C source clock period) Note: This value should not be set to 8'd0, adjust source clock rate instead if higher I2C clock rate is required
7:0	cr_i2c_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (unit: I2C source clock period)

11.9.8 i2c_fifo_config_0

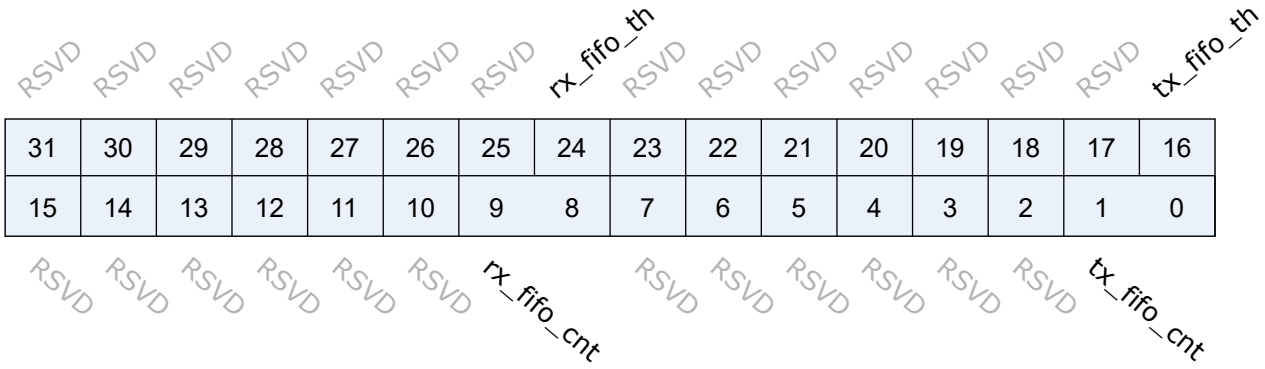
Address：0x40018080



Bits	Name	Type	Reset	Description
31:8	RSVD			
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO, RX FIFO will be empty when write 1 to this bit
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO, TX FIFO will be empty when write 1 to this bit
1	i2c_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	i2c_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

11.9.9 i2c_fifo_config_1

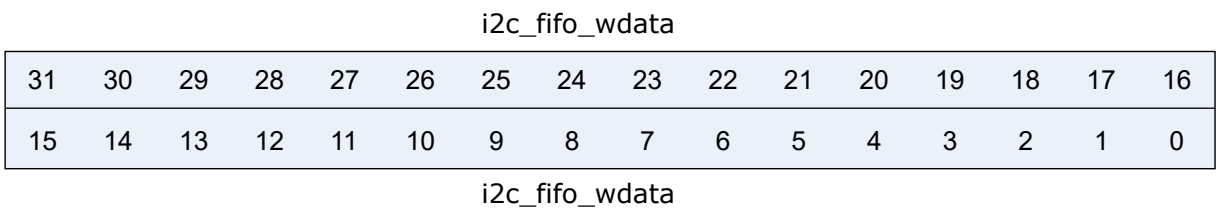
Address：0x40018084



Bits	Name	Type	Reset	Description
31:25	RSVD			
24	rx_fifo_th	r/w	1'd0	RX FIFO threshold, dma_rx_req will not be asserted if rx_-fifo_cnt is less than this value
23:17	RSVD			
16	tx_fifo_th	r/w	1'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_-fifo_cnt is less than this value
15:10	RSVD			
9:8	rx_fifo_cnt	r	2'd0	RX FIFO available count, means count of data received in RX FIFO
7:2	RSVD			
1:0	tx_fifo_cnt	r	2'd2	TX FIFO available count, means empty space remained in TX FIFO

11.9.10 i2c_fifo_wdata

Address：0x40018088



Bits	Name	Type	Reset	Description
31:0	i2c_fifo_wdata	w	x	TX FIFO, size is 4*2 = 8-byte

11.9.11 i2c_fifo_rdata

Address：0x4001808c

i2c_fifo_rdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2c_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	i2c_fifo_rdata	r	32'h0	RX FIFO, size is 4*2 = 8-byte

12 PWM

12.1 Overview

Pulse Width Modulation (PWM), an efficient technique that uses the digital signal of microprocessor to control the analog circuit, is widely used in fields like measurement, communication, and power control and conversion.

12.2 Features

- One PWM signals are generated, each containing 4-channel PWM signal outputs, with 2 pairs of complementary PWM per channel
- There are optional three clock sources (bclk、xclk、f32k), with 16-bit clock divider, up to 65535 divisions
- Each PWM channel has double threshold setting, allowing different duty ratios and phases, with more flexible pulse
- Each PWM channel has independent dead time setting
- Different active levels can be set for each PWM output pin
- Each PWM has an independent linked switch to connect/disconnect with the internal counter, and you can set the default output level when disconnected
- The software and external brake signals can set the PWM output level to a preset state
- Up to 9 trigger sources for triggering ADC conversion
- Support 10 interrupts

12.3 Functional Description

12.3.1 Clock and Frequency Divider

There are three options for PWM counter clock sources, which can be set through the reg_clk_sel field in the pwm_mc0_config0 register, and the clock source configuration is shown in the following table.

Table 12.1: Clock source

reg_clk_sel	PWM clock source
0	xclk
1	bclk
others	f32k

Each counter has its own 16-bit frequency divider, which can divide the selected clock through APB. The PWM counter will take the divided clock as the counting cycle unit, and add one every time a counting cycle ends.

12.3.2 Active Level

Different application scenarios require different active level states, some being active at low level and others being active at high level. The active level of each PWM output can be set independently.

When the <pwm_chy_ppl> bit in the register pwm_mcx_config1 is set to 1, it means that the forward channel of channel y (y = 0, 1, 2, 3) of pwm is set to be active at high level. That is, when the PWM module outputs logical ‘1’, its corresponding pin outputs high level, and when it outputs logical ‘0’, that outputs low level. When <pwm_chy_ppl> is set to 0, it means that the same is set to be active at low level. That is, when the PWM module outputs logical ‘1’, its corresponding pin outputs low level, and when it outputs logical ‘0’, that outputs high level. The setting bit of the complementary channel is <pwm_chy_npl>, and its setting rule is the same as that of the forward channel.

12.3.3 Principle of Pulse Generation

When the <pwm_chy_pen> bit in the register pwm_mcx_config1 is set to 0, it means that the forward channel of channel y (y = 0, 1, 2, 3) of pwm is set to a determined logic state, and then the state is determined by the <pwm_chy_psi> bit. When <pwm_chy_psi> is 0, the corresponding forward channel pin outputs logical ‘0’, and when the same is 1, that outputs logical ‘1’. The actual output level must be jointly determined with <pwm_chy_ppl>. The setting bit of the complementary channel is <pwm_chy_nen>, and its setting rule is the same as that of the forward channel.

When the <pwm_chy_pen> (<pwm_chy_nen>) bit in the register pwm_mcx_config1 is set to 1, the forward (complementary) channel output of channel y (y = 0, 1, 2, 3) of pwm is determined by comparing the internal counter with the two thresholds. If the counter value is between the two thresholds, the forward channel outputs logical ‘1’, while the complementary one outputs logical ‘0’. If the counter value is beyond the two thresholds, the forward channel

outputs logical ‘0’, while the complementary one outputs logical ‘1’. The waveform is shown as follows.

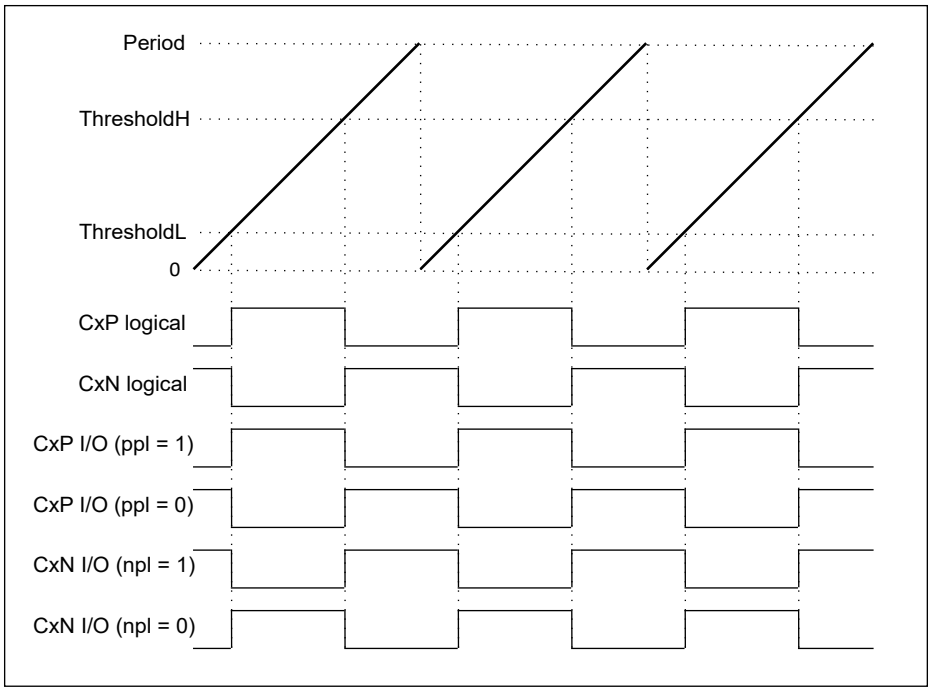


Fig. 12.1: Waveform of PWM in different configurations

12.3.4 Brake

When the bit <pwm_sw_break_en> of pwm_mc0_config0 is 1, the brake signal is generated, and the level of the PWM output will be modified to the preset state. The preset state is set by the bits <pwm_chy_pbs> and <pwm_chy_nbs> (y=0,1,2,3) in the pwm_mc0_config1 register. When this bit is set to 1, after the brake signal is generated, the PWM output logic 1, when this bit is set to 0, this PWM output logic 0 after the brake signal is generated. When <pwm_sw_break_en> is 0, it exits the braking state, and the PWM resumes the previous operating mode. The brake signal does not trigger an interrupt.

12.3.5 Dead Zone

Different dead time can be set for each PWM channel independently. When the value of dead time is not 0, the forward channel of PWM will delay the generation of jump level when it matches the ThresholdL, and immediately change the level state when it matches the ThresholdH. The complementary channel of PWM will change the level state immediately when it matches the ThresholdL, and delay the generation of jump level when it matches the ThresholdH. The length of dead zone is set by the register pwm_mc0_dead_time, the deadband length is calculated as shown in the following table.

Table 12.2: Deadband length calculation formula

pwm_chy_dtg	Deadband length
0xxxxxxx	pwm_chy_dtg[7:0]

Table 12.2: Deadband length calculation formula(continued)

pwm_chy_dtg	Deadband length
10xxxxxx	(64 + pwm_chy_dtg[5:0]) * 2
110xxxxx	(32 + pwm_chy_dtg[4:0]) * 8
111xxxxx	(32 + pwm_chy_dtg[4:0]) * 16

The output waveform after inserting the deadband for PWM is shown in the following figure.

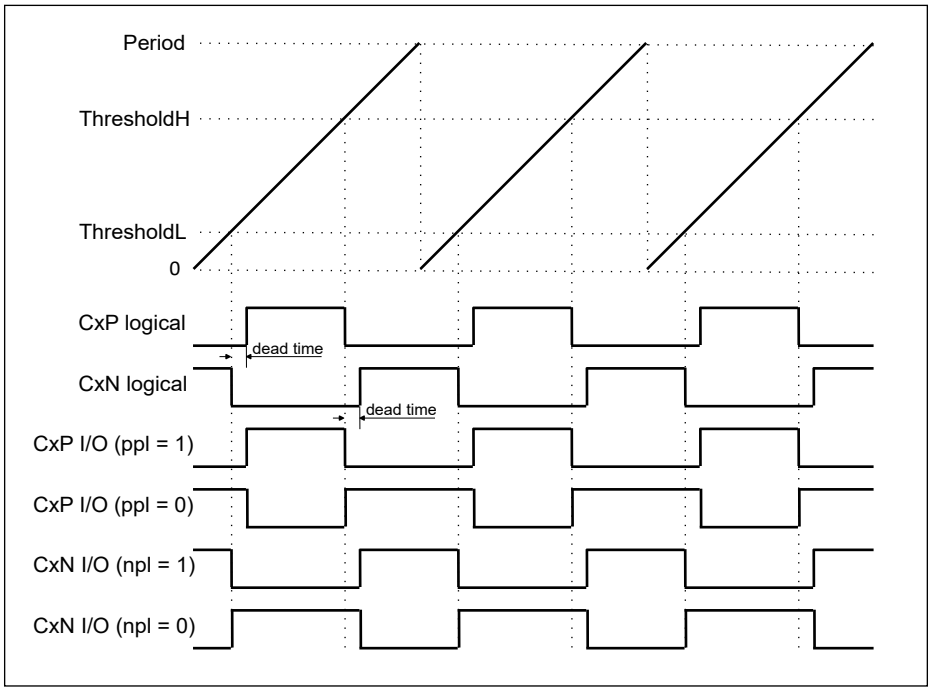


Fig. 12.2: PWM insertion deadband output waveform

12.3.6 Cycle and Duty Ratio Calculation

The period of the PWM is determined by two parts, the clock division factor and the clock duration period. The clock division factor is set by the domain pwm_clk_div in register pwm_mc0_config0, which is used to divide the source clock of the PWM. The value of the domain is the frequency division factor, it should be noted that the domain is 0 or 1 means 1 division, the maximum is 65535 means 65535 division.

The clock duration period is set by the domain pwm_period in the register pwm_mc0_period, which is used to set how many divided clock periods a PWM cycle consists of.

PWM period = PWM source clock / pwm_clk_div / pwm_period

12.3.7 PWM Interrupt

The PWM can generate a total of 10 types of interrupts, described as follows.

- Channel 0 ThresholdL Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of the field pwm_ch0_threL in the pwm_mc0_ch0_thre register.
- Channel 0 ThresholdH Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of the field pwm_ch0_threH in the pwm_mc0_ch0_thre register.
- Channel 1 ThresholdL Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of field pwm_ch1_threL in the pwm_mc0_ch1_thre register.
- Channel 1 ThresholdH Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of the field pwm_ch1_threH in the pwm_mc0_ch1_thre register.
- Channel 2 ThresholdL Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of field pwm_ch2_threL in the pwm_mc0_ch2_thre register.
- Channel 2 Threshold ThresholdH Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of the field pwm_ch2_threH in the pwm_mc0_ch2_thre register.
- Channel 3 ThresholdL Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of the field pwm_ch3_threL in the pwm_mc0_ch3_thre register.
- Channel 3 ThresholdH Interrupt
  - This interrupt is generated when the PWM count value is equal to the value of the field pwm_ch3_threH in the pwm_mc0_ch3_thre register.
- End-of-cycle interrupt
  - This interrupt is generated when the PWM count value is equal to the value of field pwm_period in register pwm_mc0_period, indicating the end of a PWM cycle.
- Cycle Repeat Interrupt
  - This interrupt is generated when the number of PWM cycles reaches the value set in the field pwm_int_-

period_cnt in the register pwm_mc0_period. This interrupt can be used in stepper motor application scenarios, where setting pwm_int_period_cnt allows the stepper motor to generate an interrupt after rotating a specific angle.

12.3.8 ADC Linkage

When the counter matches the threshold or the count value reaches the number of cycles, it will generate the signal that internally triggers ADC startup conversion. It should be noted that only PWM0 can trigger such conversion, while PWM1 cannot do that. The specific trigger source is set by the <pwm_adc_trg_src> bit in the register pwm_mcx_config0. This feature is commonly used in timing sampling. The following is an example.

Application scenario: In BLDC application, there is such a requirement that the current flowing through the coil shall be detected while the motor speed is controlled by PWM. In a PWM cycle, after PWM controls the power device to turn on, the current becomes stable after a certain period of time. Then, it is necessary to sample the current value, which means that there is a strict phase difference between the time point of triggering ADC conversion and PWM. For example, if the channel 0 of PWM is used to drive one of the phases of the motor, and a 10 KHz square wave with a duty ratio of 20% needs to be generated, and ADC sampling needs to be performed at the middle time point of the high level of the square wave, then the PWM cycle is 100us. When the clock source is 1 MHz, the cycle count value is 100, and the two thresholds of channel 1 can be set to 0 and 20 respectively. At this time, the counter is between 0 and 20, and PWM outputs a high level, and otherwise it outputs a low level. When the threshold L of channel 2 is set to 10 and the pwm_adc_trg_src in the register pwm_mc0_config0 is set to 4, pwm_ch2l_int can trigger ADC conversion. When the counter counts to 10, ADC will start sampling conversion at the middle time point of the high level generated by channel 1. This ensures that each sampling can meet the exact time requirement without CPU intervention, thus improving the performance.

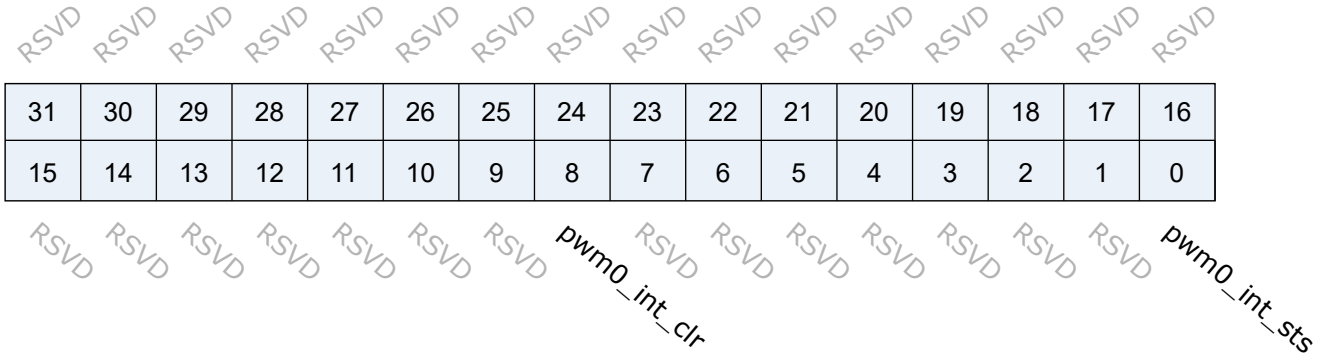
12.4 Register description

Name	Description
pwm_int_config	PWM interrupt config register
pwm_mc0_config0	PWM config0 register
pwm_mc0_config1	PWM config1 register
pwm_mc0_period	PWM period register
pwm_mc0_dead_time	PWM dead time register
pwm_mc0_ch0_thre	PWM channel0 threshold register
pwm_mc0_ch1_thre	PWM channel1 threshold register
pwm_mc0_ch2_thre	PWM channel2 threshold register
pwm_mc0_ch3_thre	PWM channel3 threshold register
pwm_mc0_int_sts	PWM interrupt status register

Name	Description
pwm_mc0_int_mask	PWM interrupt mask register
pwm_mc0_int_clear	PWM interrupt clear register
pwm_mc0_int_en	PWM interrupt enable register

12.4.1 pwm_int_config

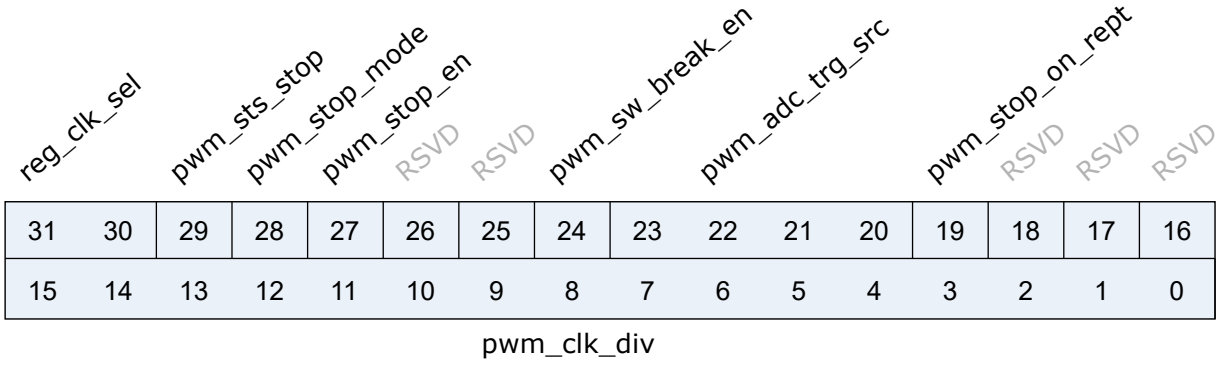
Address：0x2000a400



Bits	Name	Type	Reset	Description
31:9	RSVD			
8	pwm0_int_clr	w1c	1'b0	PWM 0 interrupt clear (clear pwm_mc0_int_sts[10:0] all)
7:1	RSVD			
0	pwm0_int_sts	r	1'b0	PWM 0 interrupt status (Check pwm_mc0_int_sts for detailed interrupt status)

12.4.2 pwm_mc0_config0

Address：0x2000a440

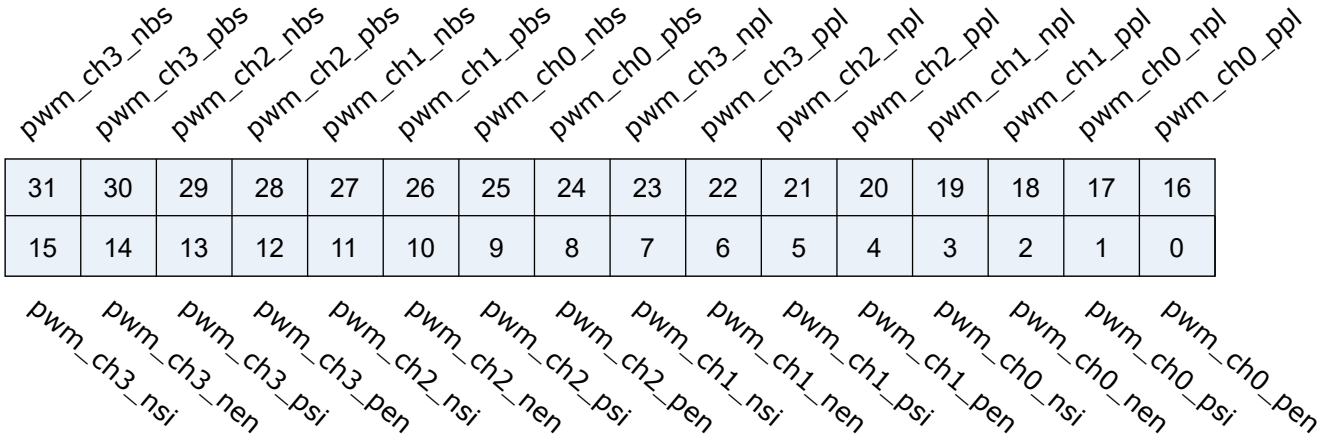


Bits	Name	Type	Reset	Description
31:30	reg_clk_sel	r/w	2'd0	PWM clock source select 2'd0: xclk 2'd1: bclk others: f32k_clk
29	pwm_sts_stop	r	1'b0	PWM stop status 0: PWM run status 1: PWM stop status
28	pwm_stop_mode	r/w	1'b1	PWM stop mode, 0: abrupt 1: graceful
27	pwm_stop_en	r/w	1'b0	PWM stop enable 0: start PWM 1: stop PWM
26:25	RSVD			
24	pwm_sw_break_en	r/w	1'b0	PWM break enable 0: Disabled, normal operation 1: Enabled, PWM output will be determined by CxPBS/CxNBS
23:20	pwm_adc_trg_src	r/w	4'hF	Select signal of ADC triggering source 4'd0: pwm_ch0l_int (Channel 0 ThresholdL reached) 4'd1: pwm_ch0h_int (Channel 0 ThresholdH reached) 4'd2: pwm_ch1l_int (Channel 1 ThresholdL reached) 4'd3: pwm_ch1h_int (Channel 1 ThresholdH reached) 4'd4: pwm_ch2l_int (Channel 2 ThresholdL reached) 4'd5: pwm_ch2h_int (Channel 2 ThresholdH reached) 4'd6: pwm_ch3l_int (Channel 3 ThresholdL reached) 4'd7: pwm_ch3h_int (Channel 3 ThresholdH reached) 4'd8: pwm_prde_int (Period End reached) Others: Disabled
19	pwm_stop_on_rept	r/w	1'b0	PWM stopped when rept_int is asserted 0: Disabled, PWM keeps running when rept_int is asserted 1: Enabled, PWM stops when rept_int is asserted. Clear rept_int to restore operation
18:16	RSVD			

Bits	Name	Type	Reset	Description
15:0	pwm_clk_div	r/w	16'b0	PWM clock division 0: 1 divisor 1: 1 divisor 2: 2 divisor ..... 65535: 65535 divisor

12.4.3 pwm_mc0_config1

Address：0x2000a444



Bits	Name	Type	Reset	Description
31	pwm_ch3_nbs	r/w	1'b0	PWM channel 3 negative break state 0: PWM channel 3 negative output logic 0 in break state 1: PWM channel 3 negativeitive output logic 1 in break state
30	pwm_ch3_pbs	r/w	1'b0	PWM channel 3 positive break state 0: PWM channel 3 positive output logic 0 in break state 1: PWM channel 3 positive output logic 1 in break state
29	pwm_ch2_nbs	r/w	1'b0	PWM channel 2 negative break state 0: PWM channel 2 negative output logic 0 in break state 1: PWM channel 2 negativeitive output logic 1 in break state
28	pwm_ch2_pbs	r/w	1'b0	PWM channel 2 positive break state 0: PWM channel 2 positive output logic 0 in break state 1: PWM channel 2 positive output logic 1 in break state
27	pwm_ch1_nbs	r/w	1'b0	PWM channel 1 negative break state 0: PWM channel 1 negative output logic 0 in break state 1: PWM channel 1 negativeitive output logic 1 in break state

Bits	Name	Type	Reset	Description
26	pwm_ch1_pbs	r/w	1'b0	PWM channel 1 positive break state 0: PWM channel 1 positive output logic 0 in break state 1: PWM channel 1 positive output logic 1 in break state
25	pwm_ch0_nbs	r/w	1'b0	PWM channel 0 negative break state 0: PWM channel 0 negative output logic 0 in break state 1: PWM channel 0 negativeitive output logic 1 in break state
24	pwm_ch0_pbs	r/w	1'b0	PWM channel 0 positive break state 0: PWM channel 0 positive output logic 0 in break state 1: PWM channel 0 positive output logic 1 in break state
23	pwm_ch3_npl	r/w	1'b1	PWM channel 3 negative polarity 0: PWM channel 3 negative output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 3 negative output low level in logic 0 state, output high level in logic 1 state
22	pwm_ch3_ppl	r/w	1'b1	PWM channel 3 positive polarity 0: PWM channel 3 positive output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 3 positive output low level in logic 0 state, output high level in logic 1 state
21	pwm_ch2_npl	r/w	1'b1	PWM channel 2 negative polarity 0: PWM channel 2 negative output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 2 negative output low level in logic 0 state, output high level in logic 1 state
20	pwm_ch2_ppl	r/w	1'b1	PWM channel 2 positive polarity 0: PWM channel 2 positive output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 2 positive output low level in logic 0 state, output high level in logic 1 state
19	pwm_ch1_npl	r/w	1'b1	PWM channel 1 negative polarity 0: PWM channel 1 negative output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 1 negative output low level in logic 0 state, output high level in logic 1 state
18	pwm_ch1_ppl	r/w	1'b1	PWM channel 1 positive polarity 0: PWM channel 1 positive output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 1 positive output low level in logic 0 state, output high level in logic 1 state

Bits	Name	Type	Reset	Description
17	pwm_ch0_npl	r/w	1'b1	PWM channel 0 negative polarity 0: PWM channel 0 negative output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 0 negative output low level in logic 0 state, output high level in logic 1 state
16	pwm_ch0_ppl	r/w	1'b1	PWM channel 0 positive polarity 0: PWM channel 0 positive output high level in logic 0 state, output low level in logic 1 state 1: PWM channel 0 positive output low level in logic 0 state, output high level in logic 1 state
15	pwm_ch3_nsi	r/w	1'b1	PWM channel 3 negative set idle state 0: PWM channel 3 negative logic 0 state 1: PWM channel 3 negative logic 1 state
14	pwm_ch3_nen	r/w	1'b0	PWM channel 3 negative enable pwm out 0: disable, PWM output is determined by pwm_ch3_nsi and pwm_ch3_npl 1: enable, PWM output is determined by counter and pwm_mc0_ch3_thre
13	pwm_ch3_psi	r/w	1'b0	PWM channel 3 positive set idle state 0: PWM channel 3 positive output logic 0 in idle state 1: PWM channel 3 positive output logic 1 in idle state
12	pwm_ch3_pen	r/w	1'b0	PWM channel 3 positive enable pwm out 0: disable, PWM output is determined by pwm_ch3_psi and pwm_ch3_ppl 1: enable, PWM output is determined by counter and pwm_mc0_ch3_thre
11	pwm_ch2_nsi	r/w	1'b1	PWM channel 2 negative set idle state 0: PWM channel 2 negative logic 0 state 1: PWM channel 2 negative logic 1 state
10	pwm_ch2_nen	r/w	1'b0	PWM channel 2 negative enable pwm out 0: disable, PWM output is determined by pwm_ch2_nsi and pwm_ch2_npl 1: enable, PWM output is determined by counter and pwm_mc0_ch2_thre
9	pwm_ch2_psi	r/w	1'b0	PWM channel 2 positive set idle state 0: PWM channel 2 positive output logic 0 in idle state 1: PWM channel 2 positive output logic 1 in idle state

Bits	Name	Type	Reset	Description
8	pwm_ch2_pen	r/w	1'b0	PWM channel 2 positive enable pwm out 0: disable, PWM output is determined by pwm_ch2_psi and pwm_ch2_ppl 1: enable, PWM output is determined by counter and pwm_mc0_ch2_thre
7	pwm_ch1_nsi	r/w	1'b1	PWM channel 1 negative set idle state 0: PWM channel 1 negative logic 0 state 1: PWM channel 1 negative logic 1 state
6	pwm_ch1_nen	r/w	1'b0	PWM channel 1 negative enable pwm out 0: disable, PWM output is determined by pwm_ch1_nsi and pwm_ch1_npl 1: enable, PWM output is determined by counter and pwm_mc0_ch1_thre
5	pwm_ch1_psi	r/w	1'b0	PWM channel 1 positive set idle state 0: PWM channel 1 positive output logic 0 in idle state 1: PWM channel 1 positive output logic 1 in idle state
4	pwm_ch1_pen	r/w	1'b0	PWM channel 1 positive enable pwm out 0: disable, PWM output is determined by pwm_ch1_psi and pwm_ch1_ppl 1: enable, PWM output is determined by counter and pwm_mc0_ch1_thre
3	pwm_ch0_nsi	r/w	1'b1	PWM channel 0 negative set idle state 0: PWM channel 0 negative logic 0 state 1: PWM channel 0 negative logic 1 state
2	pwm_ch0_nen	r/w	1'b0	PWM channel 0 negative enable pwm out 0: disable, PWM output is determined by pwm_ch0_nsi and pwm_ch0_npl 1: enable, PWM output is determined by counter and pwm_mc0_ch0_thre
1	pwm_ch0_psi	r/w	1'b0	PWM channel 0 positive set idle state 0: PWM channel 0 positive output logic 0 in idle state 1: PWM channel 0 positive output logic 1 in idle state
0	pwm_ch0_pen	r/w	1'b0	PWM channel 0 positive enable pwm out 0: disable, PWM output is determined by pwm_ch0_psi and pwm_ch0_ppl 1: enable, PWM output is determined by counter and pwm_mc0_ch0_thre

12.4.4 pwm_mc0_period

Address：0x2000a448

pwm_int_period_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm_period

Bits	Name	Type	Reset	Description
31:16	pwm_int_period_cnt	r/w	16'd0	PWM interrupt period counter threshold
15:0	pwm_period	r/w	16'd0	PWM period setting

12.4.5 pwm_mc0_dead_time

Address：0x2000a44c

pwm_ch3_dtg　　pwm_ch2_dtg

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm_ch1_dtg　　pwm_ch0_dtg

Bits	Name	Type	Reset	Description
31:24	pwm_ch3_dtg	r/w	8'h0	PWM Channel 3 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

Bits	Name	Type	Reset	Description
23:16	pwm_ch2_dtg	r/w	8'h0	PWM Channel 2 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
15:8	pwm_ch1_dtg	r/w	8'h0	PWM Channel 1 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)
7:0	pwm_ch0_dtg	r/w	8'h0	PWM Channel 0 dead time generator (DTG) DTG[7:5]=0xx => DT = DTG[7:0]*1 (unit: divided PWM clock) DTG[7:5]=10x => DT = (64+DTG[5:0])*2 (unit: divided PWM clock) DTG[7:5]=110 => DT = (32+DTG[4:0])*8 (unit: divided PWM clock) DTG[7:5]=111 => DT = (32+DTG[4:0])*16 (unit: divided PWM clock)

12.4.6 pwm_mc0_ch0_thre

Address：0x2000a450

pwm_ch0_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm_ch0_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch0_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller that threL
15:0	pwm_ch0_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger that threH

12.4.7 pwm_mc0_ch1_thre

Address：0x2000a454

pwm_ch1_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm_ch1_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch1_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller that threL
15:0	pwm_ch1_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger that threH

12.4.8 pwm_mc0_ch2_thre

Address：0x2000a458

pwm_ch2_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm_ch2_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch2_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller that threL
15:0	pwm_ch2_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger that threH

12.4.9 pwm_mc0_ch3_thre

Address：0x2000a45c

pwm_ch3_threH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

pwm_ch3_threL

Bits	Name	Type	Reset	Description
31:16	pwm_ch3_threH	r/w	16'd0	PWM HIGH counter threshold, can't be smaller that threL
15:0	pwm_ch3_threL	r/w	16'd0	PWM LOW counter threshold, can't be larger that threH

12.4.10 pwm_mc0_int_sts

Address：0x2000a460

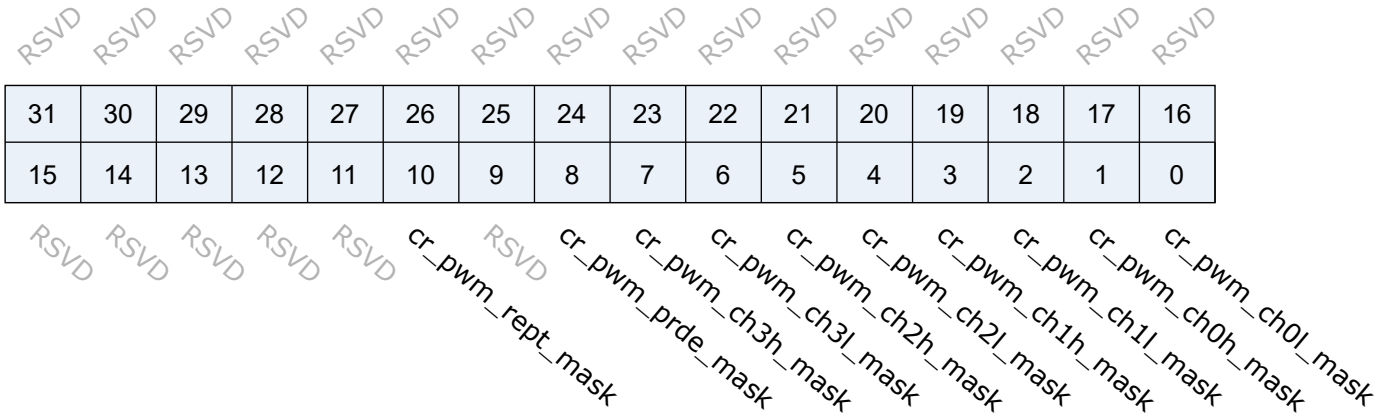
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	pwm_rept_int	RSVD	pwm_prde_int	pwm_ch3h_int	pwm_ch3l_int	pwm_ch2h_int	pwm_ch2l_int	pwm_ch1h_int	pwm_ch1l_int	pwm_ch0h_int	pwm_ch0l_int

Bits	Name	Type	Reset	Description
31:11	RSVD			
10	pwm_rept_int	r	1'b0	PWM repeat count interrupt status
9	RSVD			
8	pwm_prde_int	r	1'b0	PWM period end interrupt status
7	pwm_ch3h_int	r	1'b0	PWM Channel 3 ThresholdH interrupt status
6	pwm_ch3l_int	r	1'b0	PWM Channel 3 ThresholdL interrupt status
5	pwm_ch2h_int	r	1'b0	PWM Channel 2 ThresholdH interrupt status
4	pwm_ch2l_int	r	1'b0	PWM Channel 2 ThresholdL interrupt status
3	pwm_ch1h_int	r	1'b0	PWM Channel 1 ThresholdH interrupt status
2	pwm_ch1l_int	r	1'b0	PWM Channel 1 ThresholdL interrupt status

Bits	Name	Type	Reset	Description
1	pwm_ch0h_int	r	1'b0	PWM Channel 0 ThresholdH interrupt status
0	pwm_ch0l_int	r	1'b0	PWM Channel 0 ThresholdL interrupt status

12.4.11 pwm_mc0_int_mask

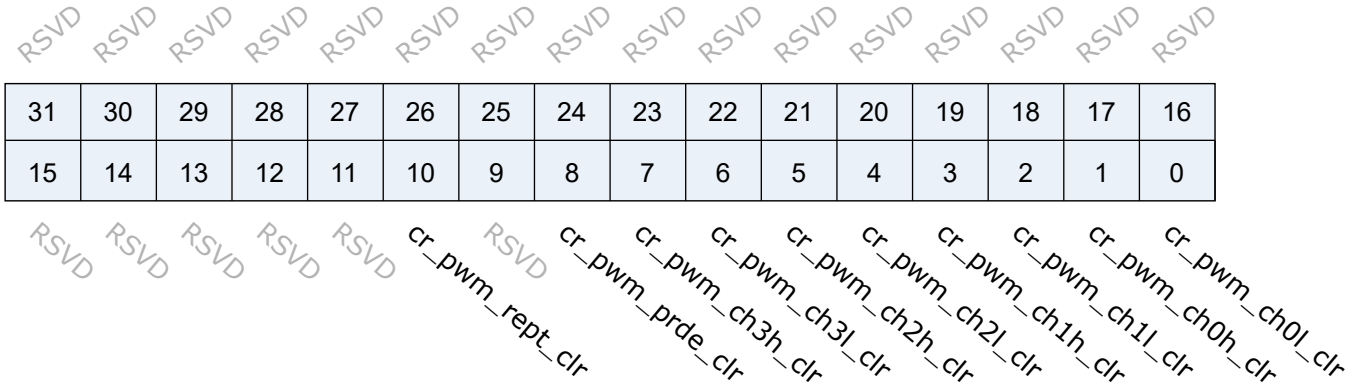
Address：0x2000a464



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_mask	r/w	1'b1	Interrupt mask of pwm_rept_int
9	RSVD			
8	cr_pwm_prde_mask	r/w	1'b1	Interrupt mask of pwm_prde_int
7	cr_pwm_ch3h_mask	r/w	1'b1	Interrupt mask of pwm_ch3h_int
6	cr_pwm_ch3l_mask	r/w	1'b1	Interrupt mask of pwm_ch3l_int
5	cr_pwm_ch2h_mask	r/w	1'b1	Interrupt mask of pwm_ch2h_int
4	cr_pwm_ch2l_mask	r/w	1'b1	Interrupt mask of pwm_ch2l_int
3	cr_pwm_ch1h_mask	r/w	1'b1	Interrupt mask of pwm_ch1h_int
2	cr_pwm_ch1l_mask	r/w	1'b1	Interrupt mask of pwm_ch1l_int
1	cr_pwm_ch0h_mask	r/w	1'b1	Interrupt mask of pwm_ch0h_int
0	cr_pwm_ch0l_mask	r/w	1'b1	Interrupt mask of pwm_ch0l_int

12.4.12 pwm_mc0_int_clear

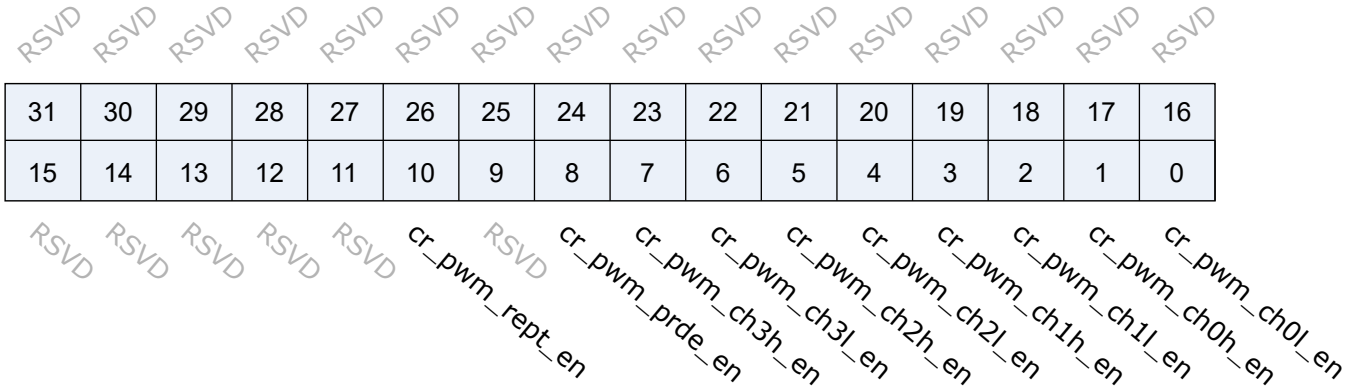
Address：0x2000a468



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_clr	w1c	1'b0	Interrupt clear of pwm_rept_int
9	RSVD			
8	cr_pwm_prde_clr	w1c	1'b0	Interrupt clear of pwm_prde_int
7	cr_pwm_ch3h_clr	w1c	1'b0	Interrupt clear of pwm_ch3h_int
6	cr_pwm_ch3l_clr	w1c	1'b0	Interrupt clear of pwm_ch3l_int
5	cr_pwm_ch2h_clr	w1c	1'b0	Interrupt clear of pwm_ch2h_int
4	cr_pwm_ch2l_clr	w1c	1'b0	Interrupt clear of pwm_ch2l_int
3	cr_pwm_ch1h_clr	w1c	1'b0	Interrupt clear of pwm_ch1h_int
2	cr_pwm_ch1l_clr	w1c	1'b0	Interrupt clear of pwm_ch1l_int
1	cr_pwm_ch0h_clr	w1c	1'b0	Interrupt clear of pwm_ch0h_int
0	cr_pwm_ch0l_clr	w1c	1'b0	Interrupt clear of pwm_ch0l_int

12.4.13 pwm_mc0_int_en

Address：0x2000a46c



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_en	r/w	1'b1	Interrupt enable of pwm_rept_int
9	RSVD			
8	cr_pwm_prde_en	r/w	1'b1	Interrupt enable of pwm_prde_int
7	cr_pwm_ch3h_en	r/w	1'b1	Interrupt enable of pwm_ch3h_int
6	cr_pwm_ch3l_en	r/w	1'b1	Interrupt enable of pwm_ch3l_int
5	cr_pwm_ch2h_en	r/w	1'b1	Interrupt enable of pwm_ch2h_int
4	cr_pwm_ch2l_en	r/w	1'b1	Interrupt enable of pwm_ch2l_int
3	cr_pwm_ch1h_en	r/w	1'b1	Interrupt enable of pwm_ch1h_int
2	cr_pwm_ch1l_en	r/w	1'b1	Interrupt enable of pwm_ch1l_int
1	cr_pwm_ch0h_en	r/w	1'b1	Interrupt enable of pwm_ch0h_int
0	cr_pwm_ch0l_en	r/w	1'b1	Interrupt enable of pwm_ch0l_int

13 TIMER

13.1 Overview

Two 32-bit counters are built in the chip, and each can independently control and configure its parameters and clock frequency.

There is one watchdog counter in the chip. Unpredictable software or hardware behavior may cause the application to malfunction, and the watchdog timer can help recover the system. If the current stage exceeds the preset time, but the watchdog is not reset or turned off, the interrupt or system reset can be triggered as configured.

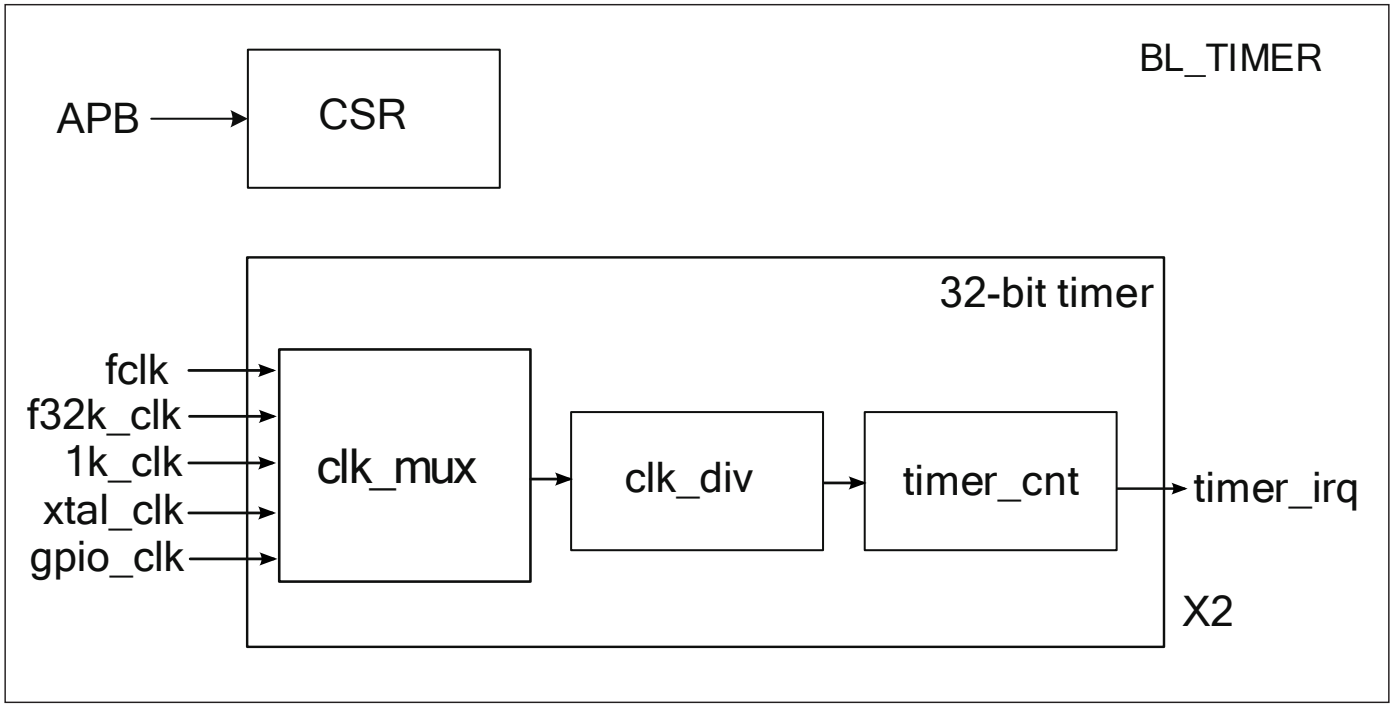


Fig. 13.1: Block diagram of timer

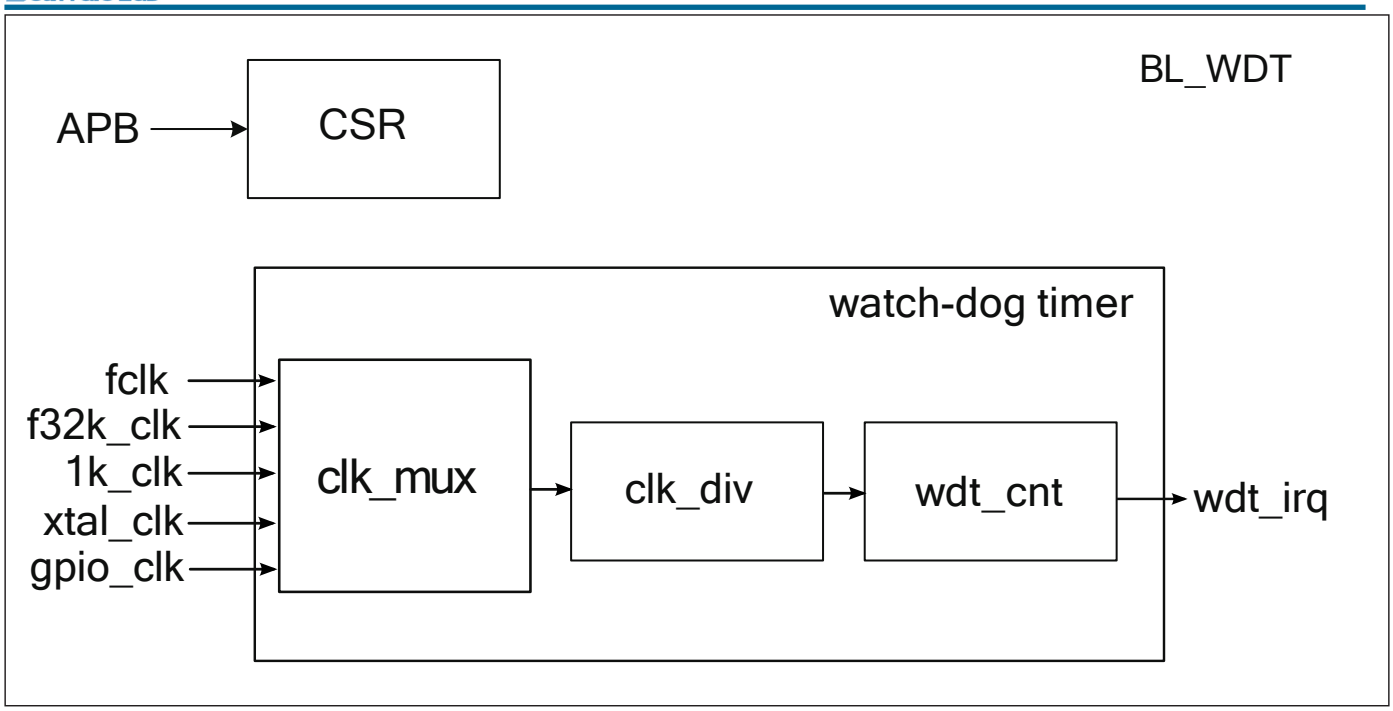


Fig. 13.2: Block diagram of watchdog timer

13.2 Features

- 32-bit timer
  - Multiple clock sources, up to 80M clock supported
  - Bit clock divider with a division factor of 1-256
  - Two 32-bit timers
  - Each timer has three alarm value settings, and the alarm when each set of alarm values overflows can be independently set.
  - Supports Free Run mode and Pre_load mode
  - Supports measuring the pulse width of external GPIO
- Watchdog timer
- With 16-bit watchdog timer
- Supports write password protection to prevent system error caused by wrong settings
- Supports two watchdog overflow modes: interrupt or reset

13.3 Functional Description

There are 5 watchdog timer clock sources, which can be selected via cs_wdt in the TCCR register:

- FCLK-bus clock
- 32K-32K clock
- 1K-1K clock
- XTAL-external crystal oscillator
- GPIO-external GPIO

There are 5 types of timer clock sources, which can be selected via cs_2 and cs_3 in register TCCR:

- FCLK-bus clock
- 32K-32K clock
- 1K-1K clock(32K frequency division)
- XTAL-external crystal oscillator
- GPIO-external GPIO

Each counter has its own 8-bit frequency divider, which can divide the clock by 1-256. Specifically, when it is set to 0, it means no frequency division. When it is set to 1, it will divide the clock by 2, and so on. The maximum division factor is 256, and the counter will take the divided clock as the counting cycle unit.

13.3.1 Working Principle of General Purpose Timer

Each general purpose timer contains three comparators, one counter, and one PreLoad register. When the clock source is set and the timer is started, the counter starts to count up cumulatively. When the value of counter is equal to that of the comparator, the comparison flag is set and a comparison interrupt can be generated.

You can configure the value of channel 0 comparator 0 by setting tclr2_0, that of channel 0 comparator 1 by setting tclr2_1, and that of channel 0 comparator 2 by setting tclr2_2 in the register TICR2. The tplvr2 in the register TPLVR2 sets the channel 0 preload value.

You can configure the value of channel 1 comparator 0 by setting tclr3_0, that of channel 1 comparator 1 by setting tclr3_1, and that of channel 1 comparator 2 by setting tclr3_2 in the register TICR3. The tplvr3 in the register TPLVR3 sets the channel 1 preload value.

The timer supports two counting modes, PreLoad mode and FreeRun mode, which are selected by register TCMR. The timer2_mode in register TCMR sets the counting mode of timer0, and the timer3_mode in register TCMR sets the counting mode of timer1.

13.3.1.1 PreLoad mode

The initial value of the counter in PreLoad mode is the value of the TPLVR register (Preload Register), from which the count is accumulated upwards. Register TPLVR2 sets the preload value of timer0, and register TPLVR3 configures the preload value of timer1. The tplcr2 in register TPLCR2 configures the PreLoad trigger condition of timer0, and the tplcr3 in register TPLCR3 configures the PreLoad trigger condition of timer1. When the PreLoad trigger condition is met, the counter value will be reset to the value of PreLoad register, and then the counter will start to count up again.

During the counter counting of the timer, once the counter value matches with one of the three comparators, the comparison flag of that comparator will be set and the corresponding comparison interrupt can be generated.

If the value of the PreLoad register is 10, and the values of comparators 0, 1, and 2 are 13, 16, and 19 respectively, the working sequence of the timer in the PreLoad mode is as follows:

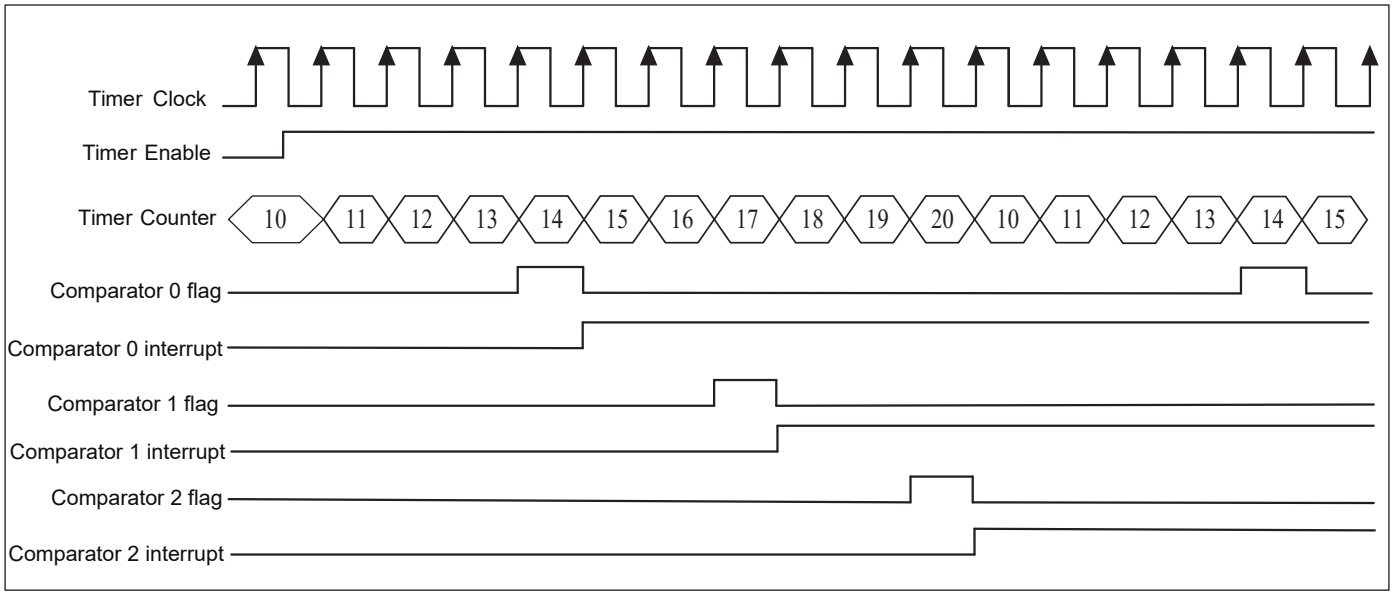


Fig. 13.3: Working sequence of timer in preLoad mode

13.3.1.2 FreeRun mode

FreeRun mode is the counter accumulation mode. In FreeRun mode, the initial value of the counter is 0. After starting the timer, the counter starts to accumulate from 0, and when the maximum value of the counter is reached, it starts to count from 0 again.

During the counter counting of the timer, once the counter value agrees with one of the three comparators. The comparison flag of that comparator will be set and the corresponding comparison interrupt can be generated.

In FreeRun mode, the timer works in the same timing as PreLoad, except that the counter will accumulate from 0 to the maximum value, and the mechanism of comparison flags and comparison interrupts generated during the period is the same as PreLoad mode.

13.3.1.3 Measuring External GPIO Pulse Width

The timer supports calculating the pulse width of external GPIOs using the internal clock source.

To configure, proceed as follows.

1. Configure the external GPIO to function as gpio_tmr_clk. This is achieved by configuring register dig_clk_cfg2 in the GLB module.

   Register dig_clk_cfg2[13:12] bits for gpio_tmr_clk_sel: Selects the GPIO clock mode. A bit in register dig_clk_-cfg2[11:8] is configured to 0 to indicate clock input mode. These two registers need to be used together and are configured as follows, and so on according to the GPIOs used.

Table 13.1: gpio_tmr_clk function configuration

GPIO	dig_clk_cfg2[13:12]	dig_clk_cfg2[11:8]
GPIO0	gpio_tmr_clk = 0	chip_clk_out_0_en = 0
GPIO1	gpio_tmr_clk = 1	chip_clk_out_1_en = 0
GPIO2	gpio_tmr_clk = 2	chip_clk_out_2_en = 0
GPIO3	gpio_tmr_clk = 3	chip_clk_out_3_en = 0
GPIO4	gpio_tmr_clk = 0	chip_clk_out_0_en = 0

2. The timer2_gpio_inv / timer3_gpio_inv bits in the GPIO register configure whether the external pulse width needs to be measured high or low. If this bit is 0, it means high level; if this bit is 1, it means low level. 3.
3. configure timer2_gpio_en in GPIO register to enable GPIO measurement function
4. Configure the timer clock source and operation mode, and enable timer
5. When gpio_lat_ok in GPIO register is set to 1, get the value of GPIO_LAT2 and GPIO_LAT1 register and calculate the width.

The calculation method of the pulse width of the external gpio: (GPIO_LAT2-GPIO_LAT1)* the width of 1 cycle of the internal clock source of the timer.

For example: the internal clock source of the timer is 80M, the frequency of the external gpio is 2M, and the duty ratio is 1:1. Write 1 to the timer2_gpio_inv bit to calculate the width of the low level of the external gpio.

- Write 1 to the timer2_gpio_inv bit, which means to calculate the width of the low level of the external gpio. After the above configuration is completed, the difference between the register GPIO_LAT2 and the register GPIO_LAT1 is 20, then the low level width of the external gpio is: 20 *(1 / 80000000) = 1 / 4000000
- Write 0 to the timer2_gpio_inv bit, which means to calculate the width of the high level of the external gpio.After the above configuration is completed, the difference between the register GPIO_LAT2 and the register GPIO_LAT1 is 20, then the high level width of the external gpio is: 20 *(1 / 80000000) = 1 / 4000000;

13.3.2 Working Principle of Watchdog Timer

The watchdog timer integrates a counter and a comparator. The counter counts up from 0 cumulatively. If the counter is reset, it counts up again from 0. When the value of counter is equal to that of the comparator, it can generate a comparison interrupt signal or a system reset signal. Users may use one of them as required. The watchdog counter will add 1 to each counting cycle unit, and the software can reset this counter to zero through APB at any time.

The wmr in the register WMR sets the comparison value.

If the comparator value is 6, the working sequence of Watchdog is shown as follows:

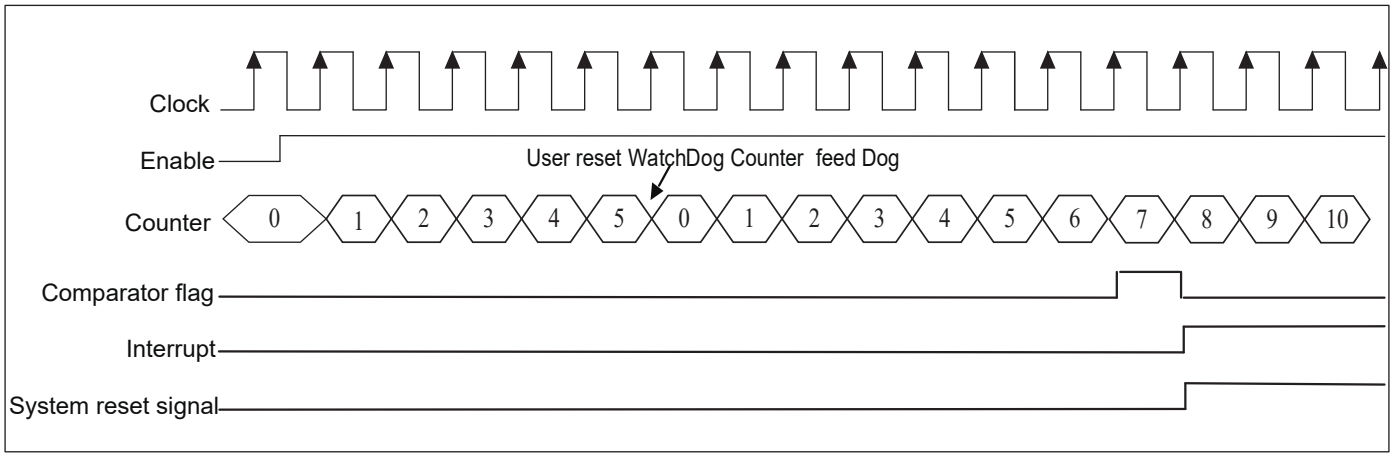


Fig. 13.4: Working sequence of watchdog

13.3.3 Alarm Setting

Each set of counter has three comparison values to provide software settings, and it can set whether each comparison value triggers an alarm interrupt. When the counter’s value matches the comparison value and an alarm will be given, the counter will notify the processor through interrupts.

Through APB, the software can read whether there is an alarm at present and which comparison value triggers the alarm interrupt. When the alarm interrupt is cleared, the alarm state will also be cleared synchronously.

13.3.4 Watchdog Alarm

Each counter can be configured with one comparison value. When the watchdog counter is too late to be reset to zero due to a system error, which causes the watchdog counter to exceed the comparison value, it will trigger the watchdog alarm. There are two alarm modes. One is to notify the software to process it by generating an interrupt. The other one is to perform watchdog reset. When the watchdog reset is triggered, it will notify the system’s reset controller and prepare for system reset. When everything is ready, the watchdog reset will be performed. It is worth noting that the software can read WSR register through APB to know whether watchdog reset has occurred.

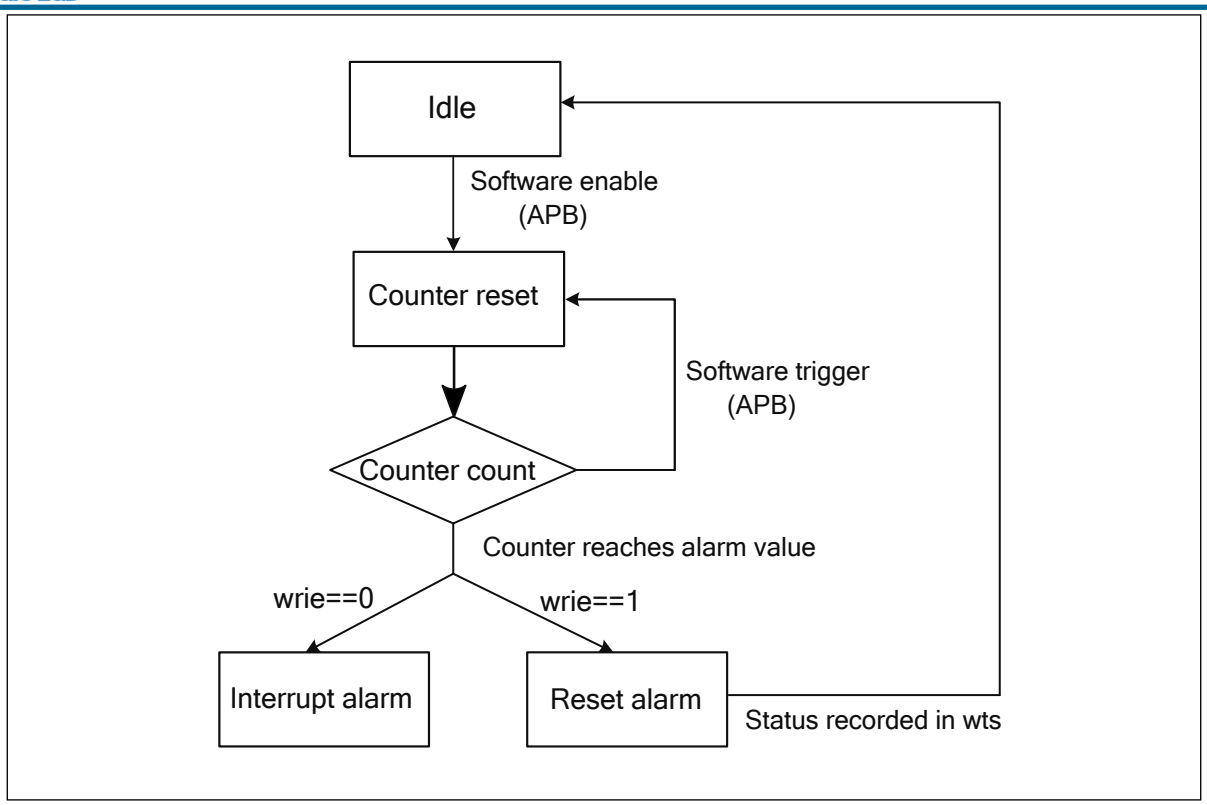


Fig. 13.5: Watchdog alarm mechanism

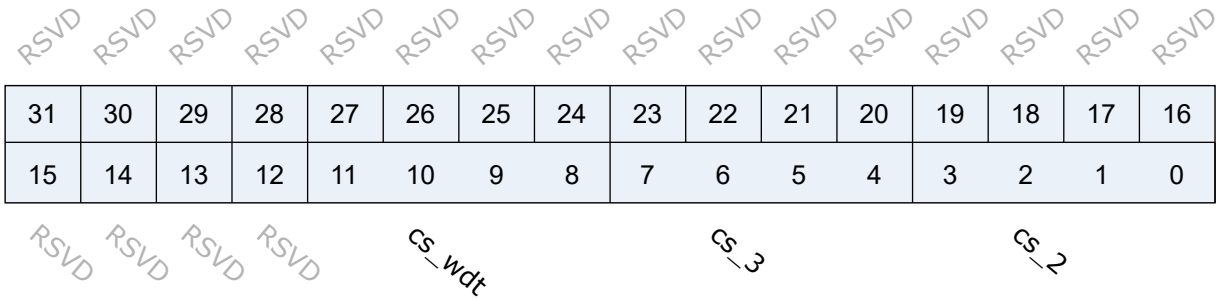
13.4 Register description

Name	Description
TCCR	Timer Clock Source
TMR2_0	Timer2 Match Value 0
TMR2_1	Timer2 Match Value 1
TMR2_2	Timer2 Match Value 2
TMR3_0	Timer3 Match Value 0
TMR3_1	Timer3 Match Value 1
TMR3_2	Timer3 Match Value 2
TCR2	Timer2 Counter Value
TCR3	Timer3 Counter Value
TSR2	Timer2 Match Status
TSR3	Timer3 Match Status
TIER2	Timer2 Match Interrupt Enable

Name	Description
TIER3	Timer3 Match Interrupt Enable
TPLVR2	Timer2 Pre-Load Value
TPLVR3	Timer3 Pre-Load Value
TPLCR2	Timer2 Pre-Load Control
TPLCR3	Timer3 Pre-Load Control
WMER	Watch-dog reset/interrupt Mode
WMR	Watch-dog Match Value
WVR	Watch-dog Counter Value
WSR	Watch-dog Reset Status
TICR2	Timer2 Interrupt Clear
TICR3	Timer3 Interrupt Clear
WICR	WDT Interrupt Clear
TCER	Timer Counter Enable/Clear
TCMR	Timer Counter Mode
TILR2	Timer2 Match Interrupt Mode
TILR3	Timer3 Match Interrupt Mode
WCR	WDT Counter Reset
WFAR	WDT Access Key1
WSAR	WDT Access Key2
TCDR	Timer Division
GPIO	GPIO Mode
GPIO_LAT1	GPIO Latch Value1
GPIO_LAT2	GPIO Latch Value2

13.4.1 TCCR

Address：0x2000a500



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	cs_wdt	r/w	4'd1	WDT 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock
7:4	cs_3	r/w	4'd5	Timer3 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock
3:0	cs_2	r/w	4'd5	Timer2 0:fclk / 1:f32k / 2:1k / 3:32M / 4:GPIO / 5:No clock

13.4.2 TMR2_0

Address：0x2000a510

tmr2_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr2_0

Bits	Name	Type	Reset	Description
31:0	tmr2_0	r/w	32'hffffffff	Timer2 Match Value 0

13.4.3 TMR2_1

Address：0x2000a514

tmr2_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr2_1

Bits	Name	Type	Reset	Description
31:0	tmr2_1	r/w	32'hffffffff	Timer2 Match Value 1

13.4.4 TMR2_2

Address：0x2000a518

tmr2_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr2_2

Bits	Name	Type	Reset	Description
31:0	tmr2_2	r/w	32'hffffffff	Timer2 Match Value 2

13.4.5 TMR3_0

Address：0x2000a51c

tmr3_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr3_0

Bits	Name	Type	Reset	Description
31:0	tmr3_0	r/w	32'hffffffff	Timer3 Match Value 0

13.4.6 TMR3_1

Address：0x2000a520

tmr3_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr3_1

Bits	Name	Type	Reset	Description
31:0	tmr3_1	r/w	32'hffffffff	Timer3 Match Value 1

13.4.7 TMR3_2

Address：0x2000a524

tmr3_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tmr3_2

Bits	Name	Type	Reset	Description
31:0	tmr3_2	r/w	32'hffffffff	Timer3 Match Value 2

13.4.8 TCR2

Address：0x2000a52c

tcr2_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tcr2_cnt

Bits	Name	Type	Reset	Description
31:0	tcr2_cnt	r	0	Timer2 Counter Value

13.4.9 TCR3

Address：0x2000a530

tcr3_cnt

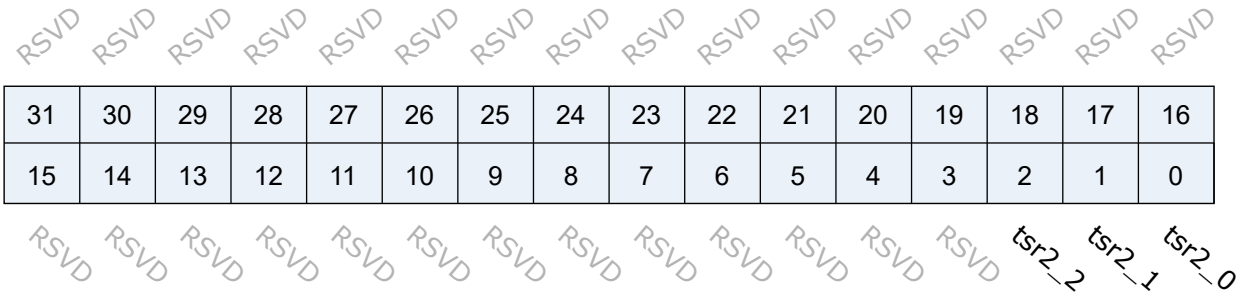
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tcr3_cnt

Bits	Name	Type	Reset	Description
31:0	tcr3_cnt	r	0	Timer3 Counter Value

13.4.10 TSR2

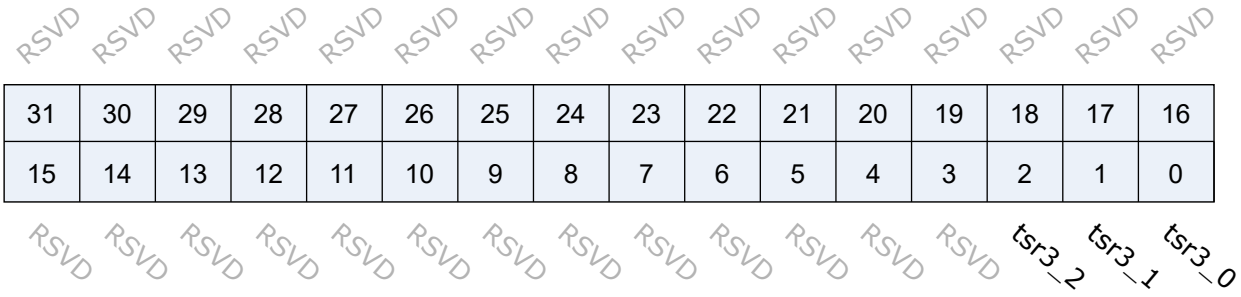
Address：0x2000a538



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tsr2_2	r	0	Timer2 match value 2 status/Clear interrupt would also clear this bit
1	tsr2_1	r	0	Timer2 match value 1 status/Clear interrupt would also clear this bit
0	tsr2_0	r	0	Timer2 match value 0 status/Clear interrupt would also clear this bit

13.4.11 TSR3

Address：0x2000a53c

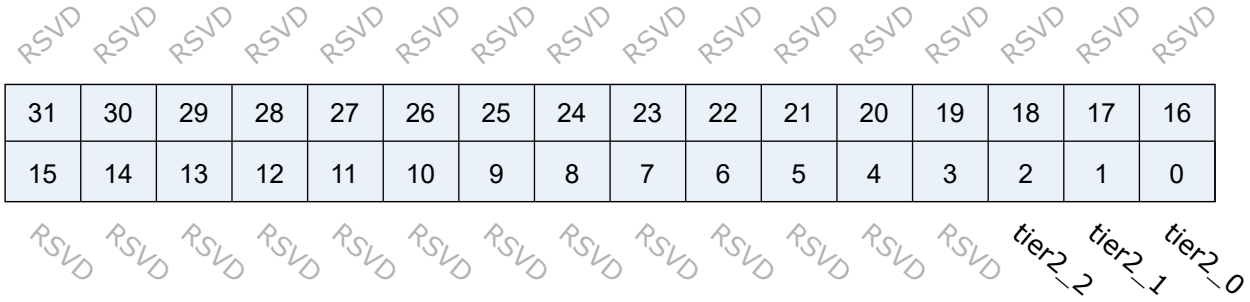


Bits	Name	Type	Reset	Description
31:3	RSVD			

Bits	Name	Type	Reset	Description
2	tsr3_2	r	0	Timer3 match value 2 status/Clear interrupt would also clear this bit
1	tsr3_1	r	0	Timer3 match value 1 status/Clear interrupt would also clear this bit
0	tsr3_0	r	0	Timer3 match value 0 status/Clear interrupt would also clear this bit

13.4.12 TIER2

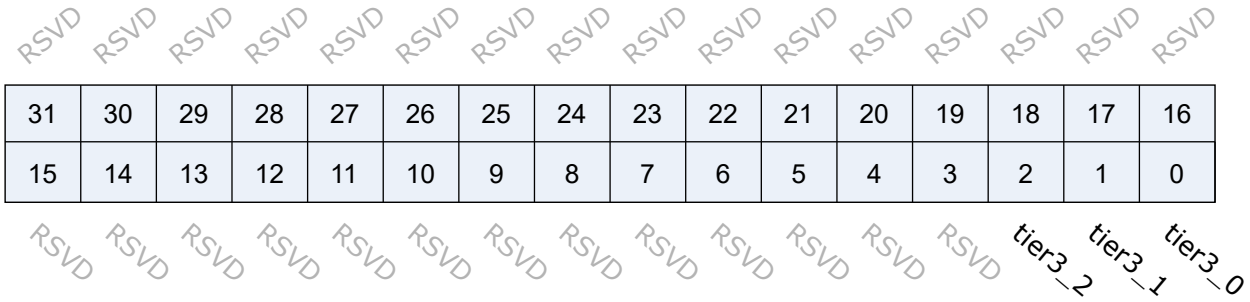
Address：0x2000a544



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tier2_2	r/w	0	Timer2 match value 2 interrupt enable
1	tier2_1	r/w	0	Timer2 match value 1 interrupt enable
0	tier2_0	r/w	0	Timer2 match value 0 interrupt enable

13.4.13 TIER3

Address：0x2000a548



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tier3_2	r/w	0	Timer3 match value 2 interrupt enable
1	tier3_1	r/w	0	Timer3 match value 1 interrupt enable
0	tier3_0	r/w	0	Timer3 match value 0 interrupt enable

13.4.14 TPLVR2

Address：0x2000a550

tplvr2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tplvr2

Bits	Name	Type	Reset	Description
31:0	tplvr2	r/w	0	Timer2 Pre-Load Value

13.4.15 TPLVR3

Address：0x2000a554

tplvr3

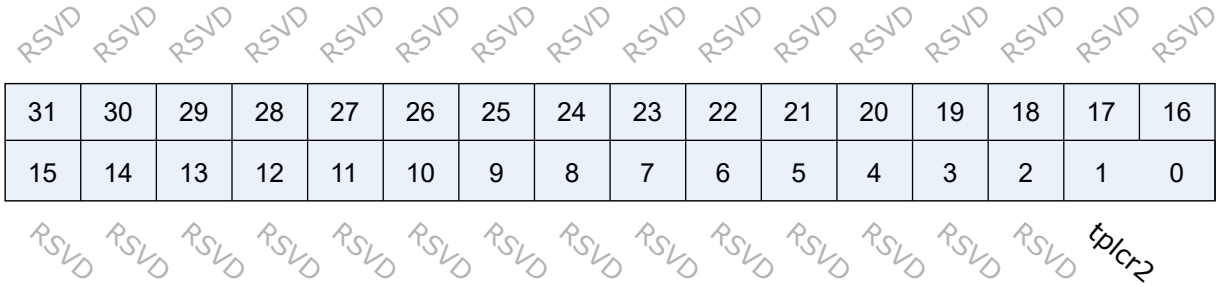
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tplvr3

Bits	Name	Type	Reset	Description
31:0	tplvr3	r/w	0	Timer3 Pre-Load Value

13.4.16 TPLCR2

Address：0x2000a55c

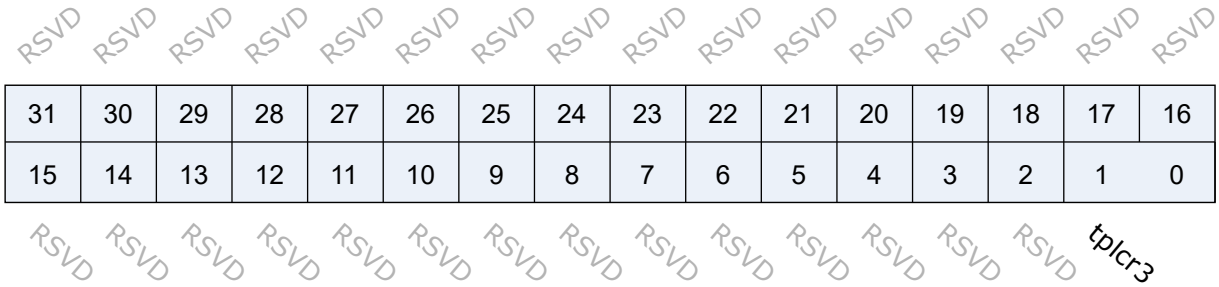


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	tplcr2	

Bits	Name	Type	Reset	Description
31:2	RSVD			
1:0	tplcr2	r/w	0	Timer2 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

13.4.17 TPLCR3

Address：0x2000a560

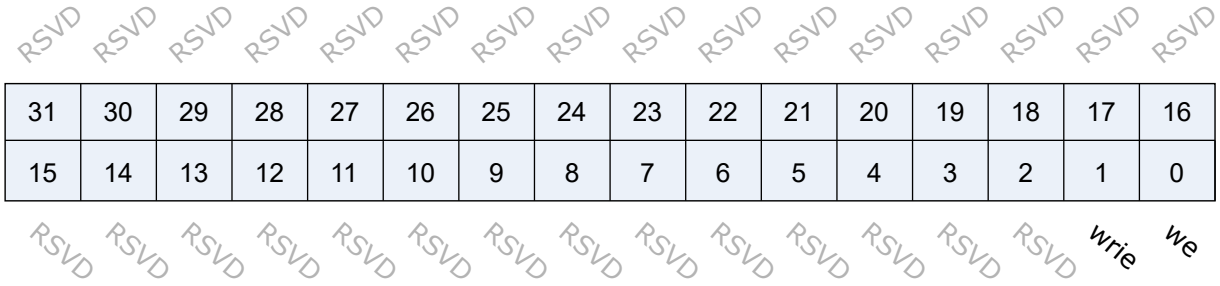


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	tplcr3	

Bits	Name	Type	Reset	Description
31:2	RSVD			
1:0	tplcr3	r/w	0	Timer3 pre-load control 2'd0 - No pre-load 2'd1 - Pre-load with match comparator 0 2'd2 - Pre-load with match comparator 1 2'd3 - Pre-load with match comparator 2

13.4.18 WMER

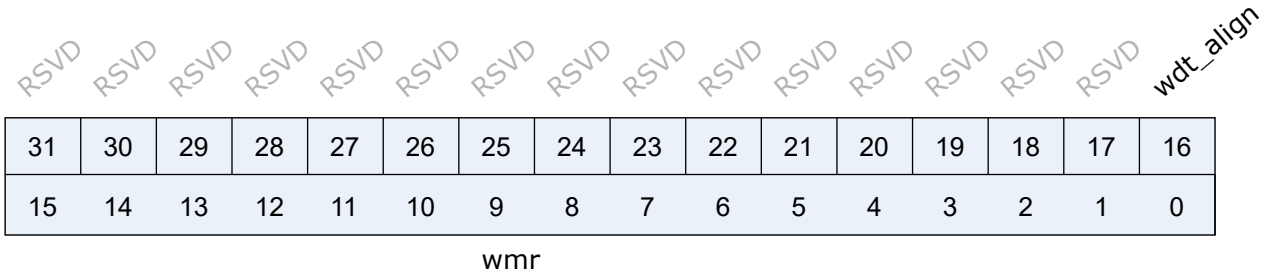
Address：0x2000a564



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	wrie	r/w	0	WDT reset/interrupt mode 1'b0 - WDT expiration to generate interrupt 1'b1 - WDT expiration to generate reset source
0	we	r/w	0	WDT enable register

13.4.19 WMR

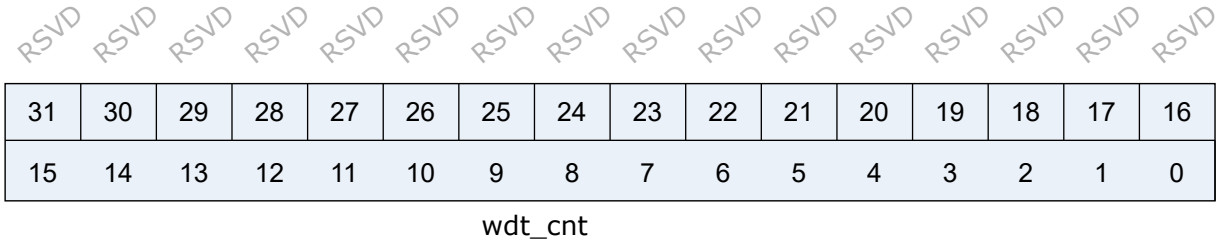
Address：0x2000a568



Bits	Name	Type	Reset	Description
31:17	RSVD			
16	wdt_align	r/w	0	WDT compare value update align interrupt
15:0	wmr	r/w	16'hffff	WDT counter match value

13.4.20 WVR

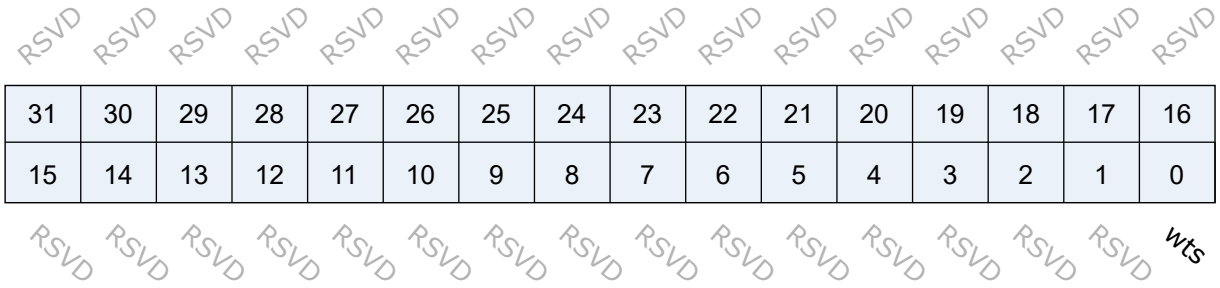
Address：0x2000a56c



Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	wdt_cnt	r	0	WDT counter value

13.4.21 WSR

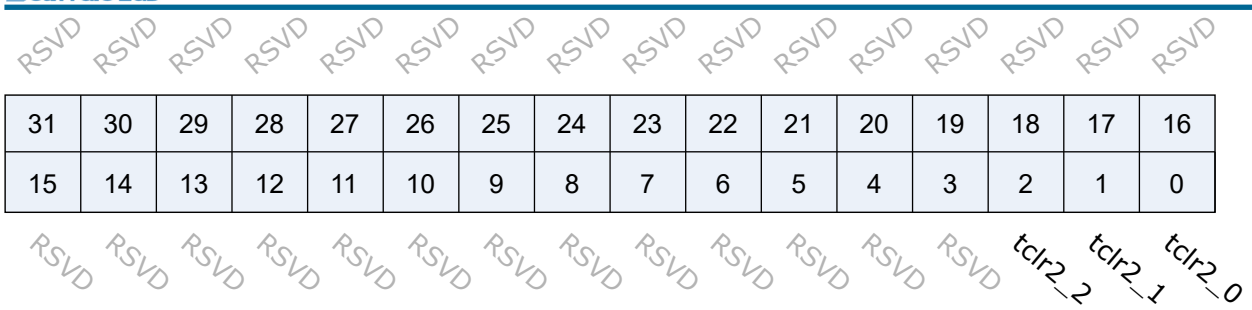
Address：0x2000a570



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	wts	w	0	WDT reset status Write 0 to clear the WDT reset status Read 1 indicates reset was caused by the WDT

13.4.22 TICR2

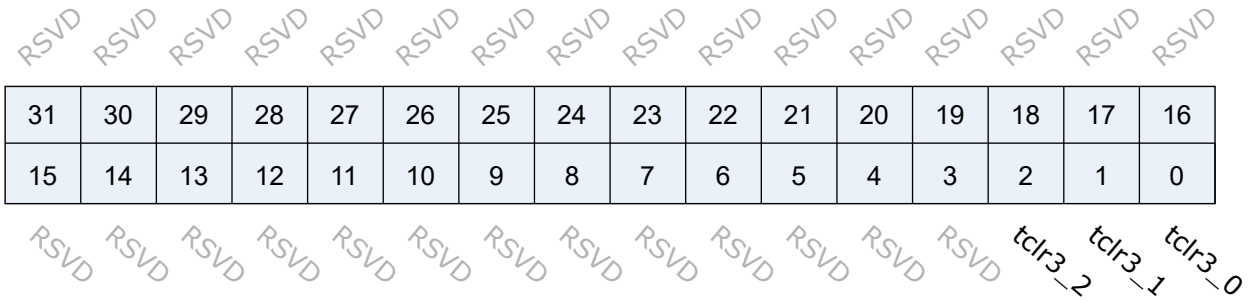
Address：0x2000a578



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tclr2_2	w	0	Timer2 Interrupt clear for match comparator 2
1	tclr2_1	w	0	Timer2 Interrupt clear for match comparator 1
0	tclr2_0	w	0	Timer2 Interrupt clear for match comparator 0

13.4.23 TICR3

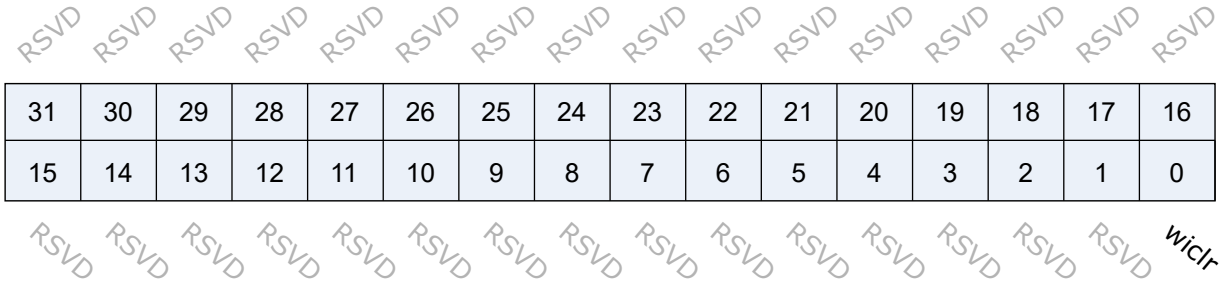
Address：0x2000a57c



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tclr3_2	w	0	Timer3 Interrupt clear for match comparator 2
1	tclr3_1	w	0	Timer3 Interrupt clear for match comparator 1
0	tclr3_0	w	0	Timer3 Interrupt clear for match comparator 0

13.4.24 WICR

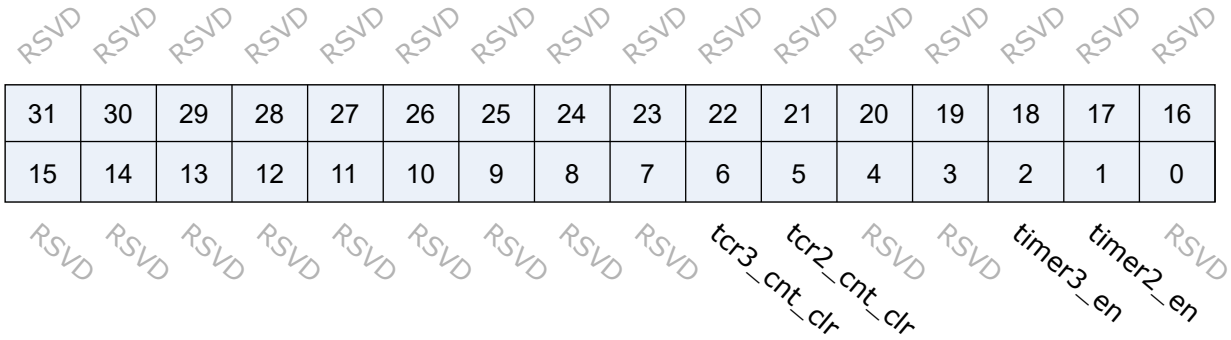
Address：0x2000a580



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	wiclr	w	0	WDT Interrupt Clear

13.4.25 TCER

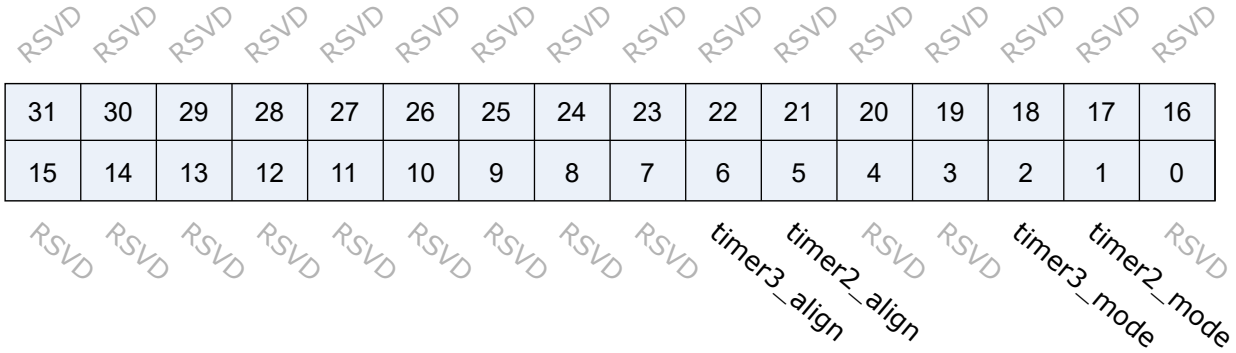
Address：0x2000a584



Bits	Name	Type	Reset	Description
31:7	RSVD			
6	tcr3_cnt_clr	r/w	0	Timer3 count clear
5	tcr2_cnt_clr	r/w	0	Timer2 count clear
4:3	RSVD			
2	timer3_en	r/w	0	Timer3 count enable
1	timer2_en	r/w	0	Timer2 count enable
0	RSVD			

13.4.26 TCMR

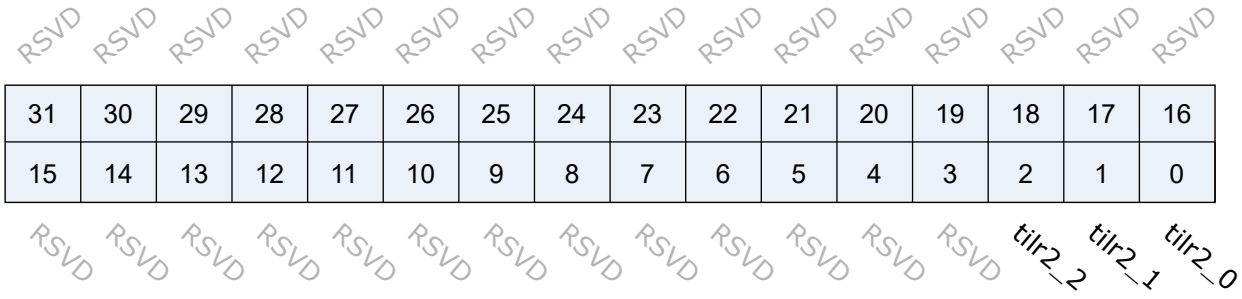
Address：0x2000a588



Bits	Name	Type	Reset	Description
31:7	RSVD			
6	timer3_align	r/w	0	Timer3 compare value update align interrupt
5	timer2_align	r/w	0	Timer2 compare value update align interrupt
4:3	RSVD			
2	timer3_mode	r/w	0	0:pre-load mode 1:free run mode
1	timer2_mode	r/w	0	0:pre-load mode 1:free run mode
0	RSVD			

13.4.27 TILR2

Address：0x2000a590

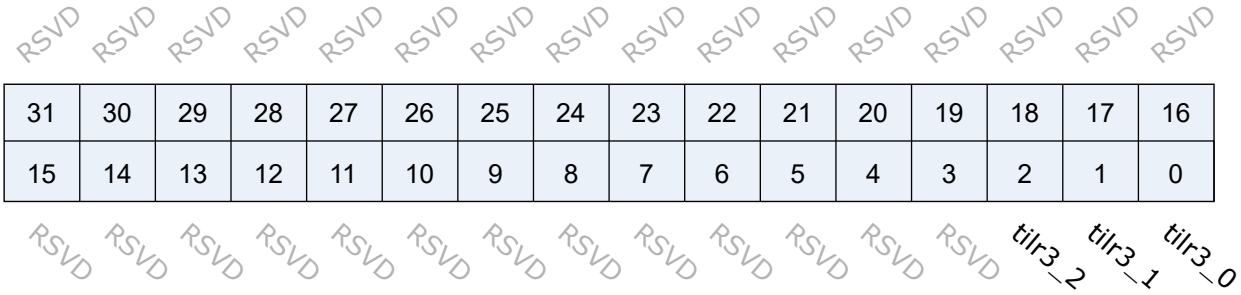


Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tilr2_2	r/w	0	0:level 1:edge
1	tilr2_1	r/w	0	0:level 1:edge

Bits	Name	Type	Reset	Description
0	tilr2_0	r/w	0	0:level 1:edge

13.4.28 TILR3

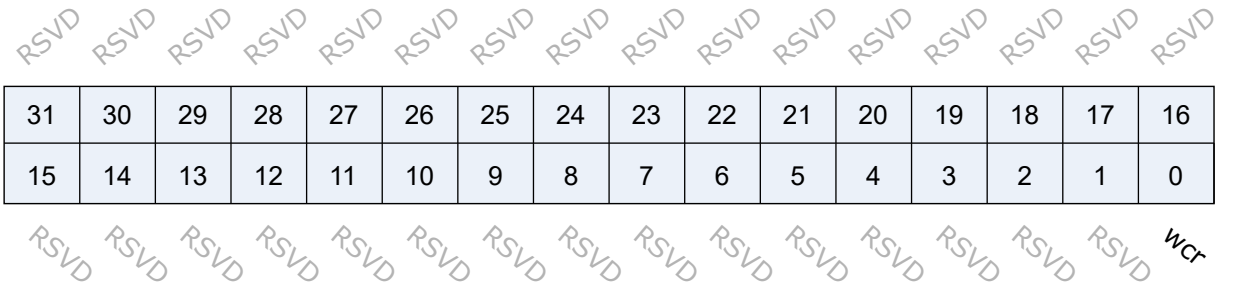
Address：0x2000a594



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	tilr3_2	r/w	0	0:level 1:edge
1	tilr3_1	r/w	0	0:level 1:edge
0	tilr3_0	r/w	0	0:level 1:edge

13.4.29 WCR

Address：0x2000a598



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	wcr	w	0	WDT Counter Reset

13.4.30 WFAR

Address：0x2000a59c

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wfar															

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	wfar	w	0	WDT access key1 - 16'hBABA

13.4.31 WSAR

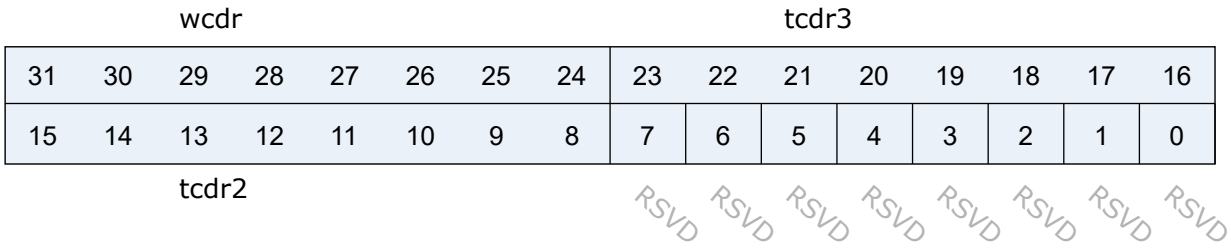
Address：0x2000a5a0

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wsar															

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	wsar	w	0	WDT access key2 - 16'hEB10
-1:32	RSVD			
31:0	tcr2_cnt_lat	r	0	Timer2 Counter Latch Value
-1:32	RSVD			
31:0	tcr3_cnt_lat	r	0	Timer3 Counter Latch Value
-1:32	RSVD			
31:0	tcr2_cnt_sync	r	0	Timer2 Counter Sync Value (continue readable)
-1:32	RSVD			
31:0	tcr3_cnt_sync	r	0	Timer3 Counter Sync Value (continue readable)

13.4.32 TCDR

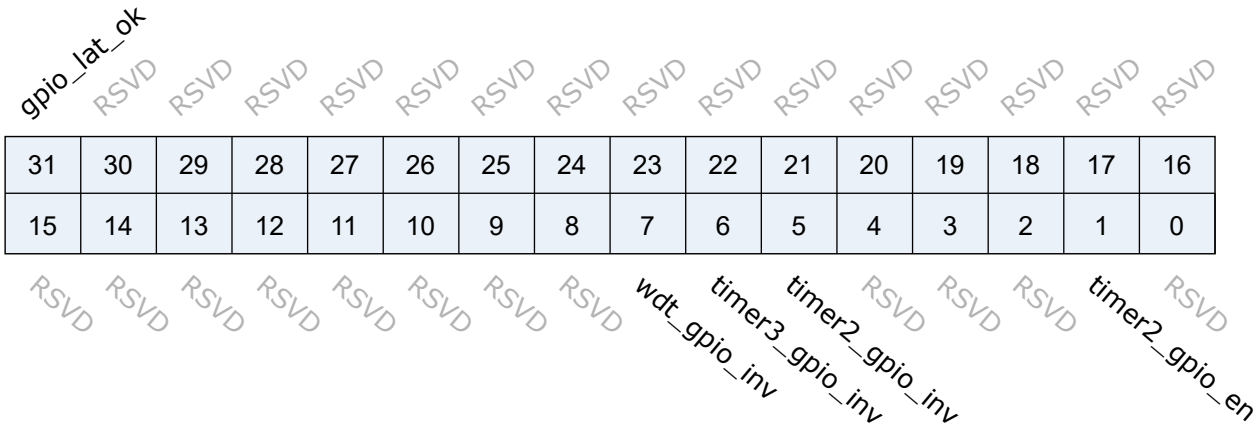
Address：0x2000a5bc



Bits	Name	Type	Reset	Description
31:24	wcdr	r/w	0	WDT clock division value register
23:16	tcdr3	r/w	0	Timer3 clock division value register
15:8	tcdr2	r/w	0	Timer2 clock division value register
7:0	RSVD			

13.4.33 GPIO

Address：0x2000a5c0



Bits	Name	Type	Reset	Description
31	gpio_lat_ok	r	0	Latch Done. Pulse width = (GPIO_LAT2 - GPIO_LAT1) * (Timer2 Cycle)
30:8	RSVD			
7	wdt_gpio_inv	r/w	0	WDT gpio polarity 0:pos 1:neg
6	timer3_gpio_inv	r/w	0	Timer3 gpio polarity 0:pos 1:neg
5	timer2_gpio_inv	r/w	0	Timer2 gpio polarity 0:pos 1:neg

Bits	Name	Type	Reset	Description
4:2	RSVD			
1	timer2_gpio_en	r/w	0	Timer2 gpio measure enable
0	RSVD			

13.4.34 GPIO_LAT1

Address：0x2000a5c4

gpio_lat1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpio_lat1

Bits	Name	Type	Reset	Description
31:0	gpio_lat1	r	0	Pos-Edge Latch Timer2

13.4.35 GPIO_LAT2

Address：0x2000a5c8

gpio_lat2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpio_lat2

Bits	Name	Type	Reset	Description
31:0	gpio_lat2	r	0	Neg-Edge Latch Timer2

14 I2S

14.1 Overview

InterIC Sound (I2S), also Integrated Interchip Sound (IIS), is a digital audio transmission specification defined by Philips in 1986 (revised in 1996) for transmitting digital audio data between internal devices of a system. I2S is a simple digital interface protocol with no address or device selection mechanism and supports full-duplex peer-to-peer transmission. In the I2S bus, the device providing the clock (BCLK and FS) is the master device. In most scenarios, there is a master device and a slave device on the I2S bus for unidirectional or bidirectional data transmission. However, in some special scenarios, only one master can provide the clock to control the data transmission between two slaves, which can precisely control the data flow.

The integrated I2S module of the chip is also compatible with the PCM/TDM format. Compared with I2S, which can only transmit two-channel audio data, the PCM/TDM interface expands the number of transmission channels through Time Division Multiplexing. The main difference between PCM/TDM and I2S lies in the position, length and frame length of the frame clock, and I2S can be considered as a special case of the PCM/TDM interface. However, PCM/TDM does not have a completely unified application standard, and different manufacturers may have slightly different applications.

14.2 Features

- Supports master and slave modes
- Supported I2S protocol
  - Normal I2S (Philips format)
  - Left-Justified
  - Right-Justified
  - The channel slot and effective data width support 8/16/24/32 bits respectively, and the effective data width cannot be greater than the channel width

- Support PCM/TDM format, frame clock position and frame length can be adjusted flexibly, support up to 6 channels, compatible with most common modes
  - Support 2-channel, 3-channel, 4-channel and 6-channel modes, channel slot supports 8/16/24/32 bits
  - PCM/TDM Mode A, after the data is valid at FS, the second rising edge of BCLK is valid
  - PCM/TDM Mode B, after the data is valid at FS, the first rising edge of BCLK is valid
  - Long Frame Sync, long frame sync mode, FS pulse width is equal to the length of 1 Slot
  - Short Frame Sync, short frame sync mode, FS pulse width is equal to 1 BCLK cycle length
- Support 8/11.025/16/22.05/32/44.1/48/96/192 KHz sampling rate, and can achieve finer control by configuring clock source and frequency division coefficient
- Support playback of monaural audio copied to dual-channel mode
- Support merging of two-channel recording data below 8/16bits to improve FIFO usage efficiency
- Support dynamic mute switching function
- Support data MSB/LSB switching
- Support to directly output the clock source, which can be used as MCLK, or output MCLK with the CLK_OUT function of GLB
- Support output signal polarity inversion
- Support DMA transfer mode
- The data transmit/receive FIFO has a width of 32 bits and a depth of 16

14.3 Functional Description

14.3.1 Data formats

The signal timing format of the I2S module is highly flexible and configurable, including the mode of the FS signal, the offset from the FS signal to valid data, the slot width and the valid data width, and is compatible with most common formats. The I2S protocol is a two-channel mode, the FS signal length is equal to a slot, and the effective data length is less than or equal to the slot length. Among them, the valid data in the Normal format is left-aligned and will be offset by one bit relative to the FS signal. Therefore, set the cr_fs_ch_cnt section of the i2s_config register to be dual-channel, set the FS signal length of the cr_fs_1t_mode section to be equal to Solt, set the cr_i2s_mode section to be data left-aligned, and set the cr_ofs_en and cr_ofs_cnt sections to be one-bit data offset, and the I2S protocol Normal format can be obtained. As shown below.

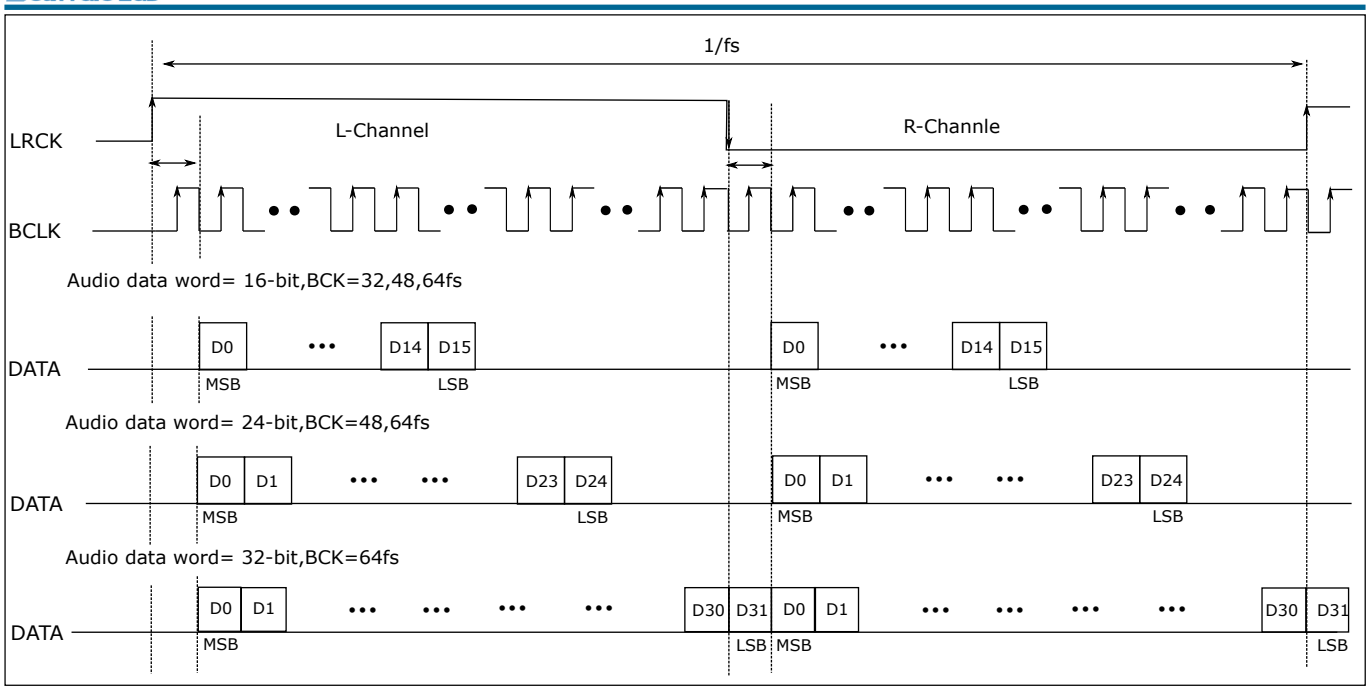


Fig. 14.1: Normal I2S

The left-justified format of the I2S protocol is similar to the Normal format, but there is no data offset. Valid data in the right-aligned format will be aligned to the right within the slot. Therefore, on the basis of the Normal format, set the cr_ofs_en section of the i2s_config register to close the data offset to obtain the Left-Justified format, and then set the cr_i2s_mode section to right-align the data to obtain the Right-Justified format, as shown in the figure below.

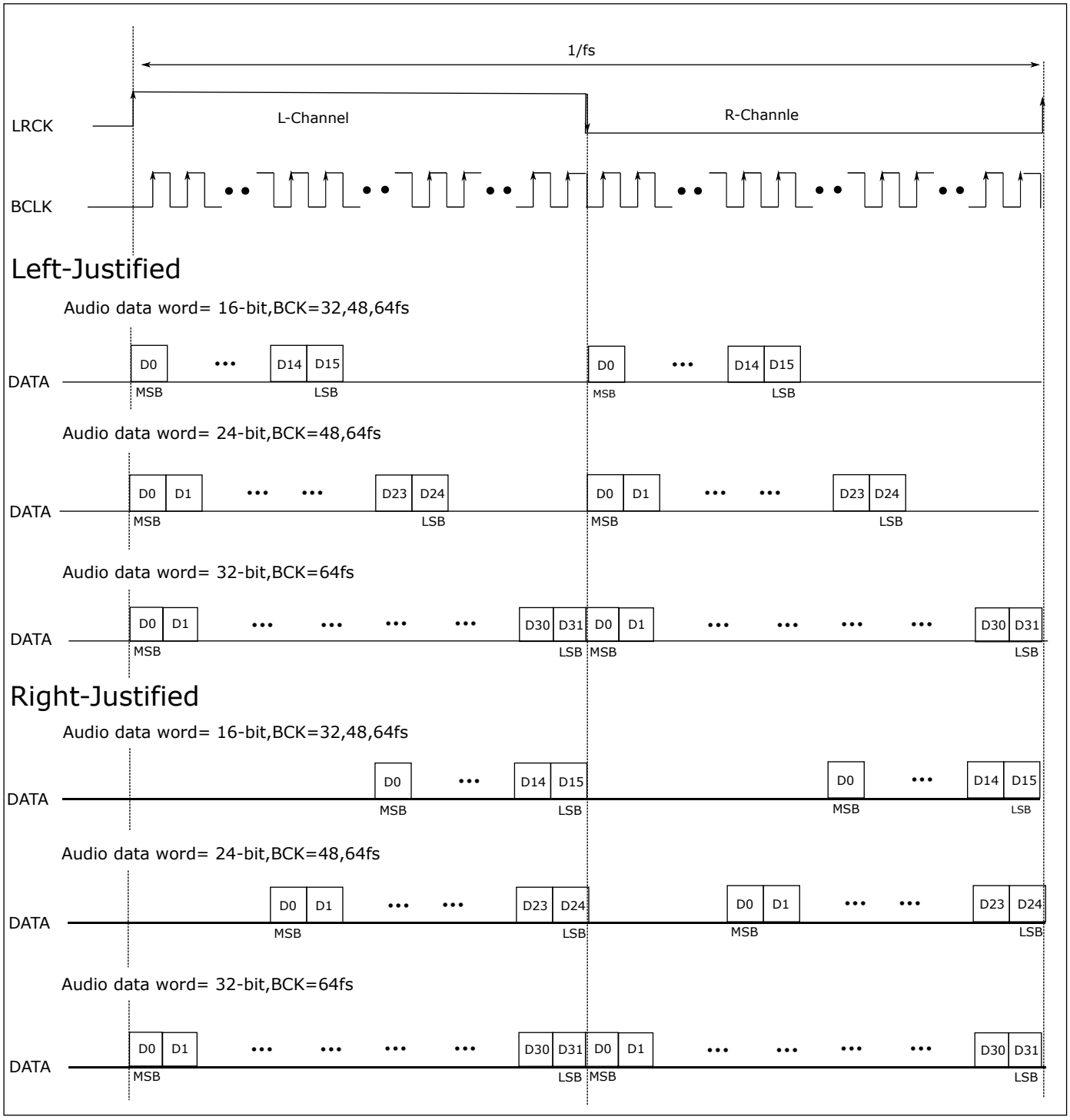


Fig. 14.2: I2S LeftJustified/RightJustified

The configuration of PCM/TDM mode is more flexible. Set the cr_i2s_mode section of the i2s_config register to 2, that is, PCM/TDM mode, set the cr_fs_ch_cnt section to select the number of channels, and set the cr_fs_1t_mode section to select long frame synchronization (FS signal length equal to Slot) or short frame synchronization (FS signal length is one BCLK cycle) mode, set cr_ofs_en section and cr_ofs_cnt section to select data offset, generally define Mode A offset by 1 bit, Mode B has no offset, but the definitions of different manufacturers may be different. The figure

below shows the timing of Mode B in Long/Short Frame Sync, shifting the data to the right by one bit is the timing of Mode A.

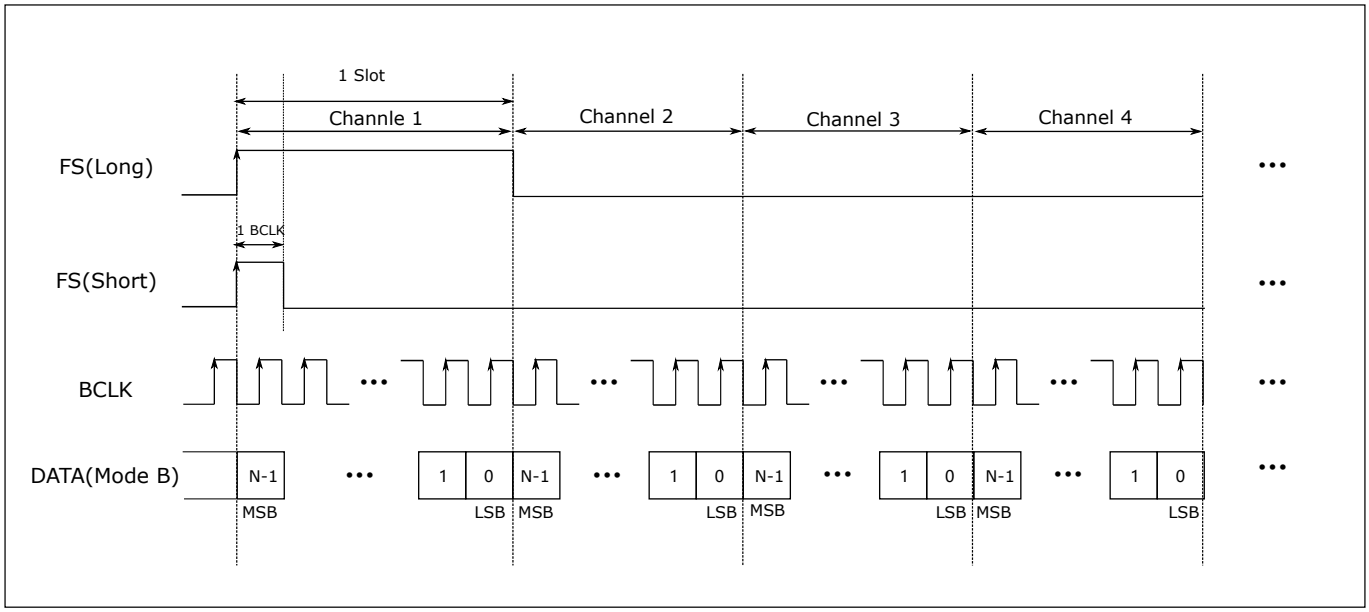


Fig. 14.3: PCM/TDM mode

14.3.2 I2S single channel mode

Usually I2S uses two-channel transmission, but there will be a waste of resources when using a single-channel device. After configuring the cr_mono_mode section of the i2s_config register as Mono mode, the mono mode will be used. It should be noted that both sending and receiving will use mono mode. At this time, the data in the TX FIFO will be placed in the left and right channels at the same time, and the data received by the left channel or the right channel can be selected through the cr_mono_rx_ch segment of the i2s_config register.

14.3.3 Data MSB/LSB justification

Usually I2S uses MSB format to transmit data, but some modes of some manufacturers' devices may use LSB format to transmit, and the format can be adjusted through the cr_endian bit of the i2s_config register.

14.3.4 Binaural data merge and exchange

When using the 8/16bit valid data of the I2S protocol, in order to increase the utilization efficiency of the FIFO, the dual-channel data can be merged through the cr_fifo_lr_merge section of the i2s_fifo_config_0 register. At this time, each data in the FIFO contains both left and right channel data. [0:15] is the left channel at 16bit, [16:31] is the right channel, [0:7] is the left channel at 8bit, and [8:15] is the right channel. At this time, the data width in DMA mode also needs to be adjusted accordingly. In merge mode, the left and right channel output can be exchanged by controlling the cr_fifo_lr_exchg bit of the i2s_fifo_config_0 register.

14.3.5 Silent mode

In some scenarios, it is necessary to pause the playback, but to keep the playback progress continuously changing, the silent mode can meet this requirement. Enter the mute mode through the cr_mute_mode of the i2s_config register. During the mute period, the data transmission speed in the TX FIFO remains unchanged, but the sent data will be forced to remain as the last data before each channel is muted.

14.3.6 Clock source

The clock source of I2S is provided by audio PLL, and the clock divider is used to divide the clock source and then generate the clock signal to drive I2S, as shown below:

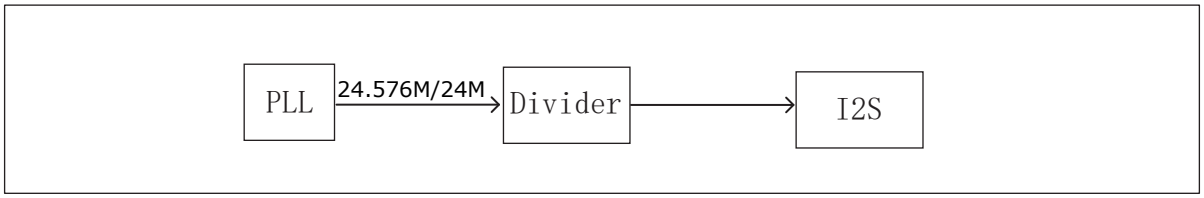


Fig. 14.4: I2S clock

14.3.7 I2S Interrupt

I2S supports the following interrupt control modes:

- TX FIFO request interrupt
  - A TX FIFO request interrupt will be generated when TX_FIFO_CNT in I2S_FIFO_CONFIG_1 is greater than TX_FIFO_TH. When the condition is not met, the interrupt flag will be cleared automatically
- RX FIFO request interrupt
  - A RX FIFO request interrupt will be generated when RX_FIFO_CNT in I2S_FIFO_CONFIG_1 is greater than RX_FIFO_TH. When the condition is not met, the interrupt flag will be cleared automatically
- Overrun/undderun error interrupt
  - If the TX/RX FIFO overflows or underflows, it will trigger the error interrupt. When the error disappears, the flag bit will be cleared automatically

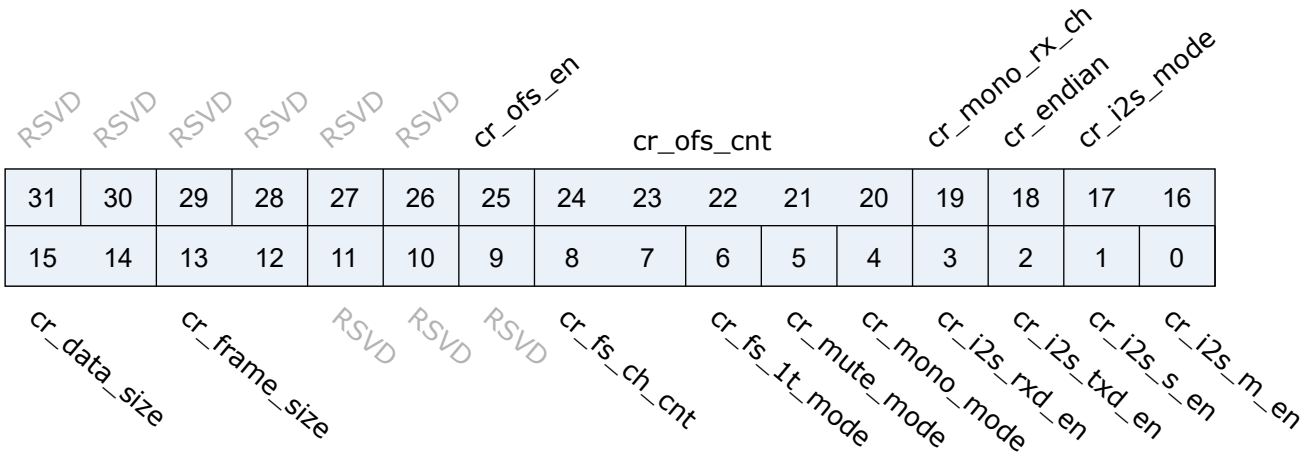
All the interrupt enable bits and interrupt flag bits of I2S are in the I2S_INT_STS register.

14.4 Register description

Name	Description
i2s_config	I2S control
i2s_int_sts	Interrupt control
i2s_bclk_config	Frequency division control
i2s_fifo_config_0	FIFO control 0
i2s_fifo_config_1	FIFO control 1
i2s_fifo_wdata	TX FIFO
i2s_fifo_rdata	RX FIFO
i2s_io_config	IO control

14.4.1 i2s_config

Address：0x2000ab00

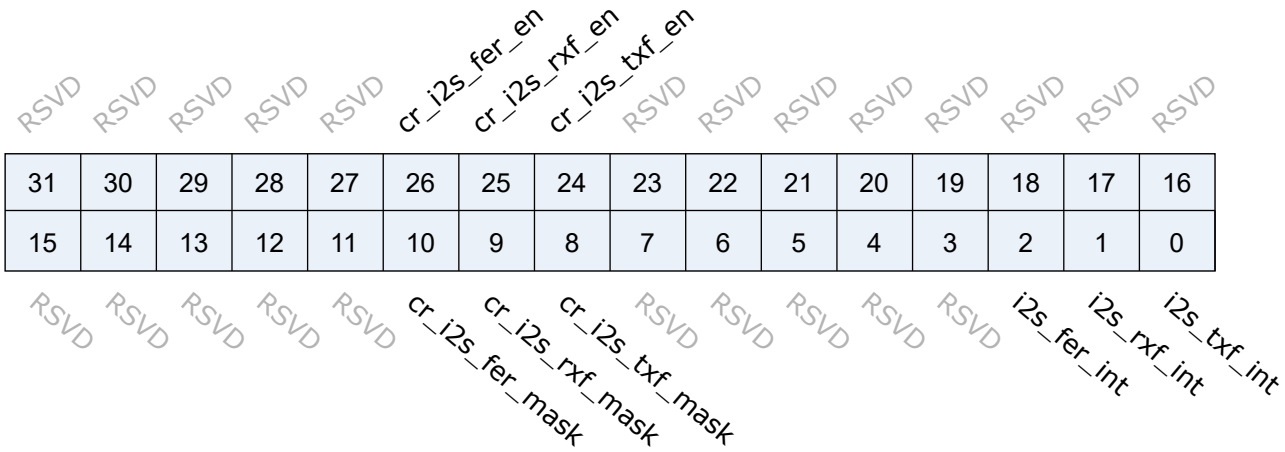


Bits	Name	Type	Reset	Description
31:26	RSVD			
25	cr_ofs_en	r/w	1'b0	Offset enable 1'b0: Disabled, 1'b1: Enabled
24:20	cr_ofs_cnt	r/w	5'd0	Offset cycle count (unit: cycle of I2S BCLK) 5'd0: 1 cycle 5'd1: 2 cycles ...

Bits	Name	Type	Reset	Description
19	cr_mono_rx_ch	r/w	1'b0	RX mono mode channel select signal 1'b0: L-channel 1'b1: R-channel
18	cr_endian	r/w	1'b0	Data endian (bit reverse) 1'b0: MSB goes out first, 1'b1: LSB goes out first
17:16	cr_i2s_mode	r/w	2'd0	2'd0: Left-Justified, 2'd1: Right-Justified, 2'd2: DSP, 2'd3: Reserved
15:14	cr_data_size	r/w	2'd1	Data bit width of each channel 2'd0: 8, 2'd1: 16, 2'd2: 24, 2'd3: 32 (bits)
13:12	cr_frame_size	r/w	2'd1	Frame size of each channel 2'd0: 8, 2'd1: 16, 2'd2: 24, 2'd3: 32 (cycles)
11:9	RSVD			
8:7	cr_fs_ch_cnt	r/w	2'd0	Channel count of each frame 2'd0: FS 2-channel mode 2'd1: FS 3-channel mode (DSP mode only) 2'd2: FS 4-channel mode (DSP mode only) 2'd3: FS 6-channel mode (DSP mode only) Note: cr_mono_mode & cr_fifo_lr_merge will be invalid in 3-channel mode Note: frame_size must equal data_size in 3/4/6-channel mode
6	cr_fs_1t_mode	r/w	1'b0	1'b0: FS high/low is even, 1'b1: FS only asserts for 1 cycle
5	cr_mute_mode	r/w	1'b0	1'b0: Normal mode, 1'b1: Mute mode
4	cr_mono_mode	r/w	1'b0	1'b0: Stereo mode, 1'b1: Mono mode Note: csr_mono_mode & csr_fifo_lr_merge should NOT be enabled at the same time
3	cr_i2s_rxd_en	r/w	1'b0	Enable signal of I2S RXD signal
2	cr_i2s_txd_en	r/w	1'b0	Enable signal of I2S TXD signal
1	cr_i2s_s_en	r/w	1'b0	Enable signal of I2S Slave function, cannot enable both csr_i2s_m_en & csr_i2s_s_en
0	cr_i2s_m_en	r/w	1'b0	Enable signal of I2S Master function, cannot enable both csr_i2s_m_en & csr_i2s_s_en

14.4.2 i2s_int_sts

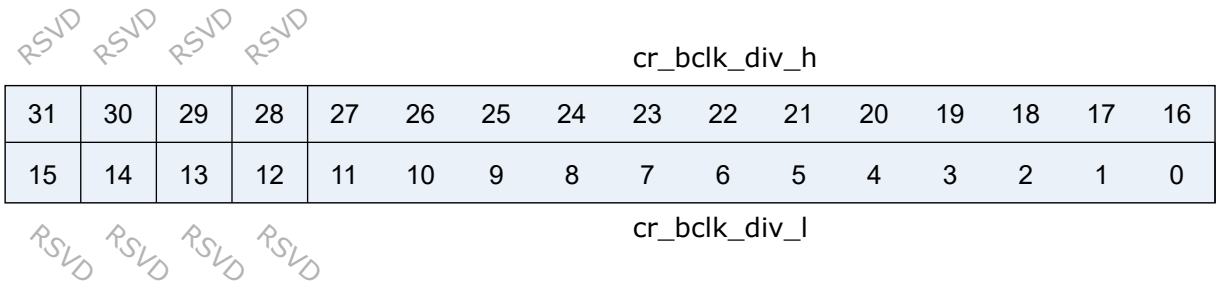
Address：0x2000ab04



Bits	Name	Type	Reset	Description
31:27	RSVD			
26	cr_i2s_fer_en	r/w	1'b1	Interrupt enable of i2s_fer_int
25	cr_i2s_rxf_en	r/w	1'b1	Interrupt enable of i2s_rxf_int
24	cr_i2s_txf_en	r/w	1'b1	Interrupt enable of i2s_txf_int
23:11	RSVD			
10	cr_i2s_fer_mask	r/w	1'b1	Interrupt mask of i2s_fer_int
9	cr_i2s_rxf_mask	r/w	1'b1	Interrupt mask of i2s_rxf_int
8	cr_i2s_txf_mask	r/w	1'b1	Interrupt mask of i2s_txf_int
7:3	RSVD			
2	i2s_fer_int	r	1'b0	I2S TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	i2s_rxf_int	r	1'b0	I2S RX FIFO ready (rx_fifo_cnt > rx_fifo_th) interrupt, auto-cleared when data is popped
0	i2s_txf_int	r	1'b1	I2S TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed

14.4.3 i2s_bclk_config

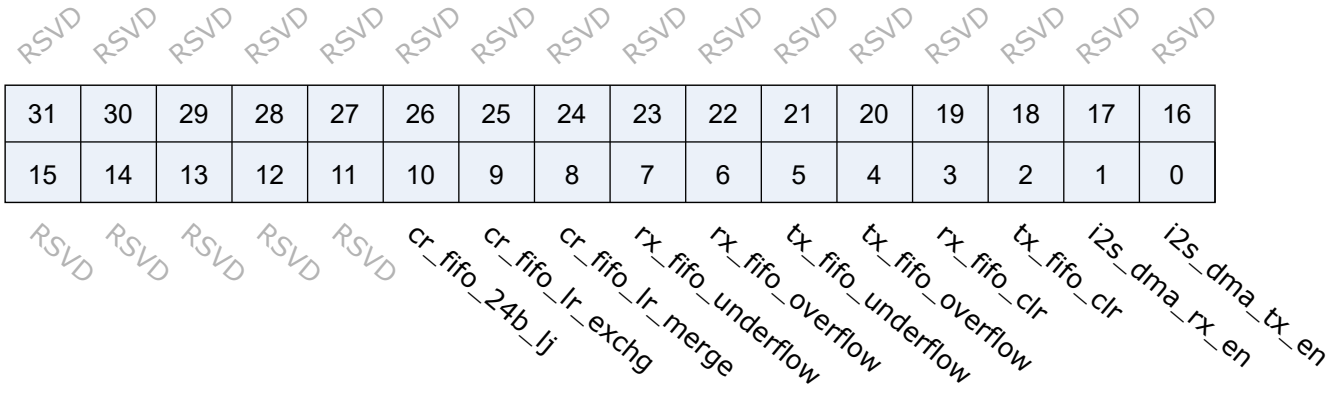
Address：0x2000ab10



Bits	Name	Type	Reset	Description
31:28	RSVD			
27:16	cr_bclk_div_h	r/w	12'd1	I2S BCLK active high period (unit: cycle of i2s_clk)
15:12	RSVD			
11:0	cr_bclk_div_l	r/w	12'd1	I2S BCLK active low period (unit: cycle of i2s_clk)

14.4.4 i2s_fifo_config_0

Address：0x2000ab80

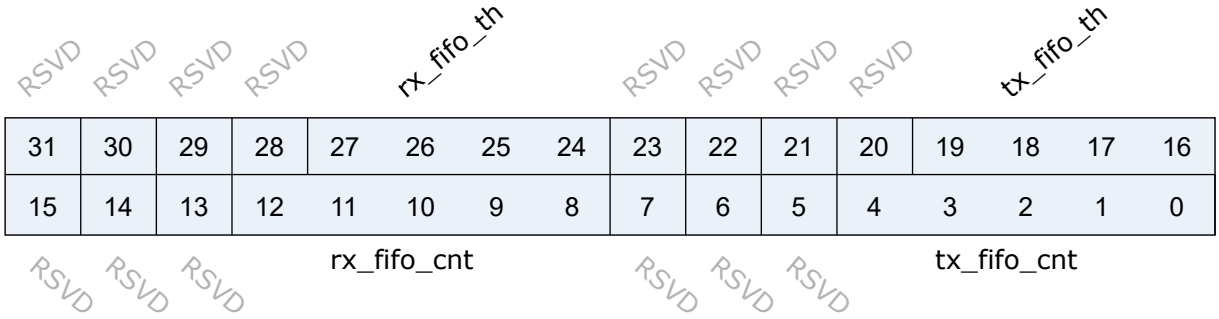


Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_fifo_24b_lj	r/w	1'b0	FIFO 24-bit data left-justified mode 1'b0: Right-justified, 8'h0, data[23:0] 1'b1: Left-justified, data[23:0], 8'h0 Note: Valid only when cr_data_size = 2'd2 (24-bit)

Bits	Name	Type	Reset	Description
9	cr_fifo_lr_exchg	r/w	1'b0	The position of L/R channel data within each entry is exchanged if this bit is enabled Can only be enabled if data size is 8 or 16 bits and csr_fifo_lr_merge is enabled
8	cr_fifo_lr_merge	r/w	1'b0	Each FIFO entry contains both L/R channel data if this bit is enabled Can only be enabled if data size is 8 or 16 bits Note: cr_fifo_lr_merge &cr_mono_mode should NOT be enabled at the same time Note: cr_fifo_lr_merge &cr_fifo_l_shift should NOT be enabled at the same time
7	rx_fifo_underflow	r	1'b0	Underflow flag of RX FIFO, can be cleared by rx_fifo_clr
6	rx_fifo_overflow	r	1'b0	Overflow flag of RX FIFO, can be cleared by rx_fifo_clr
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	rx_fifo_clr	w1c	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	i2s_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	i2s_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

14.4.5 i2s_fifo_config_1

Address：0x2000ab84



Bits	Name	Type	Reset	Description
31:28	RSVD			
27:24	rx_fifo_th	r/w	4'd0	RX FIFO threshold, dma_rx_req will not be asserted if tx_fifo_cnt is less than this value

Bits	Name	Type	Reset	Description
23:20	RSVD			
19:16	tx_fifo_th	r/w	4'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_-fifo_cnt is less than this value
15:13	RSVD			
12:8	rx_fifo_cnt	r	5'd0	RX FIFO available count
7:5	RSVD			
4:0	tx_fifo_cnt	r	5'd16	TX FIFO available count

14.4.6 i2s_fifo_wdata

Address：0x2000ab88

i2s_fifo_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2s_fifo_wdata

Bits	Name	Type	Reset	Description
31:0	i2s_fifo_wdata	w	x	TX FIFO write data port

14.4.7 i2s_fifo_rdata

Address：0x2000ab8c

i2s_fifo_rdata

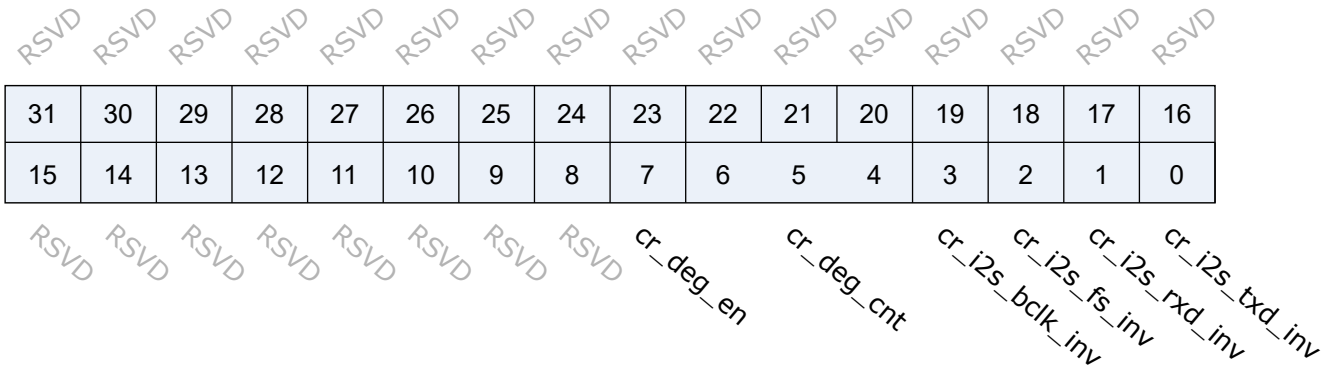
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

i2s_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	i2s_fifo_rdata	r	32'h0	RX FIFO read data port

14.4.8 i2s_io_config

Address：0x2000abfc



Bits	Name	Type	Reset	Description
31:8	RSVD			
7	cr_deg_en	r/w	1'b0	Deglitch enable (for all th input pins) 1'b0: Disabled, 1'b1: Enabled
6:4	cr_deg_cnt	r/w	3'd0	Deglitch cycle count (unit: cycle of I2S kernel clock) 3'd0: 1 cycle 3'd1: 2 cycles ...
3	cr_i2s_bclk_inv	r/w	1'b0	Inverse BCLK signal 0: No inverse, 1: Inverse
2	cr_i2s_fs_inv	r/w	1'b0	Inverse FS signal 0: No inverse, 1: Inverse
1	cr_i2s_rxd_inv	r/w	1'b0	Inverse RXD signal 0: No inverse, 1: Inverse
0	cr_i2s_txd_inv	r/w	1'b0	Inverse TXD signal 0: No inverse, 1: Inverse

15 AudioDAC

15.1 Overview

A PWM modulation module is built in the chip to output analog signals to drive the speaker, so as to realize audio playback.

15.2 Features

- One 16-bit DAC PWM is integrated, supporting one analog PWM differential output * Sampling rate: 8k－48k * Signal to noise ratio (AW): 95 dB gain * Harmonic distortion + noise: -70dB @ 0dB gain
- The GPDAC module can be used as an audio analog signal output, converting the processed audio data directly from the dual-channel GPDAC module into an analog differential signal output.
  - Independent digital volume control and support for slope adjustable fade volume adjustment and fade mute
  - Mixer support for dual-channel audio data input and the ability to select one of the mono outputs or mix it into one channel for output
  - 32-bit width FIFO, depth 32, support 32bit/24bit/20bit/16bit four data formats, support mono data source and two-channel mixed data source
  - Support DMA transfer mode

15.3 Clock Tree

Users need to first configure the Audio PLL output clock to 491.52M or 451.58M, corresponding to the 48K series and 44.1K series sample rates, respectively, and select the 5-division frequency of the Audio PLL output as the AudioDAC clock source.

Within the AudioDAC module, the crossover coefficient of each sub-module is automatically configured according to the sample rate setting, so there is no need to set the crossover coefficient manually. The clock dividing frequency is shown as follows.

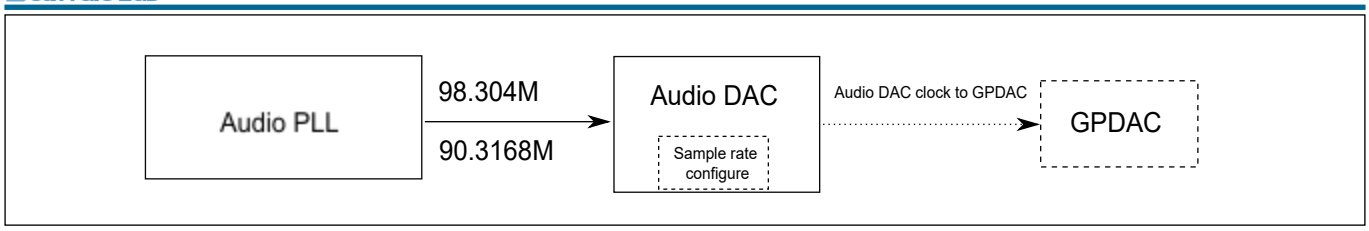


Fig. 15.1: Block Diagram of Clock

15.4 Functional Description

The block diagram of AudioDAC is shown as follows.

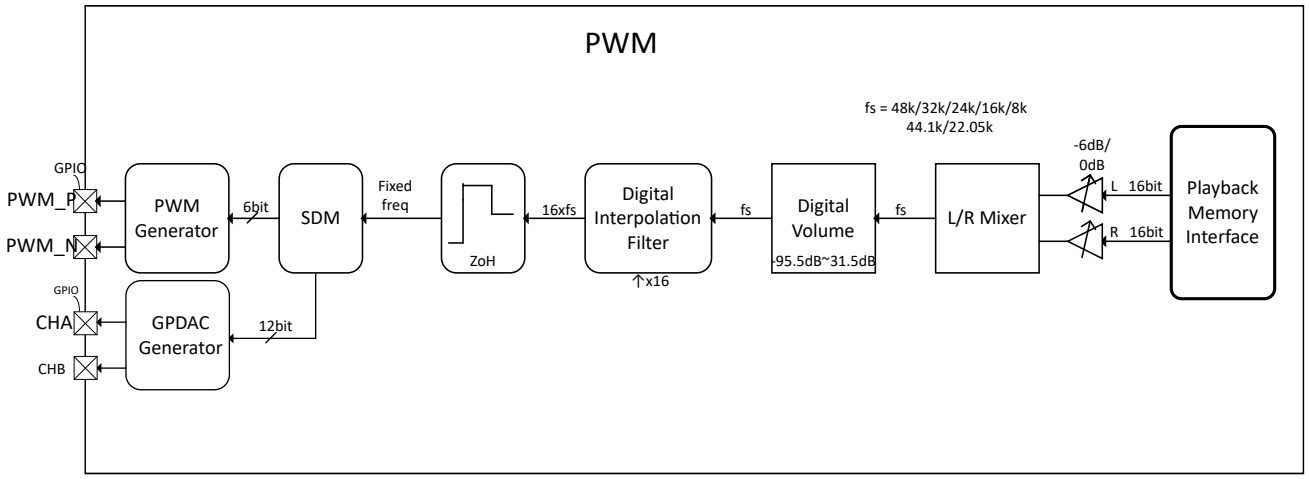


Fig. 15.2: Block Diagram of Module

AudioDAC will process the data in TX FIFO through Mixer first, then perform digital volume adjustment, interpolation filtering and other processing, DSM modulation, and according to the set output mode, generate data in the corresponding format to PWM modulator or GPDAC module. The GPDAC module can output analog signal directly, while the PWM modulator output signal needs to be converted to analog signal by external RC filter circuit.

15.4.1 AudioDAC Interrupt

AudioDAC supports the following interrupt control modes:

- TX FIFO request interrupt
  - A TX FIFO request interrupt is generated when TX_DRQ_CNT in TX_FIFO_CTRL is greater than TX_TRG_LEVEL. When the condition is not met, the interrupt flag is cleared automatically.
- TX FIFO underrun interrupt
  - When there is no data in TX FIFO, but the user enables TX FIFO state machine through TX_CH_EN in TX_FIFO_CTRL, the TX FIFO underrun interrupt is generated.

- TX FIFO overrun interrupt
  - When the user fills in data that exceeds the maximum depth of TX FIFO, it leads to TX FIFO overflow and cause a TX FIFO overrun interrupt.
- Volume adjustment complete interrupt
  - Audio DAC volume control supports the fade-in/fade-out function. When volume adjustment is completed, the "volume adjustment complete interrupt" is generated, indicating that fade-in/fade-out adjustment is completed.

15.4.2 FIFO Format Control

AUPWM_TX_FIFO_CTRL can control the format of audio data stored in FIFO and set the number of channels of audio data source. The FIFO will intercept the valid 16bit data according to the data format and channel number setting, and use it as the input of the corresponding channel of the next Mixer level.

The FIFO controller supports the following four data storage formats, which are determined by FIFO_CTRL[25:24].

- Mode 0:

  32bit mode, DATA[15:0] = {FIFO[31:16]}, the highest bit of valid data is at 31 bits

- Mode 1:

  24bit mode, DATA[15:0] = {FIFO[23:8]}}, the highest bit of valid data is 23 bits

- Mode 2:

  20bit mode, DATA[15:0] = {FIFO[19:4]}}, the highest bit of valid data is 19 bits

- Mode 3:

  16bit mode, DATA[15:0] = {FIFO[15:0]}, the highest bit of valid data is 15 bits

In most cases, Mode3 is selected when the audio source data is 16bit wide, and the maximum resolution of the Audio DAC is 16bits. Other modes exist because if the size of the audio data to be played is 32bit, where the effective data width is 32/24/20 bits, the user does not need to convert the data to 16bit mode in the software, and the FIFO automatically converts it for efficiency.

The number of data source channels of Audio DAC is controlled by the tx_ch_en segment of audac_fifo_ctrl register, the corresponding effect is as follows:

- 0: the left and right channels are closed, the data in the FIFO will not be transferred to the Mixer and the Audio DAC stops playing
- 1: In mono mode, all data in the FIFO is used as left channel data and will be input to the Mixer’s left channel one by one.
- 2: Mono mode, where all data in the FIFO is used as right channel data and is input to the Mixer’s right channel one by one

- 3: Dual mode, where the odd numbered data in the FIFO is used as the left channel data and the even numbered data is used as the right channel data, which are input to the left and right channels of the Mixer respectively

Because the Audio DAC only supports one channel output, mono source data should be selected in most cases. However, when the source data is already dual-channel and cannot be changed, you can use the Mixer to select one of the channels or mix it for playback, as described in the Mixer later in this article.

15.4.3 Startup of FIFO and DMA Transfer

The TX FIFO data of Audio DAC can be carried via DMA. To enable DMA mode, you need to set audac_0 register's dac_itf_en bit DMA interface enable to 1, and audac_fifo_ctrl register's tx_drq_en bit DMA request enable to position 1.

The FIFO data volume threshold for triggering DMA requests is selected by configuring the tx_drq_cnt segment of the audac_fifo_ctrl register with four options, 8/16/32, and the same interrupt threshold as configured in the tx_trg_level segment.

The user can get the number of the current number of free FIFOs in real time through txa_cnt of audac_fifo_status register.

When the value of the number of free in the FIFO is greater than the set threshold, a DMA carry will be triggered.

Note that if the remaining number of free in the TX FIFO is less than the number of DMA moves in at one time, an overrun error is triggered. If the DMA does not move in the data in time when the Audio ADC is working, an underrun error is also triggered, so pay attention to the configuration of the FIFO threshold and DMA brust.

15.4.4 Configurable Mixer Channel Mixer

The Audio DAC supports playing only one channel of audio playback. If the source data is a cross-mix of two channels, the Mixer can be used to select one of the audio channels for playback or to mix the two channels for playback. Controlled by the dac_mix_sel segment of the audac_1 register, it supports the following four modes:

- Selecting only the left channel for playback (odd times of data)
- Selecting only the right channel for playback (even number of times of data)
- Playback by summing the left and right channel values
- Playback by taking the average of the left and right channel values

Note that the channel configuration of Mixer mixer must match each other with the channel configuration of FIFO, otherwise Audio DAC will stop working. The detailed process is shown in the following figure.

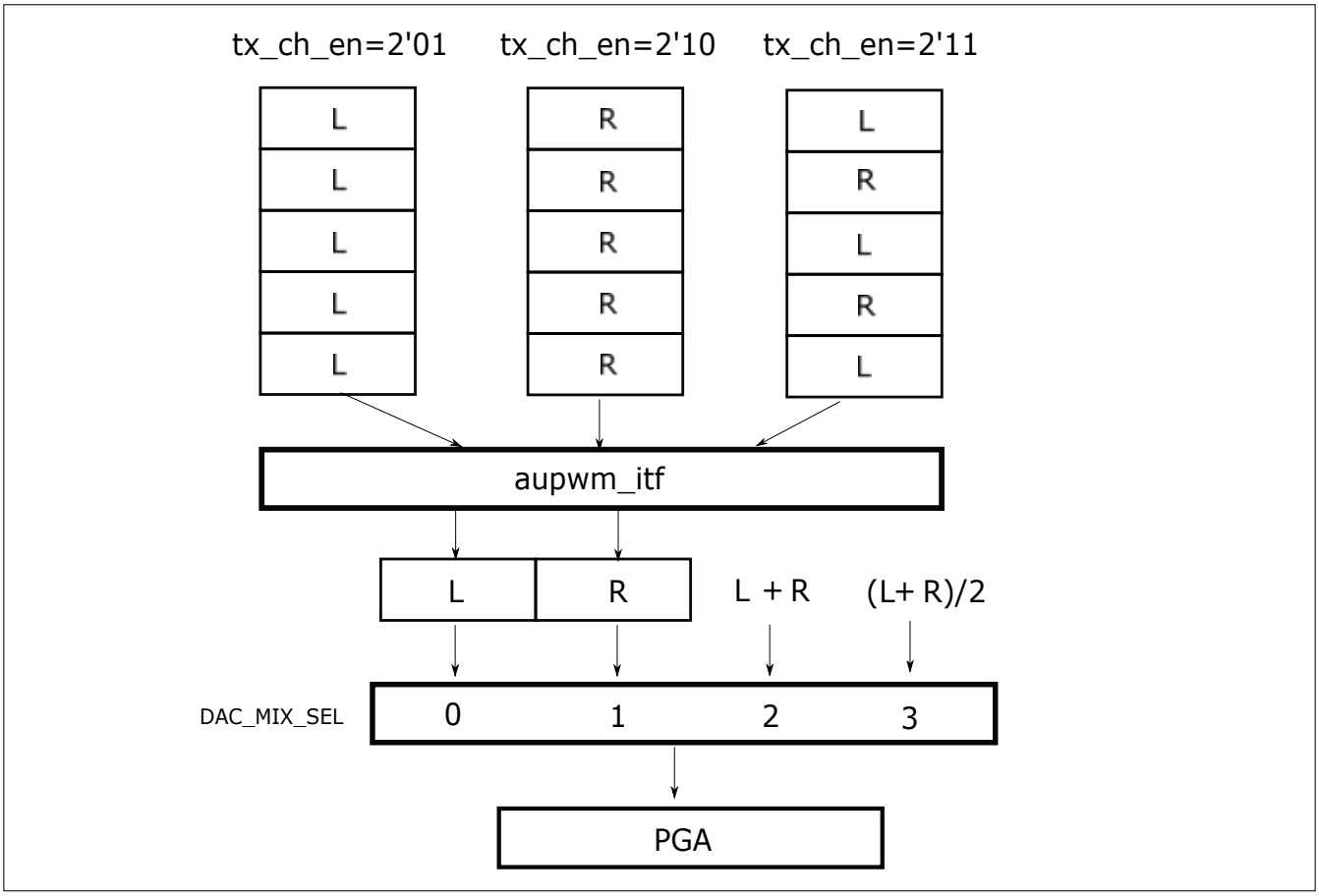


Fig. 15.3: Mixer

When the playback source is configured as stereo, the data moved sequentially by DMA will be filled into the left and right channels sequentially, which requires that the data source is also stored in L-R-L-R cross-ordering. At this point, you can use the Mixer selector to choose which channel is needed or to sum or average the two channels into the next level of modulation circuit.

Note the tx_ch_en and Mixer selection. If you put the mono data into the left channel by tc_ch_en, but the Mixer selects the right channel, it will cause an error and cannot be played.

15.4.5 Volume Control

The user can configure the volume through the dac_s0_volume register of audac_s0, the gain range is -95.5dB-18dB, and the step unit is 0.5dB. When dac_s0_volume_update register is 1, the volume will be updated to take effect.

User can select the volume adjustment mode by dac_s0_mute_softmode, dac_s0_ctrl_mode, there are 3 adjustment modes:

- 0: Direct change, the new volume will be changed from the original volume to the new volume configuration immediately after the new volume takes effect.
- 1: Over zero fade mode: After the new volume takes effect, it will be changed gradually and linearly from the

original volume to the new volume, and will only change the volume controversy when the audio data is 0.

- 2: Fade mode, the new volume will gradually change linearly from the original volume to the new volume after the new volume takes effect.

The slope of the over-zero fade is controlled by dac_s0_ctrl_zcd_rate and the slope of the fade mode is controlled by dac_s0_ctrl_rmp_rate and can be configured to change 0.5dB every 2 to 2048 samples, with the sample points calculated to the 2^(n+1) power. When the volume tries to change during a fade but no zero data is encountered for a long time, a step of 0.5dB will be forcibly updated when the value set by dac_s0_ctrl_zcd_timeout is exceeded.

It also supports setting mute directly without changing the volume, and mute also supports fade mode. The fade slope of mute and the fade slope of mute release are controlled by dac_s0_mute_rmpdn_rate and dac_s0_mute_rmpup_-rate respectively.

15.4.6 ZeroDetect

AudioDAC provides ZeroDetect function. When this function is turned on (zd_en position 1 of audac_zd_0 register), if the Mixer mixer output keeps zero data and the amount exceeds the threshold (zd_time segment of audac_zd_0 register), it will turn off the DSM modulator and keep the output at zero. The DSM modulator will be turned off and the output will be kept at 0. The purpose of this is to reduce the bottom noise caused by broadcasting zero data with muting.

15.5 Configuration Process

1. According to the audio sample rate to be broadcast, select the corresponding sample rate.
2. according to the actual demand, configure to PWM output mode or GPDAC output mode, GPDAC mode requires additional configuration of the DAC module, see the DAC description document for details.
3. according to the number of audio channels to be broadcast, configure the FIFO and Mixer channel mixer.
4. Configure DMA to carry data to the AudioDAC TX FIFO.
5. Enable the channel via tx_ch_en of audac_fifo_ctrl to start playback.
6. Adjust the volume during playback (optional).

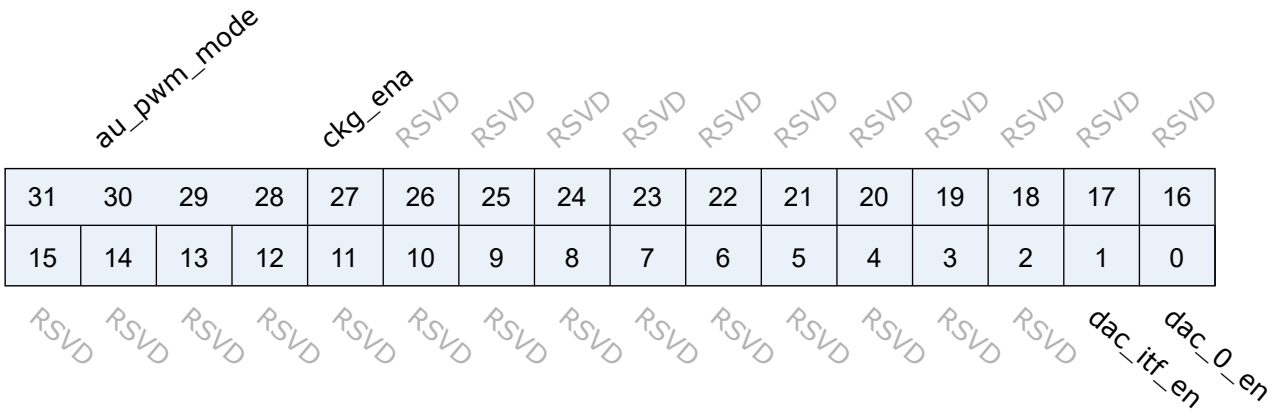
15.6 Register description

Name	Description
audac_0	Clock control register
audac_status	Status register
audac_s0	Volume control register 1
audac_s0_misc	Volume control register 2

Name	Description
audac_zd_0	zero detect control register
audac_1	
audac_fifo_ctrl	fifo control register
audac_fifo_status	fifo status register
audac_fifo_data	fifo data register

15.6.1 audac_0

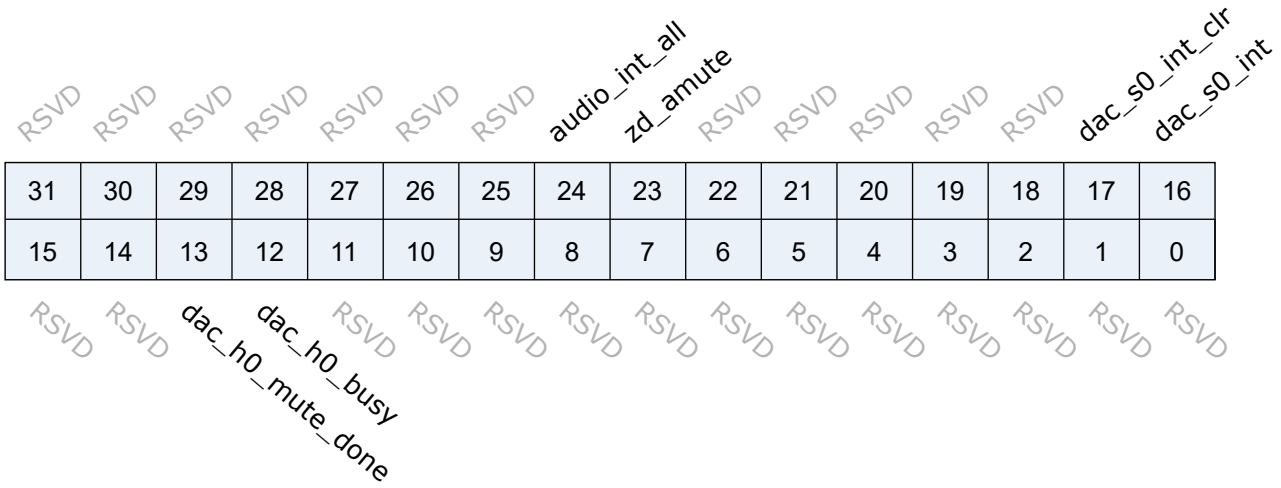
Address：0x20055000



Bits	Name	Type	Reset	Description
31:28	au_pwm_mode	r/w	4'd0	pwm output mode,rang 0 6: 0:8KHz, 1:16KHz, 2:32KHz, 3:24KHz, 4:48KHz, 5:22.05KHz, 6:44.1KHz gpdac output mode,rang 9 14: 9:16KHz, 10:32KHz, 11:24KHz, 12:48KHz, 13:22.05KHz, 14:44.1KHz,
27	ckg_ena	r/w	1	enabl eclock gen
26:2	RSVD			
1	dac_itf_en	r/w	0	enable dac to audio dma interface
0	dac_0_en	r/w	0	enable dac ch0

15.6.2 audac_status

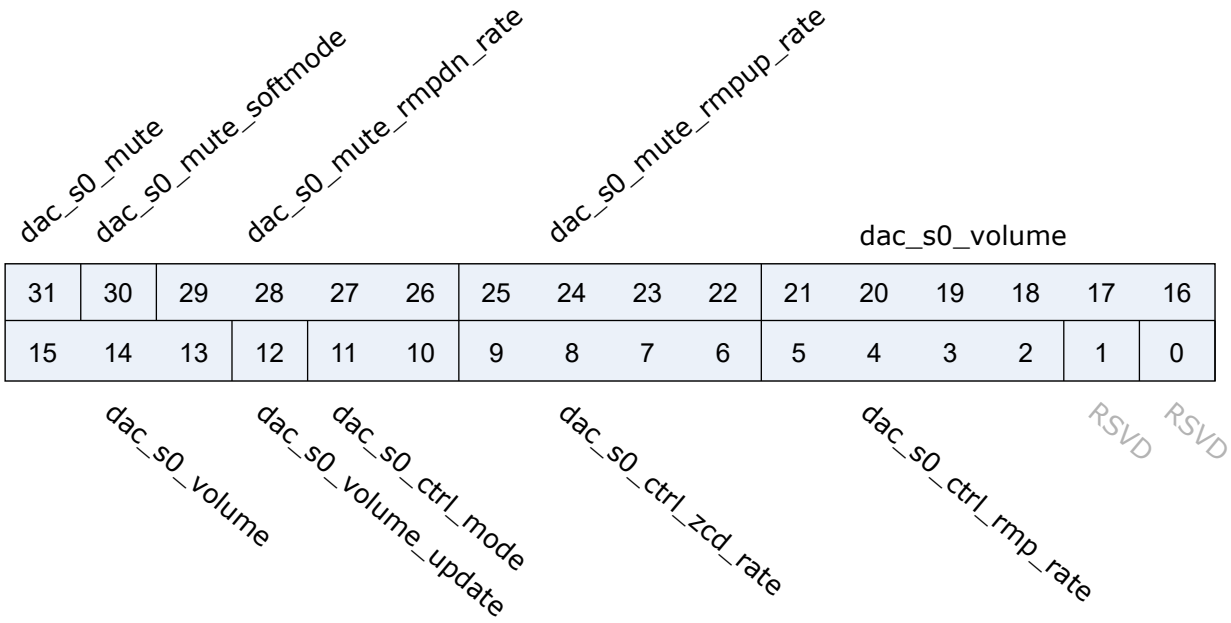
Address：0x20055004



Bits	Name	Type	Reset	Description
31:25	RSVD			
24	audio_int_all	r	0	mute signal to analog
23	zd_amute	r	0	zero detect signal to analog
22:18	RSVD			
17	dac_s0_int_clr	r/w	0	clear and close interrupt
16	dac_s0_int	r	0	mute done interrupt status
15:14	RSVD			
13	dac_h0_mute_done	r	1	dvga mute done
12	dac_h0_busy	r	0	dvga busy
11:0	RSVD			

15.6.3 audac_s0

Address：0x20055008



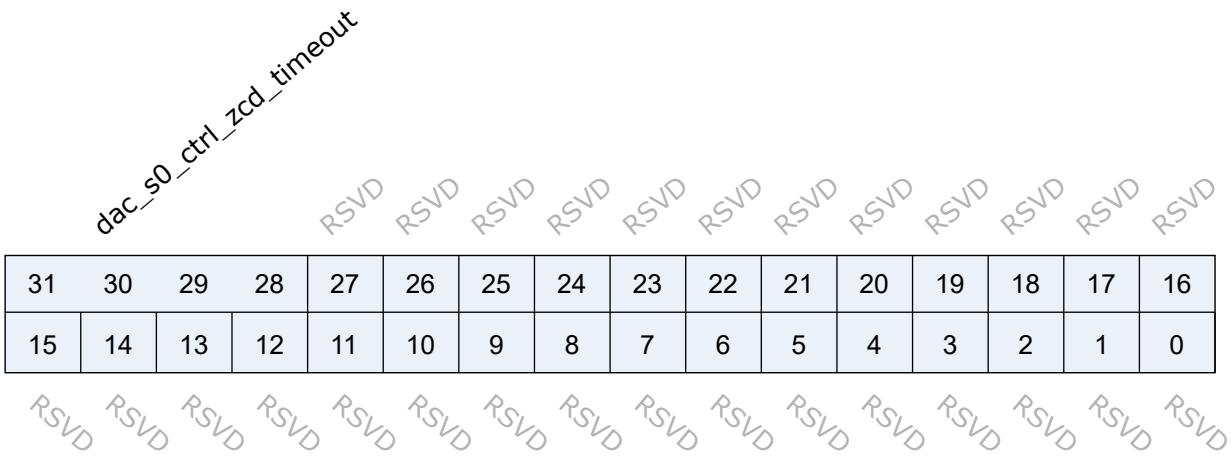
Bits	Name	Type	Reset	Description
31	dac_s0_mute	r/w	0	dac dpga ch0 sw volume control 1:mute
30	dac_s0_mute_softmode	r/w	1	0:mute directly, 1:mute with ramp down
29:26	dac_s0_mute_rmpdn_rate	r/w	4'd6	mute ramp down rate: 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample

Bits	Name	Type	Reset	Description
25:22	dac_s0_mute_rmpup_rate	r/w	4'd0	mute ramp up rate 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample
21:13	dac_s0_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step
12	dac_s0_volume_update	r/w	0	enable volume update
11:10	dac_s0_ctrl_mode	r/w	2'd2	0:direct force volume, 1:update volume at zero crossing, 2:update volume with ramp
9:6	dac_s0_ctrl_zcd_rate	r/w	4'd2	zero crossing rate 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample
5:2	dac_s0_ctrl_rmp_rate	r/w	4'd6	ramp rate 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample

Bits	Name	Type	Reset	Description
1:0	RSVD			

15.6.4 audac_s0_misc

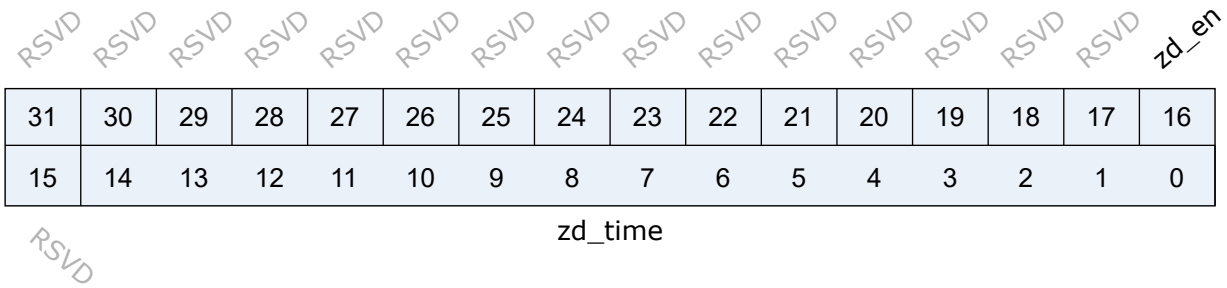
Address：0x2005500c



Bits	Name	Type	Reset	Description
31:28	dac_s0_ctrl_zcd_timeout	r/w	4'd4	zero crossing time out period 0:2 fs sample 1:4 fs sample 2:8 fs sample 3:16 fs sample 4:32 fs sample 5:64 fs sample 6:128 fs sample 7:256 fs sample 8:512 fs sample 9:1024 fs sample 8:2048 fs sample
27:0	RSVD			

15.6.5 audac_zd_0

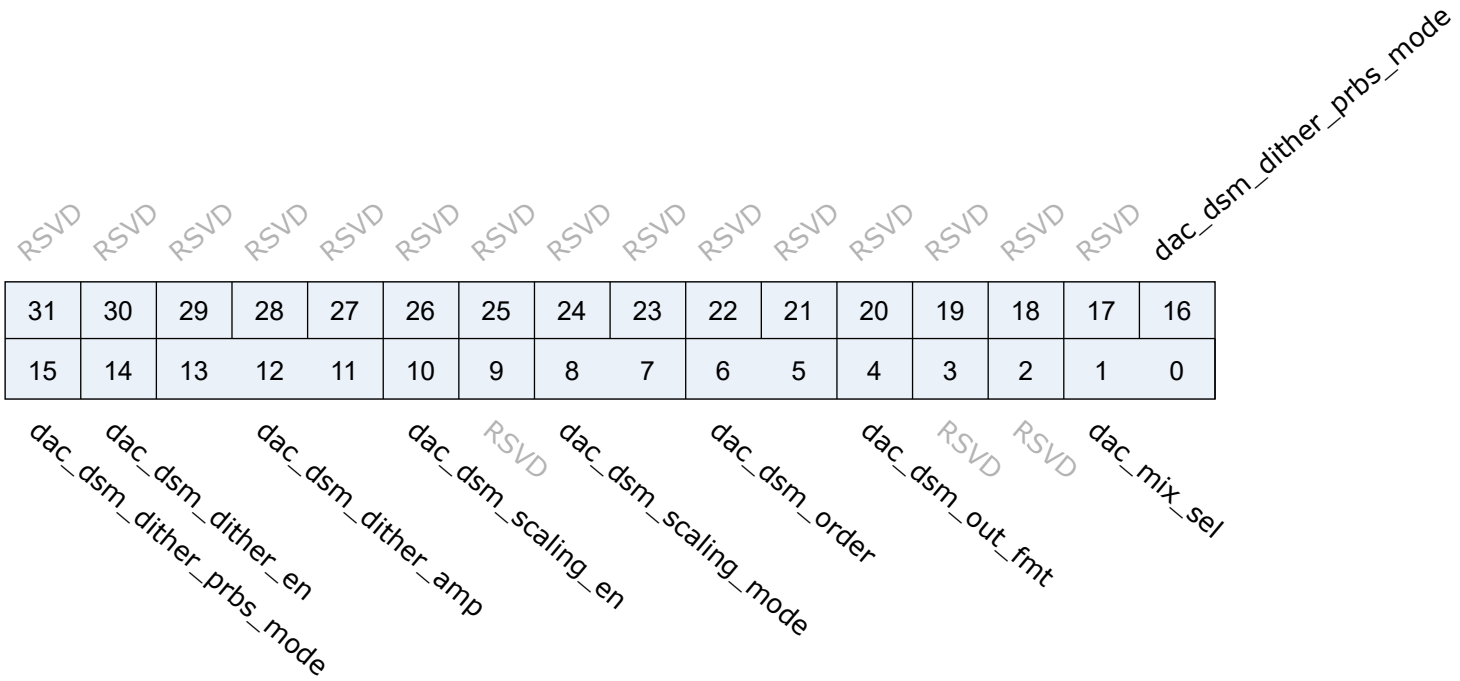
Address：0x20055010



Bits	Name	Type	Reset	Description
31:17	RSVD			
16	zd_en	r/w	0	enable zero detect
15	RSVD			
14:0	zd_time	r/w	15'd512	number of zeros

15.6.6 audac_1

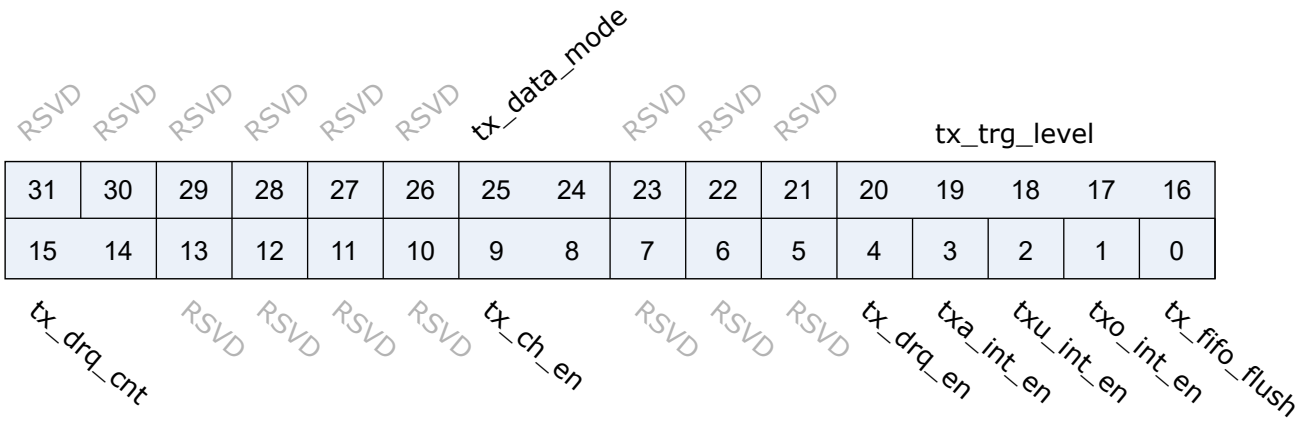
Address：0x20055014



Bits	Name	Type	Reset	Description
31:17	RSVD			
16:15	dac_dsm_dither_prbs_mode	r/w	0	dac dsm dither lfsr mode:0:LFSR32, 1:LFSR24, 2:LFSR16, 3:LFSR12
14	dac_dsm_dither_en	r/w	1	enable dac dsm dither
13:11	dac_dsm_dither_amp	r/w	0	dac dsm dither amplitue
10	dac_dsm_scaling_en	r/w	1	enable dac dsm scaling
9	RSVD			
8:7	dac_dsm_scaling_mode	r/w	0	dac dsm scaling value; u4.4
6:5	dac_dsm_order	r/w	0	0: 2-order, 1: 3-order
4	dac_dsm_out_fmt	r/w	0	offset binary 1:2's complement
3:2	RSVD			
1:0	dac_mix_sel	r/w	0	L channel, 1:R channel, 2: L+R, 3: (L+R)/2

15.6.7 audac_fifo_ctrl

Address：0x2005508c



Bits	Name	Type	Reset	Description
31:26	RSVD			

Bits	Name	Type	Reset	Description
25:24	tx_data_mode	r/w	2'b0	TX_FIFO_DATIN_MODE. TX FIFO DATA Input Mode (Mode 0, 1, 2, 3) Mode 0: Valid data's MSB is at [31] of TX_FIFO register Mode 1: Valid data's MSB is at [23] of TX_FIFO register Mode 2: Valid data's MSB is at [19] of TX_FIFO register Mode 3: Valid data's MSB is at [15] of TX_FIFO register For 16-bits transmitted audio sample: Mode 0: FIFO_I[15:0] = TXDATA[31:16] Mode 1: FIFO_I[15:0] = TXDATA[23:8] Mode 2: FIFO_I[15:0] = TXDATA[19:4] Mode 3: FIFO_I[15:0] = TXDATA[15:0]
23:21	RSVD			
20:16	tx_trg_level	r/w	5'd7	TX_FIFO_TRG_LEVEL. TX FIFO Trigger Level (TXTL[4:0]) Interrupt and DMA request trigger level for TX FIFO room available condition IRQ/DRQ Generated when WLEVEL > TXTL[4:0] Notes: WLEVEL represents the number of room available in the TX FIFO
15:14	tx_drq_cnt	r/w	2'b0	DAC_DRQ_CLR_CNT. When TX FIFO available room less than or equal N, DRQ Request will be de-asserted. N is defined here: 00: IRQ/DRQ de-asserted when WLEVEL <= TXTL[4:0] 01: IRQ/DRQ de-asserted when WLEVEL < 2 10: IRQ/DRQ de-asserted when WLEVEL < 4 11: IRQ/DRQ de-asserted when WLEVEL < 8 WLEVEL represents the number of room available in the TX FIFO
13:10	RSVD			
9:8	tx_ch_en	r/w	2'b0	TX_FIFO_DATOUT_DST. TX FIFO Data Output Destination Select. 0: Disable 1: Enable Bit9: DAC2 data Bit8: DAC1 data When some of the above bits set to ' 1' , these data are always arranged in order from low-bit to high-bit.(bit8->bit9)
7:5	RSVD			

Bits	Name	Type	Reset	Description
4	tx_drq_en	r/w	1'b0	DAC_DRQ_EN. DAC FIFO Room Available DRQ Enable. 0: Disable 1: Enable
3	txa_int_en	r/w	1'b0	DAC_IRQ_EN. DAC FIFO Room Available IRQ Enable. 0: Disable 1: Enable
2	txu_int_en	r/w	1'b0	DAC_UNDERRUN_IRQ_EN. DAC FIFO Under Run IRQ Enable 0: Disable 1: Enable
1	txo_int_en	r/w	1'b0	DAC_OVERRUN_IRQ_EN. DAC FIFO Over Run IRQ Enable 0: Disable 1: Enable
0	tx_fifo_flush	w1c	1'b0	DAC_FIFO_FLUSH. DAC FIFO Flush. Write '1' to flush TX FIFO, self clear to '0'.

15.6.8 audac_fifo_status

Address：0x20055090

RSVD RSVD RSVD RSVD RSVD RSVD RSVD txa RSVD RSVD RSVD txa_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD txa_int RSVD txu_int txo_int RSVD

Bits	Name	Type	Reset	Description
31:25	RSVD			

Bits	Name	Type	Reset	Description
24	txa	r	1'b1	TXA. TX FIFO Room Available 0: No room for new sample in TX FIFO 1: More than one room for new sample in TX FIFO (>= 1 word)
23:21	RSVD			
20:16	txa_cnt	r	5'd16	TXA_CNT. TX FIFO Available Room Word Counter
15:5	RSVD			
4	txa_int	r	1'b0	TXA_INT. TX FIFO Room Available Pending Interrupt 0: No Pending IRQ 1: Room Available Pending IRQ Automatic clear if interrupt condition fails.
3	RSVD			
2	txu_int	r	1'b0	TXU_INT. TX FIFO Underrun Pending Interrupt 0: No Pending IRQ 1: FIFO Underrun Pending IRQ Write '1' to clear this interrupt
1	txo_int	r	1'b0	TXO_INT. TX FIFO Overrun Pending Interrupt 0: No Pending IRQ 1: FIFO Overrun Pending IRQ Write '1' to clear this interrupt
0	RSVD			

15.6.9 audac_fifo_data

Address：0x20055094

tx_data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

tx_data

Bits	Name	Type	Reset	Description
31:0	tx_data	w	32'h0	TX_DATA. Transmitting left, right channel sample data should be written this register one by one. The left channel sample data is first and then the right channel sample.

16 AudioADC

16.1 Overview

The chip has a built-in AudioADC module for single-channel recording and an integrated 16bit high-precision ADC for analog audio signal acquisition, in addition to supporting the PDM digital interface.

16.2 Features

- One 16-bit ADC is integrated to support 1 analog mic differential/single-ended input and multiplexed GPIO input
  - Sampling rate: 8k~96k
  - Signal to noise ratio (AW): 90 dB @ 6 dB gain
  - Harmonic distortion + noise: -80dB @ 6dB gain
  - Analog pre-amplifier gain: 6~42 dB, 3 dB per gear
  - Configurable analog parameters such as input impedance of ADC
- Additional support for PDM digital interface for digital mic recording, multiplexing GPIO inputs
- Additional support for DC measurement mode in addition to audio recording mode, using AudioADC as a high precision ADC, supporting differential and single-ended modes with resolutions up to 16bit
- Adjustable high-pass filter and independent digital volume control
- 32-bit width FIFO depth of 32, data support 32bit/24bit/20bit/16bit four storage formats
- Supports DMA transfer mode

16.3 Clock Tree

Users need to first configure the Audio PLL output clock to 491.52M or 451.58M, corresponding to the 48K series and 44.1K series sample rate, respectively, and select the Audio PLL output of 5 divisions as the AudioDAC clock source. The AudioADC module will automatically configure the crossover coefficient of each sub-module according to the operating mode and sample rate setting, so there is no need to set the crossover coefficient manually. This register value and crossover frequency example are shown below.

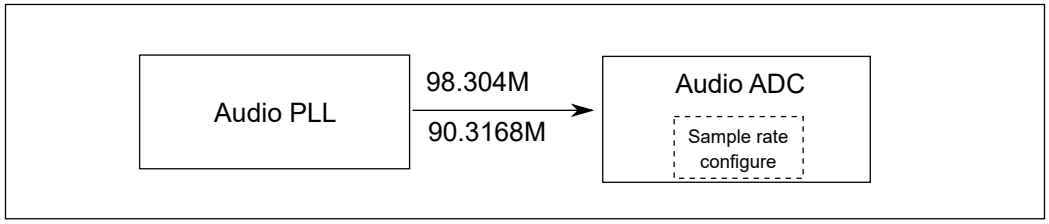


Fig. 16.1: Block Diagram of Clock

16.4 Functional Description

The block diagram of AudioADC is shown as follows.

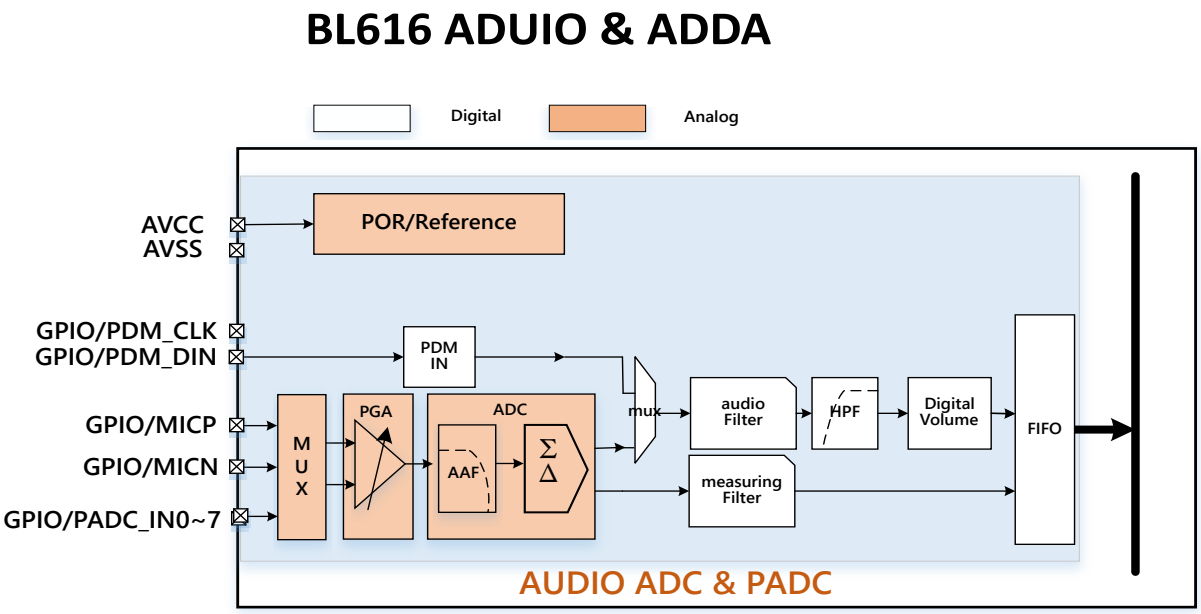


Fig. 16.2: Block Diagram of Module

The AUADC module input supports both audio analog signals and PDM digital signals. When selected as a PDM digital signal, the input data enters the audio processing channel after being processed by the PDM circuit.

When selected as a digital signal, the analog signal will first be amplified by the PGA and then result in the ADC

quantization process before entering the audio processing channel.

The volume processing channel will use SINC function low-pass filtering and resampling for the initial data, then high-pass filtering and volume gain control processing, and finally push the data into the FIFO.

16.4.1 Audio channel selection

The AudioADC module supports analog mic input and PDM digital mic interface, where the PDM timing specification has data on the rising and falling edges of the clock for the left and right channels, respectively.

When using the analog mic input, you need to configure the built-in ADC and select the integrated ADC to be After the configuration of the integrated ADC is completed, select the audio source as ADC by the adc_0_src bit of the pdm_dac_0 register.

When using the PDM digital mic, you need to start the PDM via pdm_0_en in pdm_dac_0 and then select the left channel or right channel via adc_0_pdm_sel in pdm_pdm_0.

16.4.2 AUADC Interrupt

AUADC supports the following interrupt control modes:

- RX FIFO request interrupt
- RX FIFO underrun interrupt
- RX FIFO overrun interrupt

A RX FIFO request interrupt is generated when RX_DRQ_CNT in RX_FIFO_CTRL is greater than RX_TRG_LEVEL. When the condition is not met, the interrupt flag is cleared automatically.

When there is no data in RX FIFO, but the user enables RX FIFO modulation through RX_CH_EN in RX_FIFO_CTRL, the RX FIFO underrun interrupt is generated.

When the user fills in data that exceeds the maximum depth of RX FIFO, it leads to RX FIFO overflow and cause a RX FIFO overrun interrupt.

16.4.3 FIFO Format Control

The rx_data_mode in the audadc_rx_fifo_ctrl register controls the format of the audio data stored in the FIFO.

The FIFO controller supports the following four data storage formats:

- Mode 0:

  DATA[31:0] = {FIFO[15:0],16’h0}

- Mode 1:

  DATA[31:0] = {8{FIFO[15]},FIFO[15:0],8’h0}

- Mode 2:

  DATA[31:0] = {12{FIFO[15]},FIFO[15:0],4’h0}

- Mode 3:

  DATA[31:0] = {16{FIFO[15]},FIFO[15:0]}

Distribution of MSB

- Mode 0:

  The MSB of data is 31 bits

- Mode 1:

  The MSB of data is 23 bits

- Mode 2:

  The MSB of data is 19 bits

- Mode 3:

  The MSB of data is 15 bits

If there is no special requirement for the storage format, generally, Mode3 is appropriate. As the maximum resolution of ADC is 16-bit, using 16-bit RAM to store audio can achieve the greatest efficiency. For other formats, the valid 16-bit data is placed in different positions in the 32-bit width, with low bits filled with 0 and high bits filled with sign bits.

16.4.4 Measurement Mode

The high performance ADC in AudioADC supports the use as high precision ADC in addition to audio analog signal sampling, and comes with adjustable gain PGA amplifier for single-ended and differential weak signal acquisition.

The measurement mode needs to be selected in the adc_rate segment of the audpdm_top register. The measurement mode is enabled by audadc_meas_filter_en position 1 in the audadc_cmd register.

In the audadc_pga_mode segment and audadc_pga_gain segment, select the analog channel for measurement, and in the audadc_pga_mode segment, configure the ADC mode as DC differential or DC single-ended mode, and the audio processing channel will be bypassed at this time. Other configurations such as FIFO format in measurement mode are the same as in audio mode.

16.4.5 Startup of FIFO and DMA Transfer

The data in FIFO of the PDM module can be transferred by DMA.

The user can obtain the current amount of valid data in FIFO in real time through the register PDM_RX_FIFO_STATUS.

The FIFO threshold for triggering a DMA request is selected by configuring rx_drq_cnt in audadc_rx_fifo_ctrl with 4 options, 8, 16, 32, or the same as the FIFO interrupt threshold configured in rx_trg_level.

When the value of FIFO count is greater than the set threshold and AUDADC_RX_FIFO_CTRL[4] DMA mode is enabled, a request is initiated to DMA.

Note that if the data in the FIFO is not read out in time after starting AudioADC, an overrun error will be triggered when the FIFO overflows, and data will be lost at the same time.

16.5 Configuration Process

1. For the sampling rate of the recorded audio, select the corresponding sampling rate through audpdm_top[31:28].
2. Configure the adc_0_src register of pdm_dac_0 depending on whether the recorded data source is PDM digital signal or analog signal.
3. In case of pdm format, select the channel of pdm through the adc_0_pdm_sel in pdm_pdm_0
4. Configure the DMA to transfer the RX FIFO data of Audio to the designated area in real time
5. Turn on the state machine through the rx_ch_en in audadc_rx_fifo_ctrl to start recording
6. Adjust the volume during recording (optional)

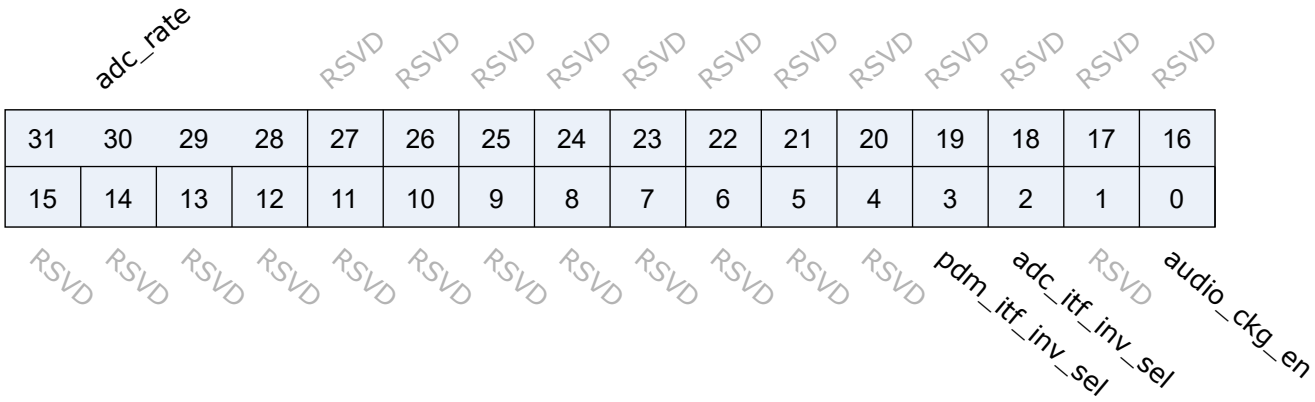
16.6 Register description

Name	Description
audpdm_top	Clock control register
audpdm_itf	Interface control register 1
pdm_adc_0	Filter control register 1
pdm_adc_1	Filter control register 2
pdm_dac_0	Interface control register 2
pdm_pdm_0	pdm control register
pdm_adc_s0	volume control register
audadc_ana_cfg1	ADC analog control register 1
audadc_ana_cfg2	ADC analog control register 2

Name	Description
audadc_cmd	ADC control register
audadc_data	measuring mode control register
audadc_rx_fifo_ctrl	fifo control register
audadc_rx_fifo_status	fifo status register
audadc_rx_fifo_data	fifo data register

16.6.1 audpdm_top

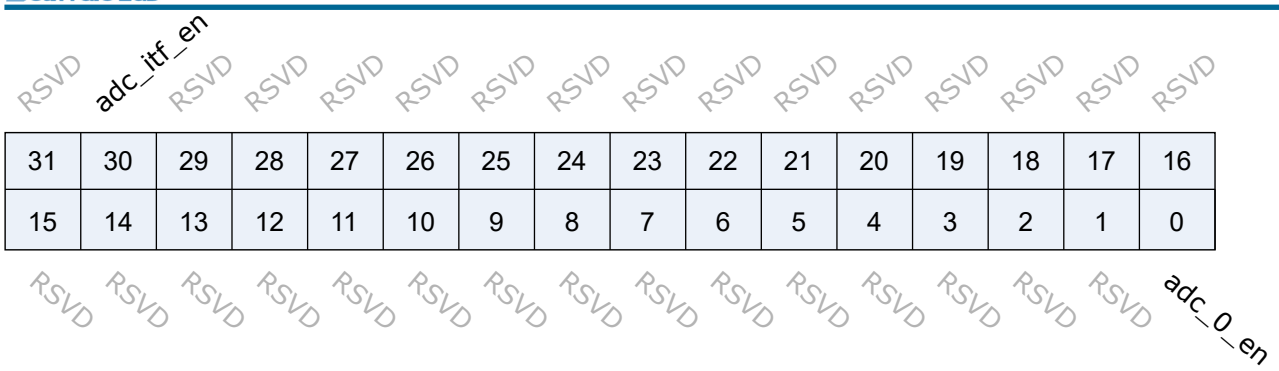
Address：0x2000ac00



Bits	Name	Type	Reset	Description
31:4	RSVD			
3	pdm_itf_inv_sel	r/w	1'd0	invert clk_pdm
2	adc_itf_inv_sel	r/w	1'd0	invert clk_adc
1	RSVD			
0	audio_ckg_en	r/w	1'd0	enable audio clock generator

16.6.2 audpdm_itf

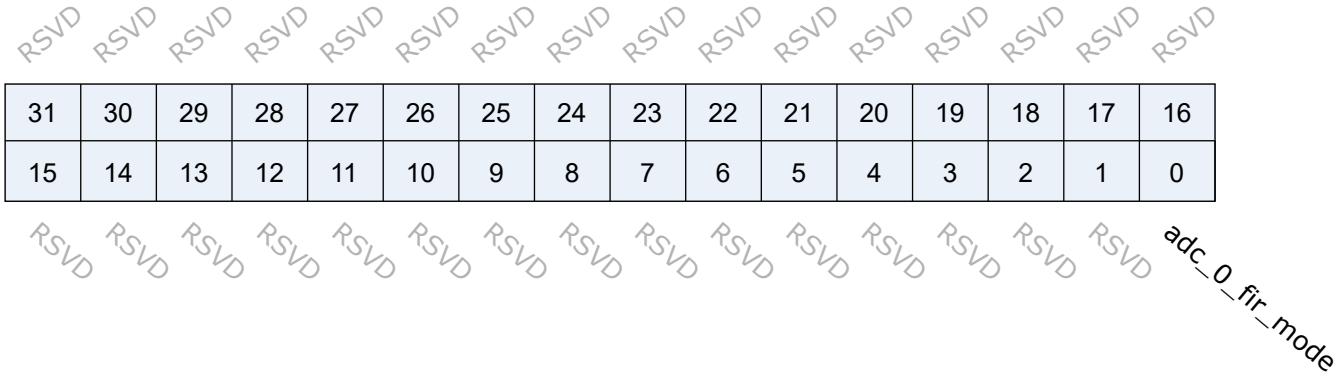
Address：0x2000ac04



Bits	Name	Type	Reset	Description
31	RSVD			
30	adc_itf_en	r/w	1'd0	enable adc to audio dma interface
29:1	RSVD			
0	adc_0_en	r/w	1'd0	enable adc

16.6.3 pdm_adc_0

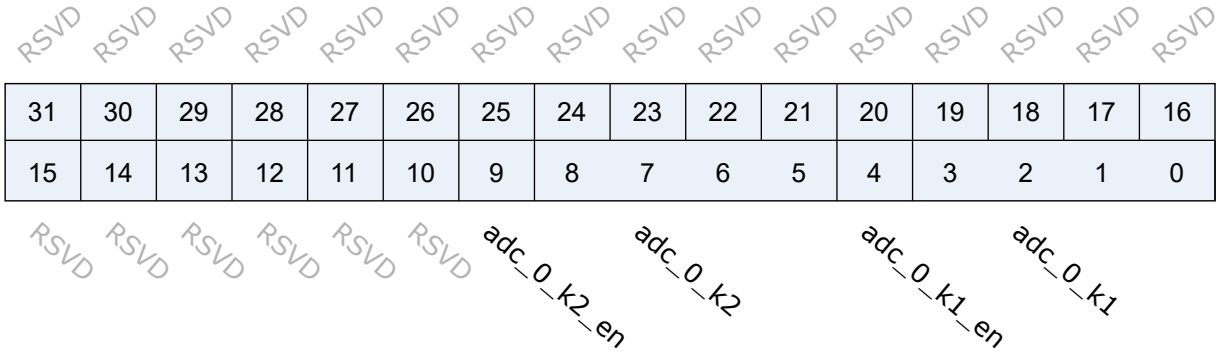
Address：0x2000ac08



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	adc_0_fir_mode	r/w	1'd0	adc fir mode

16.6.4 pdm_adc_1

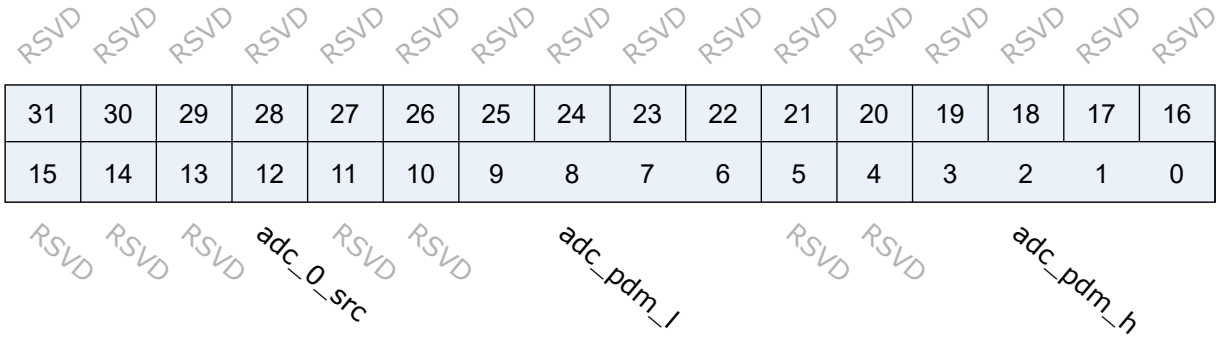
Address：0x2000ac0c



Bits	Name	Type	Reset	Description
31:10	RSVD			
9	adc_0_k2_en	r/w	1'd0	adc ch0 hpf parameter k2 enable
8:5	adc_0_k2	r/w	4'd13	adc ch0 hpf parameter k2
4	adc_0_k1_en	r/w	1'd1	adc ch0 hpf parameter k1 enable
3:0	adc_0_k1	r/w	4'd8	adc ch0 hpf parameter k1

16.6.5 pdm_dac_0

Address：0x2000ac10

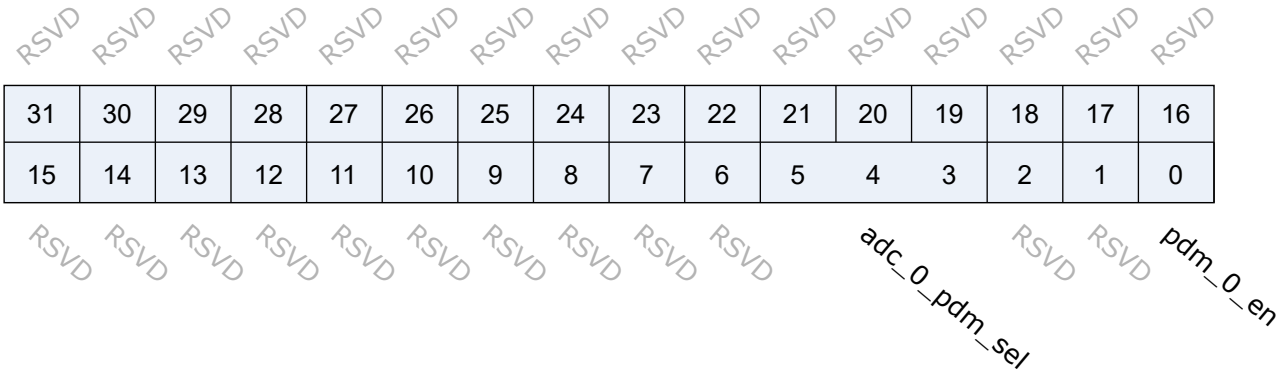


Bits	Name	Type	Reset	Description
31:13	RSVD			
12	adc_0_src	r/w	1'd0	0:adc mode, 1:pdm mode
11:10	RSVD			
9:6	adc_pdm_l	r/w	4'b1010	pdm low value

Bits	Name	Type	Reset	Description
5:4	RSVD			
3:0	adc_pdm_h	r/w	4'b0110	pdm high value

16.6.6 pdm_pdm_0

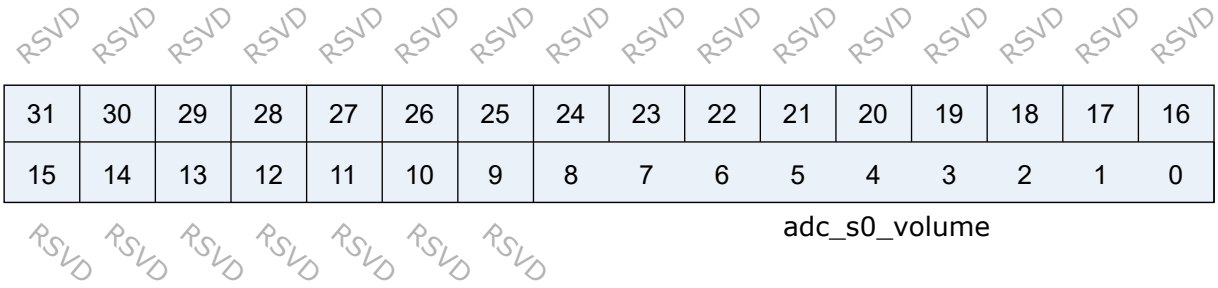
Address：0x2000ac1c



Bits	Name	Type	Reset	Description
31:6	RSVD			
5:3	adc_0_pdm_sel	r/w	3'd0	adc ch0 source select: 0:pdm_l, 1:pdm_r
2:1	RSVD			
0	pdm_0_en	r/w	1'd0	enable pdm

16.6.7 pdm_adc_s0

Address：0x2000ac38

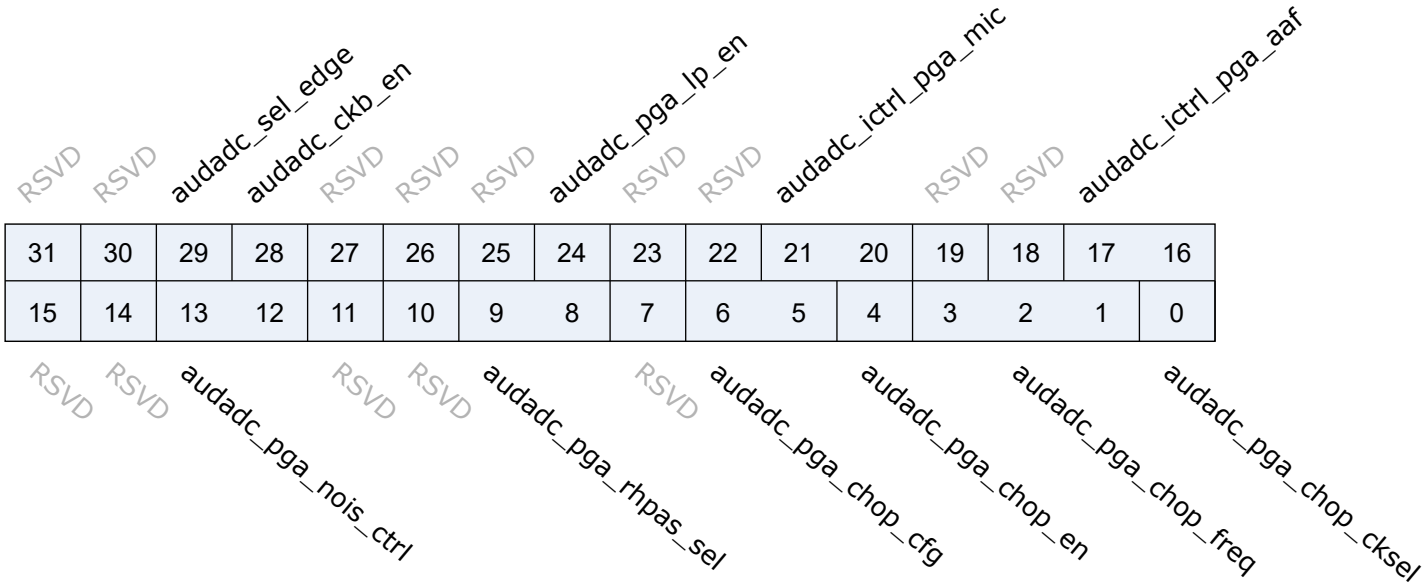


Bits	Name	Type	Reset	Description
31:9	RSVD			

Bits	Name	Type	Reset	Description
8:0	adc_s0_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

16.6.8 audadc_ana_cfg1

Address：0x2000ac60

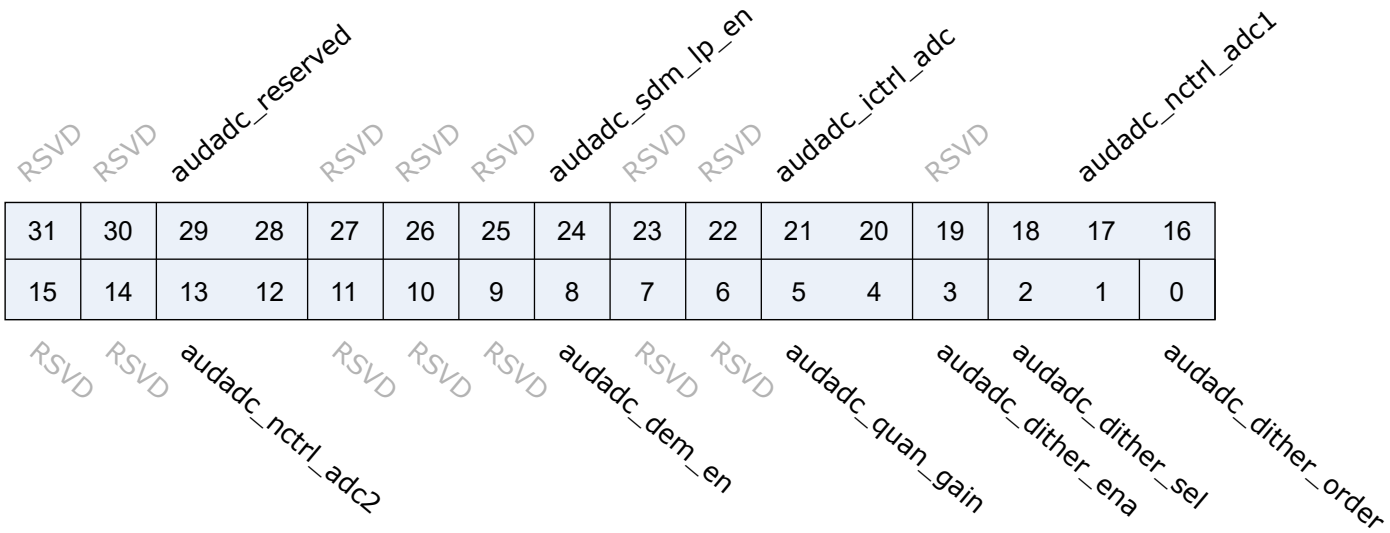


Bits	Name	Type	Reset	Description
31:30	RSVD			
29	audadc_sel_edge	r/w	1'h0	ADC output data clock edge 0 = falling edge sent, rising edge recieve 1 = rising edge sent, falling edge recieve
28	audadc_ckb_en	r/w	1'h0	AUDADC clock phase control 0 = 0° 1 = 180°
27:25	RSVD			
24	audadc_pga_lp_en	r/w	1'h0	PGA lowpower funciton reversed, not realized in circuit
23:22	RSVD			
21:20	audadc_ictrl_pga_mic	r/w	2'h1	PGA_OPMIC bias current control 00 = 4uA 01 = 5uA 10 = 6uA 11 = 7uA
19:18	RSVD			

Bits	Name	Type	Reset	Description
17:16	audadc_ictrl_pga_aaf	r/w	2'h1	PGA_OPAAF bias current control 00 = 4uA 01 = 5uA 10 = 6uA 11 = 7uA
15:14	RSVD			
13:12	audadc_pga_nois_ctrl	r/w	2'h0	PGA noise control when configured to single-ended not used
11:10	RSVD			
9:8	audadc_pga_rhpas_sel	r/w	2'h0	PGA high pass filter R control when configured to AC-coupled mode 00 = 480kΩ 01 = 320kΩ 10 = 160kΩ 11 = 4kΩ, fast startup
7	RSVD			
6:5	audadc_pga_chop_cfg	r/w	2'h3	control chopper for opmic&opaaf 00 = opmic off & opaaf off 01 = opmic off & opaaf on 10 = opmic on & opaaf off 11 = opmic on & opaaf on
4	audadc_pga_chop_en	r/w	1'h1	PGA chopper control 0 = disable 1 = enable
3:1	audadc_pga_chop_freq	r/w	3'h4	PGA chopper frequency control @Fs=2048k 000 = 8k 001 = 16k 010 = 32k 011 = 64k 100 = 128k 101 = 256k 110 = 512k 111 = 1024k
0	audadc_pga_chop_cksel	r/w	1'h0	PGA chopper clock source selection 0 = adc clock 1 = synchronized clock from SDM not used

16.6.9 audadc_ana_cfg2

Address：0x2000ac64

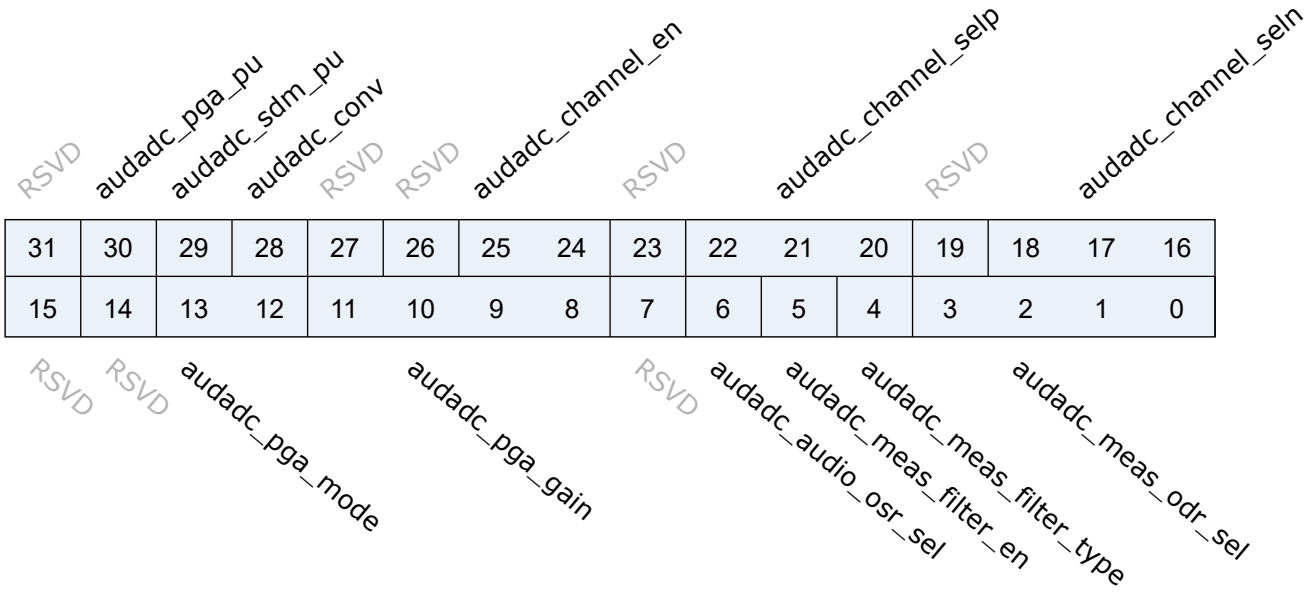


Bits	Name	Type	Reset	Description
31:30	RSVD			
29:28	audadc_reserved	r/w	2'h0	AUDADC reserved register
27:25	RSVD			
24	audadc_sdm_lp_en	r/w	1'h0	SDM lowpower funciton 0 = disable 1 = enable, 0.6 of disable
23:22	RSVD			
21:20	audadc_ictrl_adc	r/w	2'h1	SDM bias current control 00 = 4uA 01 = 5uA 10 = 6uA 11 = 7uA
19	RSVD			
18:16	audadc_nctrl_adc1	r/w	3'h3	op number control for first integrator in SDM 000 = 1(12uA) 001 = 2(24uA) 010 = 3(36uA) 011 = 4(48uA) 100 = 5(60uA) 101 = 5(60uA) 110 = 5(60uA) 111 = 5(60uA)
15:14	RSVD			
13:12	audadc_nctrl_adc2	r/w	2'h1	op number control for second integrator in SDM 00 = 1(12uA) 01 = 2(24uA) 10 = 3(36uA) 11 = 3(36uA)
11:9	RSVD			

Bits	Name	Type	Reset	Description
8	audadc_dem_en	r/w	1'h1	dem function control 0 = disable 1 = enable
7:6	RSVD			
5:4	audadc_quan_gain	r/w	2'h1	quantizer gain control for SDM 00 = Vref/14 01 = Vref/12 10 = Vref/10 11 = Vref/8
3	audadc_dither_ena	r/w	1'h1	dither control 0 = disable 1 = enable
2:1	audadc_dither_sel	r/w	2'h2	dither level control for SDM 00 = 0 01 = LSB*1/15 10 = LSB*2/15 11 = LSB*3/15
0	audadc_dither_order	r/w	1'h0	dither order control for SDM 0 = 0 order 1 = 1 order

16.6.10 audadc_cmd

Address：0x2000ac68



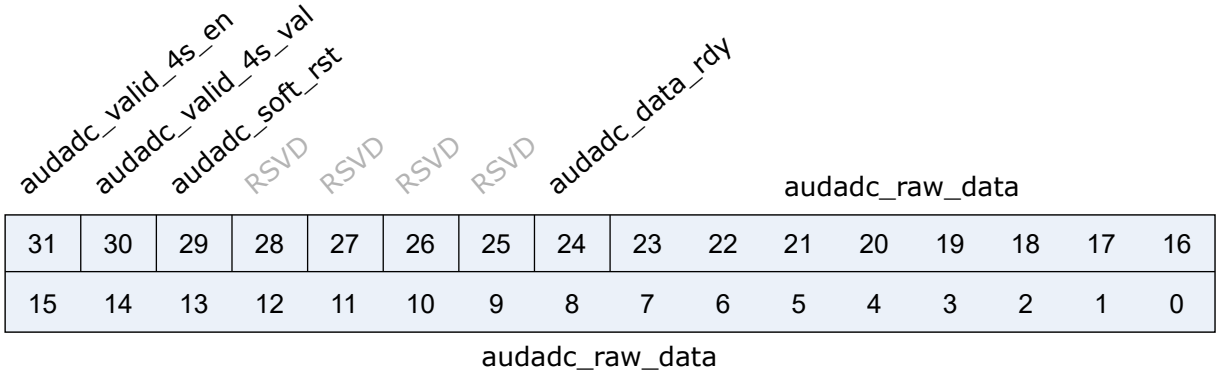
Bits	Name	Type	Reset	Description
31	RSVD			
30	audadc_pga_pu	r/w	1'h0	PGA related circuit enable 0 = disable 1 = enable

Bits	Name	Type	Reset	Description
29	audadc_sdm_pu	r/w	1'h0	SDM related circuit enable 0 = disable 1 = enable
28	audadc_conv	r/w	1'h0	SDM conversion start singal 0 = remain resetting status 1 = start conversion both analog intetrator and measuring digitial decimation filter will be reset when audadc_conv configured to low. Measuring digital decimation filter need to reset to same initial condition because it's feedback configuration for FIR. Audio filter dont need this resetting.
27:26	RSVD			
25:24	audadc_channel_en	r/w	2'h0	channel mux switch enable or disable MSB controls Positive channel, LSB controls Negative channel 0 = disable, look into each channel will see high impedance 1 = enable, one of eight channel will be choose
23	RSVD			
22:20	audadc_channel_selp	r/w	3'h0	Positive channel selection, connected to PGA positive terminal 000 = AIN0 001 = AIN1 010 = AIN2 011 = AIN3 100 = AIN4 101 = AIN5 110 = AIN6 111 = AIN7
19	RSVD			
18:16	audadc_channel_seln	r/w	3'h0	Negative channel selection, connected to PGA negative terminal 000 = AIN0 001 = AIN1 010 = AIN2 011 = AIN3 100 = AIN4 101 = AIN5 110 = AIN6 111 = AIN7
15:14	RSVD			
13:12	audadc_pga_mode	r/w	2'h0	PGA mode configuration 00: AC-Coupled & differential-ended, Audio application 01: AC-Coupled & single-ended, Audio application 10: DC-Coupled & differential-ended, Measuring application 11: DC-Coupled & single-ended, Measuring application(may not used)

Bits	Name	Type	Reset	Description
11:8	audadc_pga_gain	r/w	4'h0	PGA Gain control 0000 = 6dB 0001 = 6dB 0010 = 6dB 0011 = 9dB 0100 = 12dB 0101 = 15dB 0110 = 18dB 0111 = 21dB 1000 = 24dB 1001 = 27dB 1010 = 30dB 1011 = 33dB 1100 = 36dB 1101 = 39dB 1110 = 42dB 1111 = 42dB
7	audadc_audio_filter_en	r/w	1'h0	audio mode enable, audio filter is on when set to high 0 = disable 1 = enable
6	audadc_audio_osr_sel	r/w	1'h0	audio osr configuration 0 = 128 1 = 64
5	audadc_meas_filter_en	r/w	1'h0	measuring mode enable, measuring filter is on when set to high 0 = disable 1 = enable
4	audadc_meas_filter_type	r/w	1'h0	digital dicimation filter selection when in measuring mode 0 = SINC3 1 = Low-Latency
3:0	audadc_meas_odr_sel	r/w	4'h3	audadc ouput data rate selection when configured to measuring mode 0000 = 2.5SPS 0001 = 5SPS 0010 = 10SPS 0011 = 20SPS 0100 = 25SPS 0101 = 50SPS 0110 = 100SPS 0111 = 200SPS 1000 = 400SPS 1001 = 800SPS 1010 = 1000SPS 1011 = 2000SPS 1100 = 4000SPS 1101 = 4000SPS 1110 = 4000SPS 1111 = 4000SPS

16.6.11 audadc_data

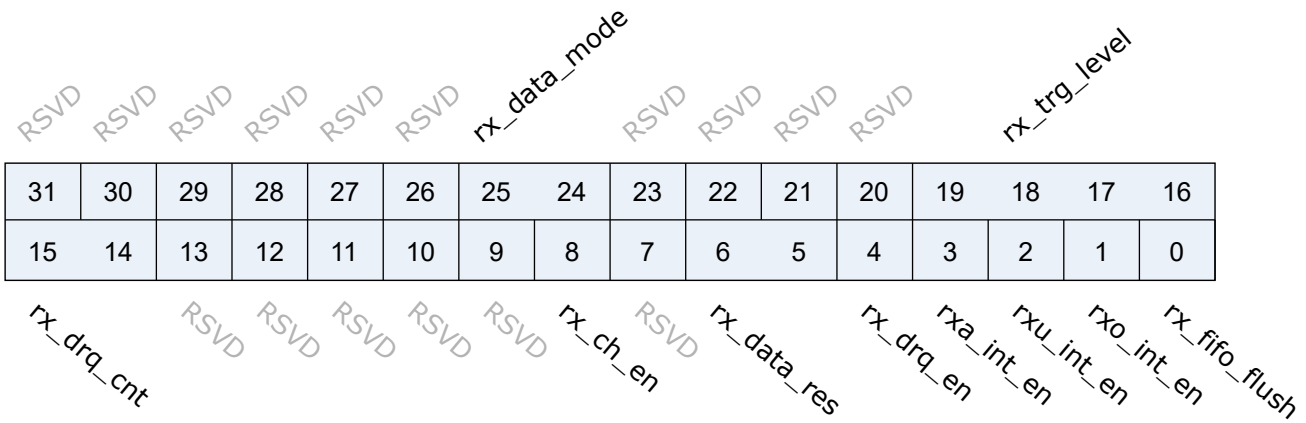
Address：0x2000ac6c



Bits	Name	Type	Reset	Description
31:30	RSVD			
29	audadc_soft_rst	r/w	1'h0	don't care
28:25	RSVD			
24	audadc_data_rdy	r	1'h0	audadc data ready indicator when measuring mode selected, auto reset to 0 after read 0 = not ready 1 = ready
23:0	audadc_raw_data	r	16'h0	audadc output 16bit data, 2's

16.6.12 audadc_rx_fifo_ctrl

Address：0x2000ac80



Bits	Name	Type	Reset	Description
31:26	RSVD			

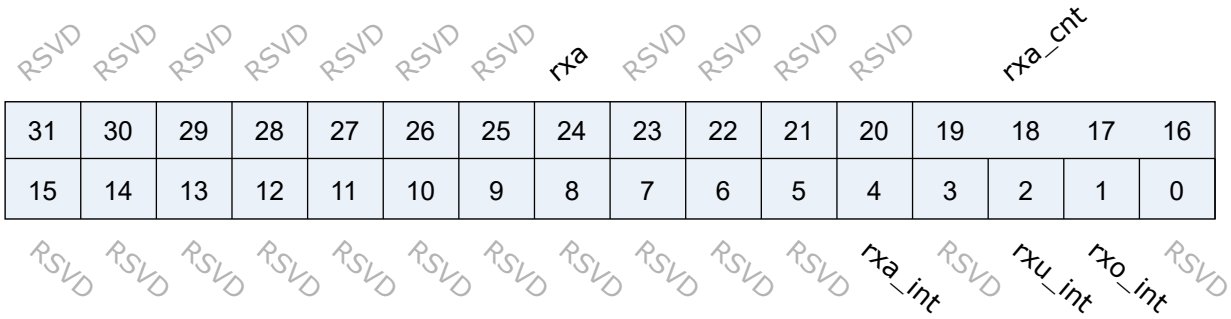
Bits	Name	Type	Reset	Description
25:24	rx_data_mode	r/w	2'b0	RX_FIFO_DATOUT_MODE. RX FIFO DATA Output Mode (Mode 0, 1, 2, 3) Mode 0: Valid data's MSB is at [31] of RX_FIFO register Mode 1: Valid data's MSB is at [23] of RX_FIFO register Mode 2: Valid data's MSB is at [19] of RX_FIFO register Mode 3: Valid data's MSB is at [15] of RX_FIFO register Note: Expanding '0' at LSB of RX FIFO register (data invalid region) Expanding sign bit at MSB of RX FIFO register (data invalid region) For 24-bit received audio sample resolution: Mode 0: RXDATA[31:0] = FIFO_O[23:0], 8' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[23], FIFO_O[23:0] Mode 2: RXDATA[31:0] = 12FIFO_O[23], FIFO_O[23:4] Mode 3: RXDATA[31:0] = 16FIFO_O[23], FIFO_O[23:8] For 20-bit received audio sample resolution: Mode 0: RXDATA[31:0] = FIFO_O[23:4], 12' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[23], FIFO_O[23:4], 4' h0 Mode 2: RXDATA[31:0] = 12FIFO_O[23], FIFO_O[23:4] Mode 3: RXDATA[31:0] = 16FIFO_O[23], FIFO_O[23:8] For 16-bit received audio sample resolution: Mode 0: RXDATA[31:0] = FIFO_O[23:8], 16' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[23], FIFO_O[23:8], 8' h0 Mode 2: RXDATA[31:0] = 12FIFO_O[23], FIFO_O[23:8], 4'h0 Mode 3: RXDATA[31:0] = 16FIFO_O[23], FIFO_O[23:8]
23:20	RSVD			
19:16	rx_trg_level	r/w	4'd3	RX_FIFO_TRG_LEVEL. RX FIFO Trigger Level (RXTL[3:0]) Interrupt and DMA request trigger level for RX FIFO Data Available condition IRQ/DRQ Generated when WLEVEL > RXTL[3:0] Notes: WLEVEL represents the number of valid samples in the RX FIFO

Bits	Name	Type	Reset	Description
15:14	rx_drq_cnt	r/w	2'b0	RX_DRQ_CLR_CNT. When RX FIFO available data less than or equal N, DRQ Request will be de-asserted. N is defined here: 00: IRQ/DRQ de-asserted when WLEVEL <= RXTL[3:0] 01: IRQ/DRQ de-asserted when WLEVEL < 1 10: IRQ/DRQ de-asserted when WLEVEL < 2 11: IRQ/DRQ de-asserted when WLEVEL < 4 WLEVEL represents the number of valid samples in the RX FIFO
13:9	RSVD			
8	rx_ch_en	r/w	1'b0	RX_FIFO_DATIN_SRC. RX FIFO Data Input Source Select. 0: Disable 1: Enable Bit8: ADC1 data
7	RSVD			
6:5	rx_data_res	r/w	2'd0	RX_SAMPLE_BITS. Receiving Audio Sample Resolution 0: 16 bits 1: 20 bits 2: 24 bits 3: Reserved
4	rx_drq_en	r/w	1'b0	ADC_DRQ_EN. ADC FIFO Data Available DRQ Enable. 0: Disable 1: Enable
3	rxa_int_en	r/w	1'b0	ADC_IRQ_EN. ADC FIFO Data Available IRQ Enable. 0: Disable 1: Enable
2	rxu_int_en	r/w	1'b0	ADC_UNDERRUN_IRQ_EN. ADC FIFO Under Run IRQ Enable 0: Disable 1: Enable
1	rxo_int_en	r/w	1'b0	ADC_OVERRUN_IRQ_EN. ADC FIFO Over Run IRQ Enable 0: Disable 1: Enable

Bits	Name	Type	Reset	Description
0	rx_fifo_flush	w1c	1'b0	ADC_FIFO_FLUSH. ADC FIFO Flush. Write '1' to flush TX FIFO, self clear to '0'.

16.6.13 audadc_rx_fifo_status

Address：0x2000ac84



Bits	Name	Type	Reset	Description
31:25	RSVD			
24	rxa	r	1'b0	RXA. RX FIFO Available 0: No available data in RX FIFO 1: More than one sample in RX FIFO (>= 1 word)
23:20	RSVD			
19:16	rxa_cnt	r	4'h0	RXA_CNT. RX FIFO Available Sample Word Counter
15:5	RSVD			
4	rxa_int	r	1'b0	RXA_INT. RX FIFO Data Available Pending Interrupt 0: No Pending IRQ 1: Data Available Pending IRQ Automatic clear if interrupt condition fails.
3	RSVD			
2	rxu_int	r	1'b0	RXU_INT. RX FIFO Underrun Pending Interrupt 0: No Pending IRQ 1: FIFO Underrun Pending IRQ Write '1' to clear this interrupt

Bits	Name	Type	Reset	Description
1	rxo_int	r	1'b0	RXO_INT. RX FIFO Overrun Pending Interrupt 0: No Pending IRQ 1: FIFO Overrun Pending IRQ Write '1' to clear this interrupt
0	RSVD			

16.6.14 audadc_rx_fifo_data

Address：0x2000ac88

rx_data

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

rx_data

Bits	Name	Type	Reset	Description
31:0	rx_data	r	32'h0	RX_DATA. RX Sample Host can get one sample by reading this register. The left channel sample data is first and then the right channel sample.

17 Emac

17.1 Overview

The EMAC module is a 100Mbps Ethernet Media Access Controller (Ethernet MAC) compatible with IEEE 802.3. It consists of status and control register set, RX/TX module, RX/TX buffer descriptor (BD) set, master interface, MDIO interface, and physical layer chip (PHY) interface.

The status and control register set contains the status and control bits of EMAC. As the interface with the user program, it controls data sending and receiving and reports the status.

The RX/TX module obtains data frames from the specified memory according to the control words in the RX/TX descriptor, adds preamble and CRC, and expands short frames to send them through PHY. Or, it receives data from PHY, and puts them into the specified memory according to the RX/TX BD. Relevant event flags are set after sending and receiving are completed. If the event interrupt is enabled, an interrupt request will be sent to the master for processing.

MDIO and MII/RMII interfaces communicate with PHY, including reading and writing the registers of PHY and sending and receiving data packets.

17.2 Features

- Compatible with the MAC layer defined by IEEE 802.3
- PHY supporting MII/RMII interface defined by IEEE 802.3
- Interacts with PHY through MDIO interface
- Supports 100 Mbps Ethernet
- Supports half-duplex and full-duplex
- Supports automatic flow control and control frame generation in the full-duplex mode
- Supports collision detection and retransmission in the half-duplex mode

- Supports the generation and verification of CRC
- Generates and removes data frame preamble
- Supports automatic extension of short data frames when sending
- Detects too long/short data frames (length limit)
- Transmits long data frames (> standard Ethernet frame length)
- Automatically discards data packets with over-limit retransmission times or too small frame gap
- Broadcast packet filtering
- Internal RAM for storing up to 128 BDs
- Splits and configures a data packet to multiple consecutive Bds when sending
- Various event flags sent or received
- Generates a corresponding interrupt when an event occurs

17.3 Functional Description

The composition of EMAC module is as follows.

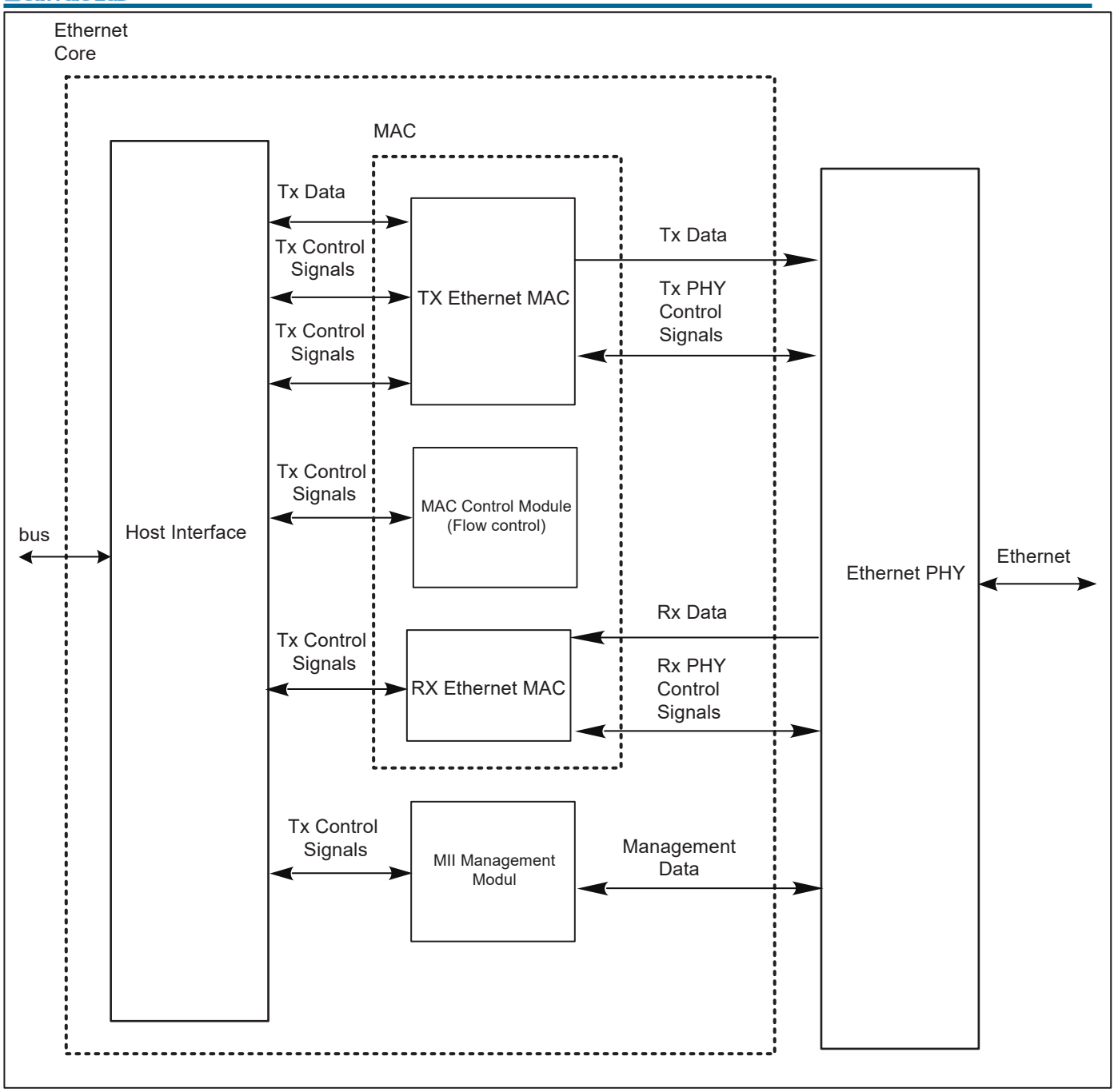


Fig. 17.1: Block diagram of EMAC

Through MDIO interface, the control register can read and write PHY’s registers, to perform configuration, mode selection (half/full duplex), negotiation, and other operations. The RX module filters and checks the received data frames for a valid preamble, FCS, and length, and stores the data in the specified memory address according to the BD. The TX module gets data from the memory according to the data BD, adds preamble, FCS, and pad, and then sends them out using the CSMA/CD protocol. If CRS is detected, retry will be delayed. The RX/TX BD set is connected to the system RAM, which is used to store the sent and received Ethernet data frames. Each descriptor contains the corresponding control status word and buffer memory address. There are 128 descriptors for RX/TX, and can be allocated flexibly.

17.4 Clock

EMAC needs a clock for synchronous transmission and reception (25 MHz (MII) or 50 MHz (RMII) at 100 Mbps, BL618 currently only supports RMII interface). The clock must be synchronized between EMAC and PHY. BL618 supports the output of a 50MHz reference clock source and also allows for the input of an external reference clock source.

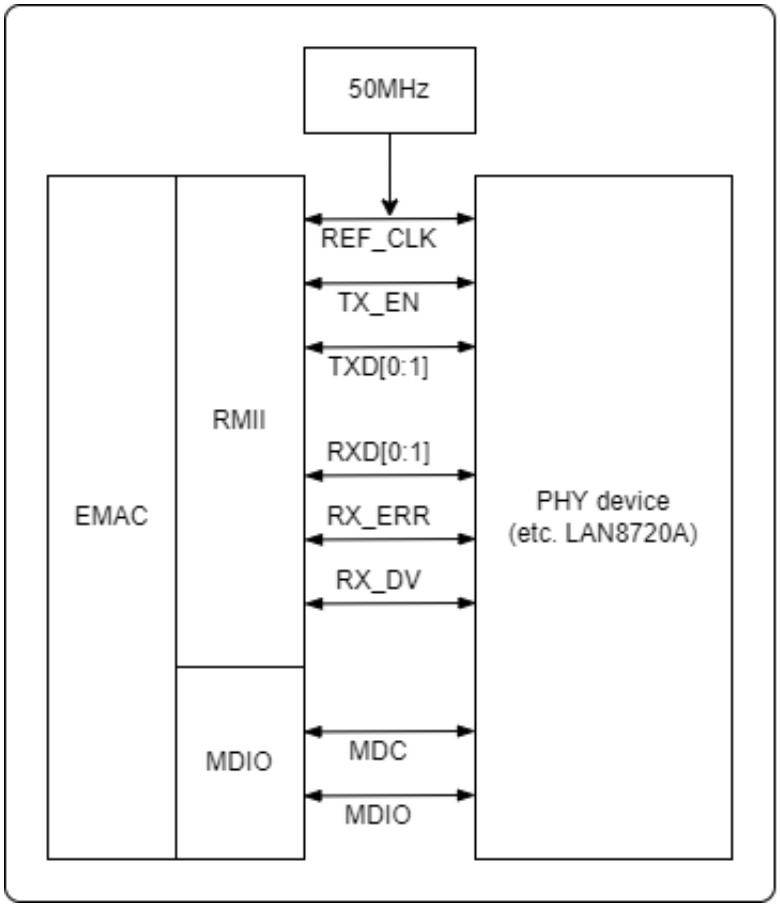


Fig. 17.2: Utilize an external reference clock source

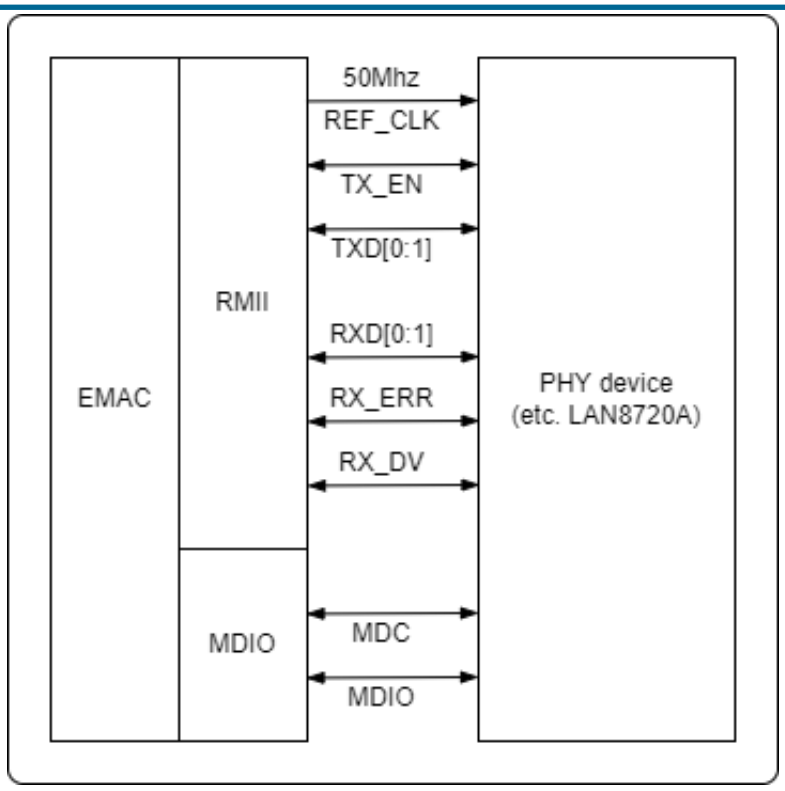


Fig. 17.3: Internal output 50MHz reference clock source

17.5 RX/TX BD

The RX/TX BD provides the association between EAMC and data frame cache address information, controls RX/TX data frames, and gives RX/TX status prompt. Each descriptor consists of two consecutive words (1 word = 32 bits). The word0 with low address provides the length, control bits, and status bits of the data frames contained in this buffer. The word1 with high address is the memory pointer.

Specific description of word0 in TX BD:

[31:16]: TX packet length (LEN).

[15]: TX BD Ready (RD) flag. The software writes “1” to inform EMAC that this BD contains data to be sent, and the hardware writes “0” to indicate that the BD data has been sent or an error has occurred.

[14]: Interrupt Request (IRQ) flag bit. When set, this BD can request TXE or TXB interrupt.

[13]: Wraparound (WR) flag. When set, it indicates that this BD is the last TX BD, and the hardware sends it again from the starting BD.

[12]: Padding (PAD). When it is set and a padding permission is set in EMAC, TX BD automatically fills the too short packet.

[11]: Cyclic Redundancy Check (CRC). When set, EMAC automatically calculates the CRC of the sent packet and attach it to the packet.

[10]: End of Frame (EoF) flag. If a frame of data occupies multiple BDs, this bit marks the end of this frame of data.

[8]: Underrun (UR) flag. When set, it indicates that the FIFO underrun error occurred during BD transmission.

[7:4]: Retry (RTRY) times counter. It counts the retry times.

[3]: Retry Limit (RL) flag. When set, it indicates that the retry times exceed the maximum retry times (MAXRET) configured in COLLCONF.

[2]: Late Collision (LC) flag. When set, it indicates that late collision occurred when this BD is sent.

[1]: Defer Indication (DF) flag. When set, it indicates that this packet is delayed.

[0]: Carrier Sense (CS) failure. If no carrier is detected during sending, it is set.

Specific description of word0 in RX BD:

[31:16]: TX packet length (LEN).

[15]: RX BD (RD) empty flag. When set, it indicates that this BD is empty (no received data is saved). "Clearing" indicates that this BD has received data or an error occurred during receiving.

[14]: Interrupt Request (IRQ) flag bit. When set, this BD can request RXE or RXB interrupt.

[13]: Wraparound (WR) flag. When set, it indicates that the BD is the last RX BD, and the hardware sends it again from the starting BD.

[8]: Control Frame (CF) flag. When set, it indicates that this BD has received one Control Frame.

[7]: Miss (M) flag. If a packet is received in promiscuous mode but it is marked as Miss by the internal address logic, EMAC sets this flag bit.

[6]: Overrun (OR) flag. When set, it indicates that the FIFO overrun error occurred during receiving.

[5]: Receive Error (RE) flag. When set, it indicates that the RX ERR signal sent by PHY is received during receiving.

[4]: Dribble Nibble (DN) flag. When set, it indicates that an odd number of nibbles have been received.

[3]: Too Long (TL) packet flag. When set, it indicates that the received packet is too long, exceeding the setting value.

[2]: Too Short (SF) packet flag. When set, it indicates that the received packet is shorter than the minimum allowable length.

[1]: CRC (CRC) error flag. When set, it indicates that the CRC of the received packet fails.

[0]: Late Collision (LC) flag. When set, it indicates that Late Collision occurred when data is received to this BD.

It should be noted that BD must be written by word.

EMAC supports 128 BDs, which are shared by the RX/TX logic and can be freely combined. But the TX BD always occupies the preceding contiguous area. The number of BD is specified by the TXBDNUM field in the register MAC_TX_BD_NUM.

EMAC circularly processes the RX/TX BDs according to their order, until it finds the BDs marked WR, and goes back to the first RX/TX BD respectively.

17.6 PHY Interaction

The interactive register set of PHY provides a way to communicate commands and data needed for interaction with PHY. EMAC controls the working mode of PHY through MDIO interface, and ensures the matching of both working modes (such as rate, full/half-duplex).

Data packets interact between EMAC and PHY through MII/RMII interface, which is selected by the RMII_EN bit in the EMAC’s mode register (EMAC_MODE): When this bit is 1, RMII mode is selected, and otherwise the MII mode is selected.

RMII modes support the transmission rates of 100 Mbps specified in IEEE 802.3u. The transmission signals of RMII is described as follows.

Table 17.1: Transmission signal

Name	MII	RMII
EXTCK_EREFCK	ETXCK: Send clock signal	EREFCK:reference clock
ECRS	ECRS: carrier detection	-
ECOL	ECOL: collision detection	-
ERXDV	ERXDV: valid data	ECRSDV: carrier detection/valid data
ERX0-ERX3	ERX0ERX3: 4-bit received data	ERX0ERX1: 2-bit received data
ERXER	ERXER: Receive error indication	ERXER: Receive error indication
ERXCK	ERXCK: Receive clock signal	-
ETXEN	ETXEN: TX enable	ETXEN: TX enable
ETX0-ETX3	ETX0ETX3: 4-bit sent data	ETX0ETX1: 2-bit sent data
ETXER	ETXER: Send error indication	-
EMDC	MDIO Clock	MDIO Clock
EMDIO	MDIO Data Input Output	MDIO Data Input Output

The RMII interface has fewer pins, and 2-bit data lines are used for transmission and reception. At 100 Mbps, a reference clock of 50 MHz is required.

17.7 Programming Flow

17.7.1 PHY initialization

- Select a proper connection mode by setting the RMII_EN bit in the register EMAC_MODE according to the PHY type.
- Set the MAC address of EMAC to EMAC_MAC_ADDR0 and EMAC_MAC_ADDR1
- Set an appropriate clock for the MDIO part by programming the CLKDIV field in the register EMAC_MIIMODE
- Set the address of the corresponding PHY to the FIAD field of the register EMAC_MIIADDRESS
- According to PHY’s manual, send commands through registers EMAC_MIICOMMAND and EMAC_MIITX_DATA
- Store the data read from PHY in the register EMAC_MIIRX_DATA
- Query the status of interaction with PHY commands through the register EMAC_MIISTATUS

After basic interaction is completed, PHY shall be switched to the auto-negotiation state. Upon negotiation completed, the mode must be programmed to the FULLD bit in the EMAC_MODE register based on the negotiation result.

17.7.2 Send Data Frame

- Configure data frame format and interval bit fields in the register EMAC_MODE
- Specify the number of TX BDs by setting the TXBDNUM field in the register EMAC_TX_BD_NUM, so that the rest is the number of RX BDs
- Prepare the data frames to be sent in the memory
- Fill in the address of data frames in the data pointer field (word 1) corresponding to the TX BDs
- Clear the status flag in the control and status fields (word 0) corresponding to the TX BDs, and set the control field (CRC enable, PAD enable, and interrupt enable)
- Write the data frame length, and set the RD field to inform EMAC that this BD data needs to be sent. If necessary, set the upper IRQ bit to enable interrupt
- Especially, if it is the last TX BD, the upper WR bit must be set. EMAC will ”go back” to the first TX BD after this BD is processed
- If there are multiple BDs to be sent, repeat the steps of setting BD to pad all the TX BDs
- If one packet is contained in only one BD, set its EOF bit to 1
- If one packet is sent in multiple BDs, just mark the last BD it occupies as the end of the packet by setting the EOF bit
- To enable the TX interrupt, configure the TX-related bits in the register EMAC_INT_MASK

- Configure the TXEN bit in the register EMAC_MODE to enable TX
- If an interrupt is enabled, in the TX interrupt, obtain the current BD through the TXBDNUM field in the register EMAC_TX_BD_NUM
- Complete processing based on the status word of the current BD
- For BDs whose data has been sent, the RD bit in its control field will be cleared by hardware and BDs will not be used for TX again. Only after new data is padded and RD is set, can this BD be used for TX again

17.7.3 Receive Data Frame

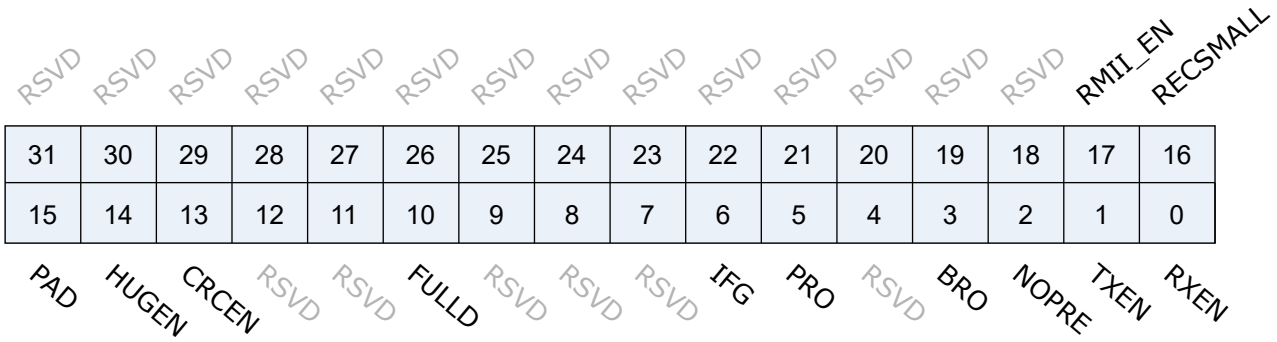
- Configure data frame format and interval bit fields in the register EMAC_MODE
- Specify the number of TX BDs by setting the TXBDNUM field in the register EMAC_TX_BD_NUM, so that the rest is the number of RX BDs
- Prepare an area in memory for receiving data
- Fill in the address of data frames in the data pointer field (word 1) corresponding to the RX BDs
- Clear the status flag in the control and status fields (word 0) corresponding to the RX BDs, and set the control field (interrupt enable)
- Write the receivable data frame length, and set the E-bit field to inform EMAC that this BD is free and can receive data. If necessary, set the upper IRQ bit to enable the interrupt
- Especially, if it is the last valid RX BD, the upper WR bit must be set. EMAC will ”go back” to the first RX BD after this BD is processed
- If there are multiple BDs for receiving data, repeat the steps of setting BD to pad all the BDs
- To enable the RX interrupt, configure the RX-related bits in the register EMAC_INT_MASK
- Configure the RXEN bit in the register EMAC_MODE to enable RX
- If an interrupt is enabled, in the RX interrupt, obtain the current BD through the RXBDNUM field in the register EMAC_TX_BD_NUM
- Complete processing based on the status word of the current BD
- For BDs whose data has been received, the E bit in its control field will be cleared by hardware and BDs will not be used for RX again. Only after you take out the data and set the E bit, can this BD be used for RX again

17.8 Register description

Name	Description
MODE	ethernet mac mode setting register
INT_SOURCE	interrupt control register
INT_MASK	interrupt mask register
IPGT	inter packet gap register
PACKETLEN	packet length control register
COLLCONFIG	collision and retry config register
TX_BD_NUM	transmit buffer number register
MIIMODE	MII mode config register
MIICOMMAND	MII command config register
MIIADDRESS	MII address config register
MIITX_DATA	MII transmit data register
MIIRX_DATA	MII read data register
MIISTATUS	MII status register
MAC_ADDR0	ethernet mac address0 register
MAC_ADDR1	Ethernet mac address1 register
HASH0_ADDR	hash0 address register
HASH1_ADDR	hash1 address register
TXCTRL	Transmit control register

17.8.1 MODE

Address：0x20070000



Bits	Name	Type	Reset	Description
31:18	RSVD			
17	RMII_EN	r/w	1'b0	RMII mode enable 0: MII PHY I/F is used 1: RMII PHY I/F is used
16	RECSMALL	r/w	1'b0	Receive small frame enable 0: Frames smaller than MINFL are ignored. 1: Frames smaller than MINFL are accepted.
15	PAD	r/w	1'b1	Padding enable 0: Do not add pads to frames shorter than MINFL. 1: Add pads to short frames, until the length equals MINFL.
14	HUGEN	r/w	1'b0	Huge frames enable 0: The maximum frame length is MAXFL. All additional bytes are dropped. 1: Frame size is not limited by MAXFL and can be up to 64K bytes.
13	CRCEN	r/w	1'b1	CRC Enable 0: TX MAC does not append CRC field. 1: TX MAC will append CRC field to every frame.
12:11	RSVD			
10	FULLD	r/w	1'b0	Full duplex 0: Half duplex mode. 1: Full duplex mode.
9:7	RSVD			
6	IFG	r/w	1'b0	Inter frame gap check 0: IFG is verified before each frame be received. 1: All frames are received regardless to IFG requirement.
5	PRO	r/w	1'b0	Promiscuous mode enable 0: The destination address is checked before receiving. 1: All frames received regardless of the address.

Bits	Name	Type	Reset	Description
4	RSVD			
3	BRO	r/w	1'b1	Broadcast address enable 0: Reject all frames containing the broadcast address unless the PRO bit is asserted. 1: Receive all frames containing broadcast address.
2	NOPRE	r/w	1'b0	No preamble mode 0: 7-byte preamble will be sent. 1: No preamble will be sent.
1	TXEN	r/w	1'b0	Transmit enable 0: Transmitter is disabled. 1: Transmitter is enabled. If TX_BD_NUM equals 0x0 (zero buffer descriptors are used), then the transmitter is disabled regardless of TXEN.
0	RXEN	r/w	1'b0	Receiver enable 0: Receiver is disabled. 1: Receiver is enabled. If TX_BD_NUM equals 0x80 (all buffer descriptors are used for TX), then the receiver is disabled regardless of RXEN.

17.8.2 INT_SOURCE

Address：0x20070004

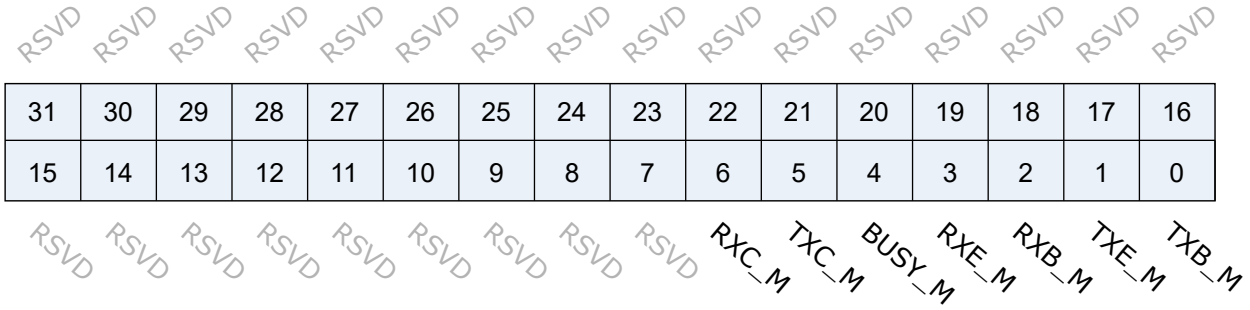
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RXC	TXC	BUSY	RXE	RXB	TXE	TXB

Bits	Name	Type	Reset	Description
31:7	RSVD			
6	RXC	r/w	1'b0	Receive control frame This bit indicates that the control frame was received. It is cleared by writing 1 to it. Bit RXFLOW in the CTRLMODE register must be set to 1 in order to get the RXC bit set.

Bits	Name	Type	Reset	Description
5	TXC	r/w	1'b0	Transmit control frame This bit indicates that a control frame was transmitted. It is cleared by writing 1 to it. Bit TXFLOW in the CTRLMODE register must be set to 1 in order to get the TXC bit set.
4	BUSY	r/w	1'b0	Busy This bit indicates that RX packet is being received and there is no empty buffer descriptor to use. It iscleared by writing 1 to it. This bit appears regardless to the IRQ bits in the Receive Buffer Descriptor.
3	RXE	r/w	1'b0	Receive error This bit indicates that an error occurred while receiving data (overrun, receiver error, dribble nibble, too long, >64K, CRC error, bus error or late collision. It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Receive Buffer Descriptor.
2	RXB	r/w	1'b0	Receive frame This bit indicates that a frame was received. It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Receive Buffer Descriptor.
1	TXE	r/w	1'b0	Transmit error This bit indicates that a buffer was not transmitted due to a transmit error (underrun, retransmission limit, late collision, bus error or defer time-out). It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Transmit Buffer Descriptor.
0	TXB	r/w	1'b0	Transmit buffer This bit indicates that a buffer has been transmitted. It is cleared by writing 1 to it. This bit appears only when IRQ bit is set in the Transmit Buffer Descriptor.

17.8.3 INT_MASK

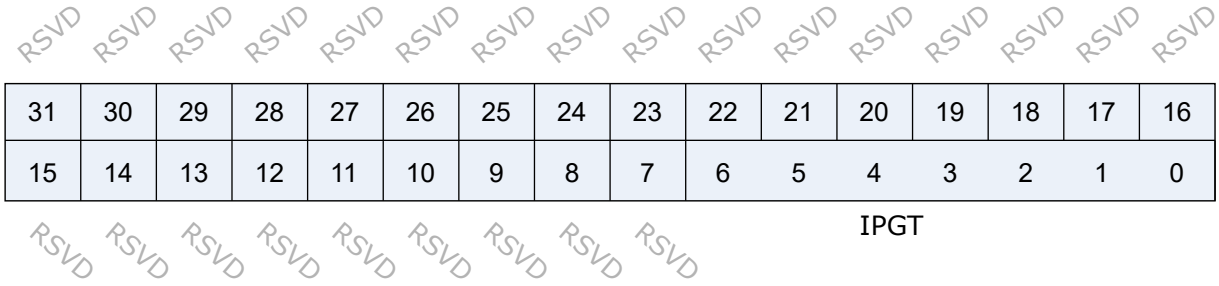
Address：0x20070008



Bits	Name	Type	Reset	Description
31:7	RSVD			
6	RXC_M	r/w	1'b1	Receive control frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
5	TXC_M	r/w	1'b1	Transmit control frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
4	BUSY_M	r/w	1'b1	Busy mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
3	RXE_M	r/w	1'b1	Receive error mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
2	RXB_M	r/w	1'b1	Receive frame mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
1	TXE_M	r/w	1'b1	Transmit error mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked
0	TXB_M	r/w	1'b1	Transmit buffer mask ENABLE 0: Interrupt is un-masked 1: Interrupt is masked

17.8.4 IPGT

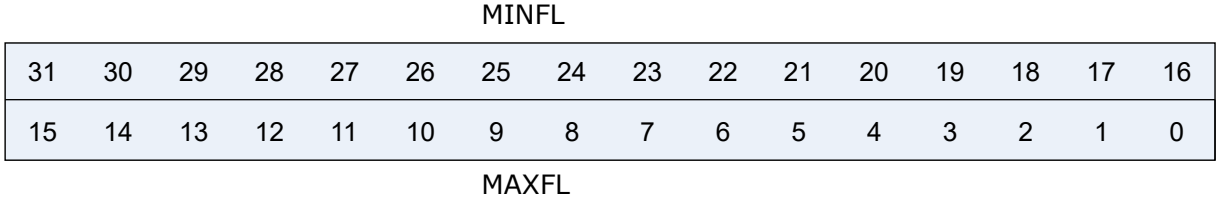
Address：0x2007000c



Bits	Name	Type	Reset	Description
31:7	RSVD			
6:0	IPGT	r/w	7'h18	Inter packet gap The recommended value is 0x18 (24 clock cycles), which equals 0.96 us for 100 Mbps mode

17.8.5 PACKETLEN

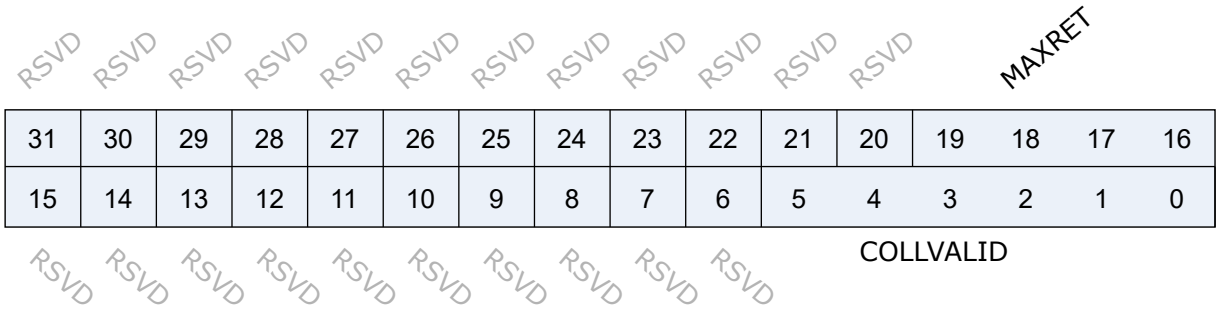
Address：0x20070018



Bits	Name	Type	Reset	Description
31:16	MINFL	r/w	16'h40	Minimum frame length The minimum Ethernet packet is 64 bytes long (0x40). To receive small packets, assert the RECSMALL bit or change the MINFL value. To transmit small packets, assert the PAD bit or change the MINFL value.
15:0	MAXFL	r/w	16'h600	Maximum frame length The maximum Ethernet packet is 1518 bytes long. To support this and to have some additional space for tags, a default maximum packet length equals to 1536 bytes (0x600). For bigger packets, you can assert the HUGEN bit or increase the value of MAXFL field.

17.8.6 COLLCONFIG

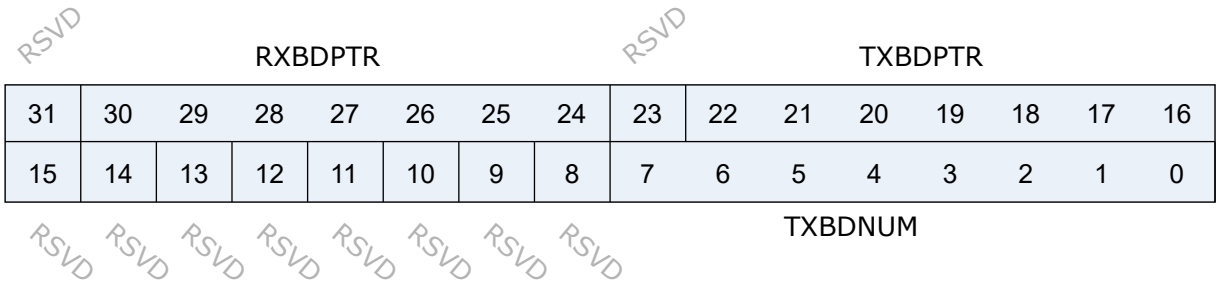
Address：0x2007001c



Bits	Name	Type	Reset	Description
31:20	RSVD			
19:16	MAXRET	r/w	4'hF	Maximum retry This field specifies the maximum number of consequential retransmission attempts after the collision is detected. When the maximum number has been reached, the TX MAC reports an error and stops transmitting the current packet. According to the Ethernet standard, the MAXRET default value is set to 0xf (15).
15:6	RSVD			
5:0	COLLVALID	r/w	6'h3F	Collision valid This field specifies a collision time window. A collision that occurs later than the time window is reported as a "Late Collisions" and transmission of the current packet is aborted.

17.8.7 TX_BD_NUM

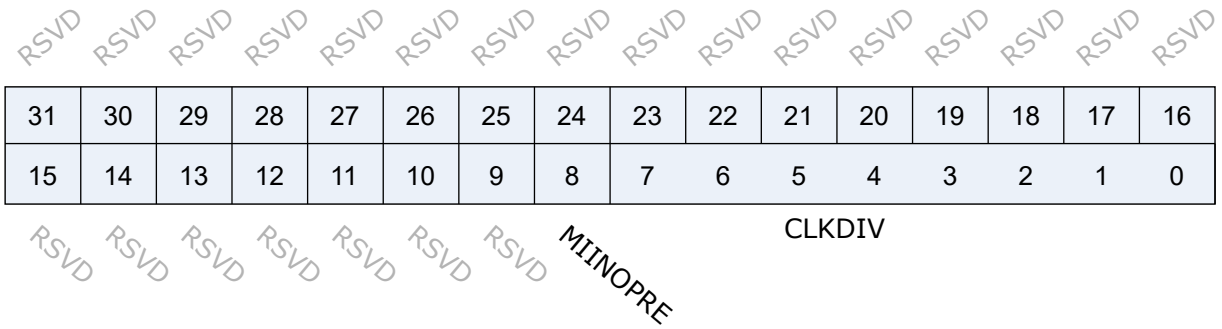
Address：0x20070020



Bits	Name	Type	Reset	Description
31	RSVD			
30:24	RXBDPTR	r	7'h0	RX buffer descriptors (BD) pointer, pointing at the RXBD currently being used
23	RSVD			
22:16	TXBDPTR	r	7'h0	TX buffer descriptors (BD) pointer, pointing at the TXBD currently being used
15:8	RSVD			
7:0	TXBDNUM	r/w	8'h40	TX buffer descriptors (BD) number Number of TX BD. TX and RX share 128 (0x80) descriptors, so the number of RX BD equals 0x80 - TXBDNUM. The maximum number of TXBDNUM is 0x80. Values greater then 0x80 cannot be written into this register.

17.8.8 MIIMODE

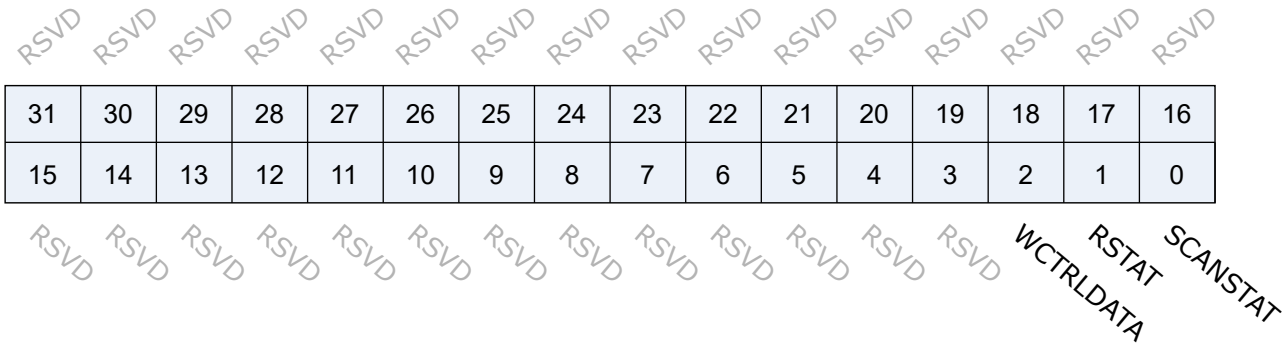
Address：0x20070028



Bits	Name	Type	Reset	Description
31:9	RSVD			
8	MIINOPRE	r/w	1'b0	No preamble for Management Data (MD) 0: 32-bit preamble will be sent. 1: No preamble will be sent.
7:0	CLKDIV	r/w	8'h64	Clock divider for Management Data Clock (MDC) The source clock is bus clock and can be divided by any even number.

17.8.9 MIICOMMAND

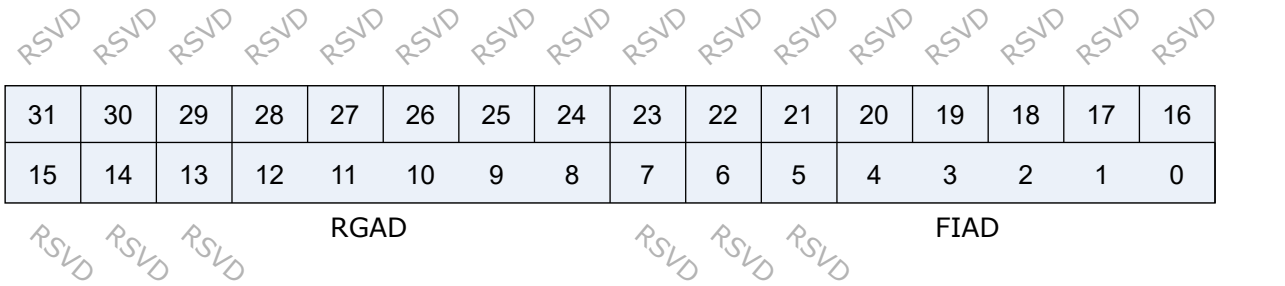
Address：0x2007002c



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	WCTRLDATA	r/w	1'b0	Write control data, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time
1	RSTAT	r/w	1'b0	Read status, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time
0	SCANSTAT	r/w	1'b0	Scan status, setting this bit to 1 will trigger the command (auto cleared) Note: [2]/[1]/[0] cannot be asserted at the same time, execute one command at a time

17.8.10 MIIADDRESS

Address：0x20070030



Bits	Name	Type	Reset	Description
31:13	RSVD			
12:8	RGAD	r/w	5'h0	Register Address
7:5	RSVD			
4:0	FIAD	r/w	5'h0	PHY Address

17.8.11 MIITX_DATA

Address：0x20070034

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRLDATA															

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	CTRLDATA	r/w	16'h0	Control Data to be written to PHY

17.8.12 MIIRX_DATA

Address：0x20070038

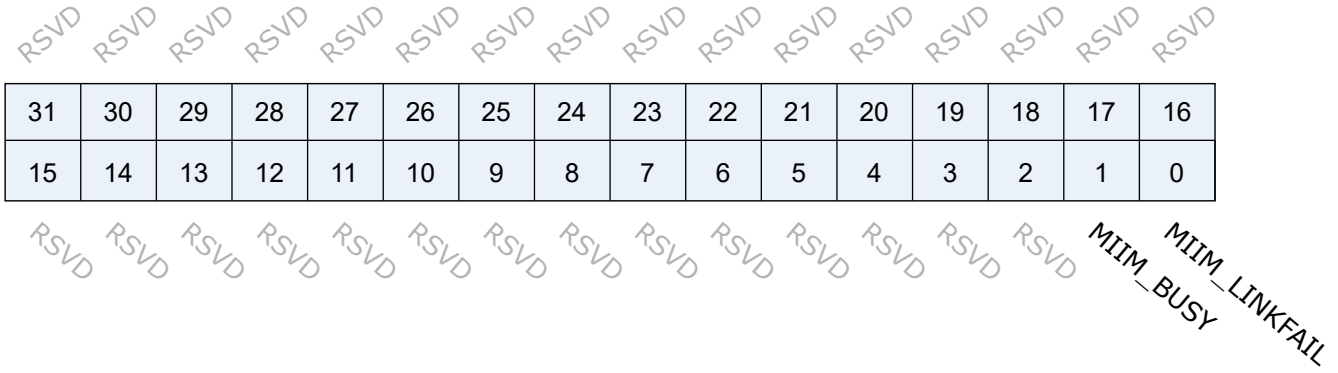
RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRSD															

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	PRSD	r	16'h0	Received Data from PHY

17.8.13 MIISTATUS

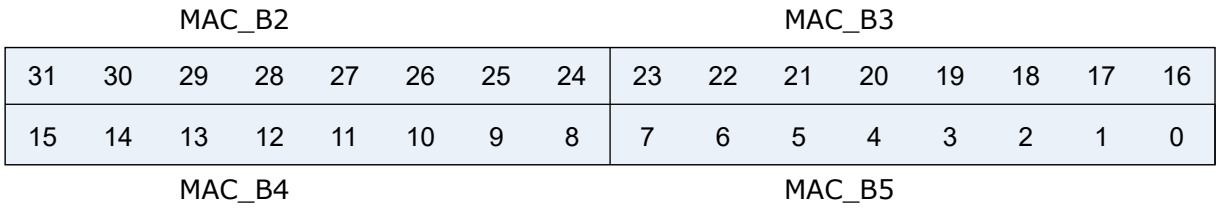
Address：0x2007003c



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	MIIM_BUSY	r	1'b0	MIIM I/F busy signal 0: The MIIM I/F is ready. 1: The MIIM I/F is busy.
0	MIIM_LINKFAIL	r	1'b0	MIIM I/F link fail signal

17.8.14 MAC_ADDR0

Address：0x20070040



Bits	Name	Type	Reset	Description
31:24	MAC_B2	r/w	8'd0	Ethernet MAC address byte 2
23:16	MAC_B3	r/w	8'd0	Ethernet MAC address byte 3
15:8	MAC_B4	r/w	8'd0	Ethernet MAC address byte 4
7:0	MAC_B5	r/w	8'd0	Ethernet MAC address byte 5

17.8.15 MAC_ADDR1

Address：0x20070044

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAC_B0								MAC_B1							

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:8	MAC_B0	r/w	8'd0	Ethernet MAC address byte 0
7:0	MAC_B1	r/w	8'd0	Ethernet MAC address byte 1

17.8.16 HASH0_ADDR

Address：0x20070048

HASH0															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HASH0															

Bits	Name	Type	Reset	Description
31:0	HASH0	r/w	32'h0	Lower 32-bit of HASH register

17.8.17 HASH1_ADDR

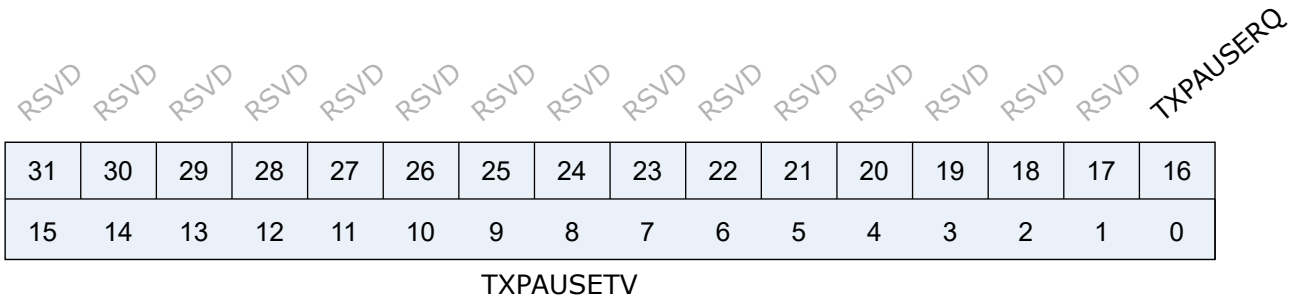
Address：0x2007004c

HASH1															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HASH1															

Bits	Name	Type	Reset	Description
31:0	HASH1	r/w	32'h0	Upper 32-bit of HASH register

17.8.18 TXCTRL

Address：0x20070050



Bits	Name	Type	Reset	Description
31:17	RSVD			
16	TXPAUSERQ	r/w	1'b0	TX Pause Request Writing 1 to this bit starts sending control frame and is automatically cleared to zero.
15:0	TXPAUSETV	r/w	16'h0	TX Pause Timer Value The value that is sent in the pause control frame.

18 USB

18.1 Overview

19 CAM

19.1 Overview

The CAM (Camera) module is responsible for converting the parallel interface (DVP) into a general bus interface (AHB), and writing the pixel data generated by the image sensor into the system memory for subsequent image transmission or compression. The CAM module has a flexible output format configuration, which can meet a variety of image processing needs.

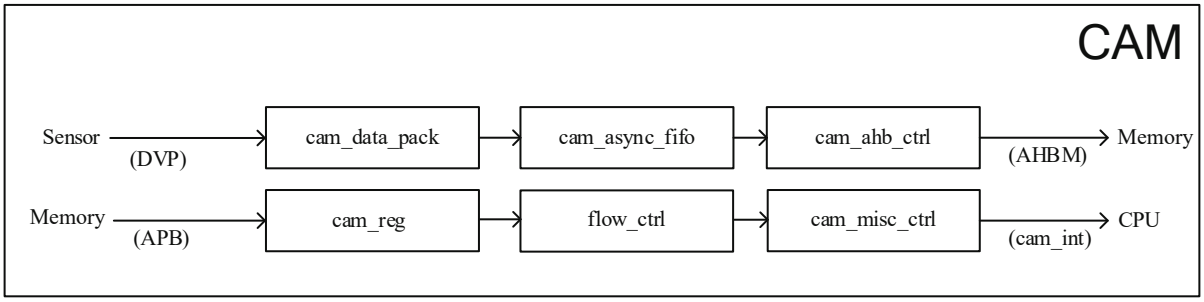


Fig. 19.1: CAM block diagram

19.2 Features

- Parallel interface 8-bit DVP signal, high-speed data transmission (80M), configurable DVP signal effective level and logic combination
- Support 8-bit/16-bit/24-bit input pixel width
- Support converting RGB888 input format to RGB565 or RGBA8888 output
- Support movie mode and photo mode
- Configurable drop patterns including:
  - Discard odd-digit data
  - Discard even-digit data

  - Discard odd-numbered data in odd-numbered rows
  - Discard even-digit data of odd-numbered rows
- Configurable image sensor line frame sync signal selection and polarity selection
- Support image rectangle cropping
- Frame selection function with a cycle of 1~32
- Support integrity detection of line frame synchronization signal
- AHB bus communication interface
- 512-byte buffer FIFO for occasional busy bus status
- Continuously cache up to 4 sets of image information
- A variety of application interruptions are conducive to flexible use and error prompts

19.3 Function description

19.3.1 DVP (Digital Video Port) signal and configuration

DVP (Digital Video Port) is a parallel interface, mainly including clock, frame synchronization, line synchronization and 8-bit data pins. The limit of the clock is limited to 80MHz, so it is generally used for sensors with resolutions below 5MP. The effective level of frame synchronization and line synchronization can be independently configured in the chip, and four modes are provided on the effective data:

A. Frame sync and line sync are active at the same time (”&” logic)

B. Either frame sync or line sync is valid (“|” logic)

C. Frame synchronization is valid

D. Line synchronization is valid

19.3.2 YCbCr format

The luminance signal is called Y, and the chrominance signal is composed of two independent signals. Depending on the color system and format, the two chrominance signals are often referred to as U, V or Pb, Pr or Cb, Cr. This results from different encoding formats, but in fact they have basically the same concept.

Since there are more retinal rod cells that recognize brightness than retinal cone cells that recognize chrominance on the human retina, the human eye is more sensitive to brightness than to chrominance. Therefore, part of the chrominance information can be discarded without being perceived by the human eye.

The format with the highest resolution of the chrominance signal is 4:4:4, that is, every 4-point Y sampling corresponds to 4-point Cb and 4-point Cr sampling. And 4:2:2 is that every 4 points of Y sampling corresponds to 2 points of Cb and 2 points of Cr sampling. In this format, the number of scan lines for chrominance signals is as many as that for

luminance signals, but the color of each scan line is The degree sample points are only half of the luminance signal. Different from the format mentioned above, 4:2:0 does not correspond to 2 points of Cb and 0 points of Cr sampling for every 4 points of Y sampling, but corresponds to 1 point of Cb and 1 point of Cr sampling for every 4 points of Y sampling. 4:0:0 is to discard all chrominance information, that is, the grayscale image.

19.3.3 Movie Mode/Photo Mode

In the photo mode, when the fixed-size memory given by the software is full, the CAM will stop, and the software will need to perform a POP operation to free up the space before continuing to write.

In video mode, it will keep rewriting on the fixed-size memory provided by the software, that is, using the memory as a ring buffer, without software for POP operation, to ensure that the image is taken out in real time or linked with the MJPEG module.

19.3.4 RGB888 to RGB565/RGBA8888 output

For image data whose input format is RGB888, you can choose to convert it to RGB565 or RGBA8888 and write it to the memory. If it is converted to RGB565 format, the arrangement order of R, G, B can be controlled by the bit FORMAT_565 of the register MISC. The arrangement order corresponding to different values is as follows:

- 0: byte2[7:3],byte1[7:2],byte0[7:3]
- 1: byte1[7:3],byte2[7:2],byte0[7:3]
- 2: byte2[7:3],byte0[7:2],byte1[7:3]
- 3: byte0[7:3],byte2[7:2],byte1[7:3]
- 4: byte1[7:3],byte0[7:2],byte2[7:3]
- 5: byte0[7:3],byte1[7:2],byte2[7:3]

If it is converted to RGBA8888 format, the value of A is the same as the value filled in bit ALPHA of the register MISC.

19.3.5 Image rectangle crop

By setting the start and end positions of the line synchronization signal and the frame synchronization signal cropping by the high 16 bits and the low 16 bits of the registers HSYNC_CONTROL and VSYNC_CONTROL, the image in the rectangular window of the specified position and size can be cropped, and the image beyond the rectangular part can be cropped. Data will be discarded. The start and end of the line synchronization signal are set to the pixel serial number, the start and end of the frame synchronization signal are set to the line number, and the cropped image contains the start point but not the end point.

19.3.6 Frame rounding function

A frame period n can be set through the register FRAME_PERIOD, the value range of n is 0~31, and the corresponding actual value is n+1, and then the register FRAME_VLD is used to set which frames of images are retained in a frame period. Write 1 in the corresponding bit position if you need to keep it, and write 0 in the corresponding bit position if you need to discard it. For example, if n is set to 5 and the value of FRAME_VLD is set to 0x13, in every 6 frames of images, the 1st, 2nd, and 5th frames will be written into the memory, and the 3rd, 4th, and 6th frames will be discarded. The cycle is performed with 6 frames of images as a cycle.

19.3.7 Line Frame Sync Signal Integrity Detection

The line synchronization signal comparison value and the frame synchronization signal comparison value can be set respectively through the lower 16 bits and the upper 16 bits of the register FRAME_SIZE_CONTROL, and the integrity of the signal can be detected. The line sync signal sets the total number of pixels in each row, and the frame sync signal sets the total number of lines. When the count value of the line or frame synchronization signal of a frame image is not equal to the comparison value, a corresponding interrupt will be generated.

19.3.8 Cache image information

The module contains 4 groups of FIFO to record the image address and image ID. Whenever this module completely writes a frame to the memory, it will record the start address and image ID of the frame image in this FIFO, but it should be noted that when the memory is insufficient, or the 4 sets of FIFO are full. In this case, the module will automatically discard the information of the next image. In the part where the image information is taken out, the oldest image information can be emptied by the pop operation. At this time, the FIFO will automatically advance to ensure the timing of the image information in the FIFO, as shown in the figure below:

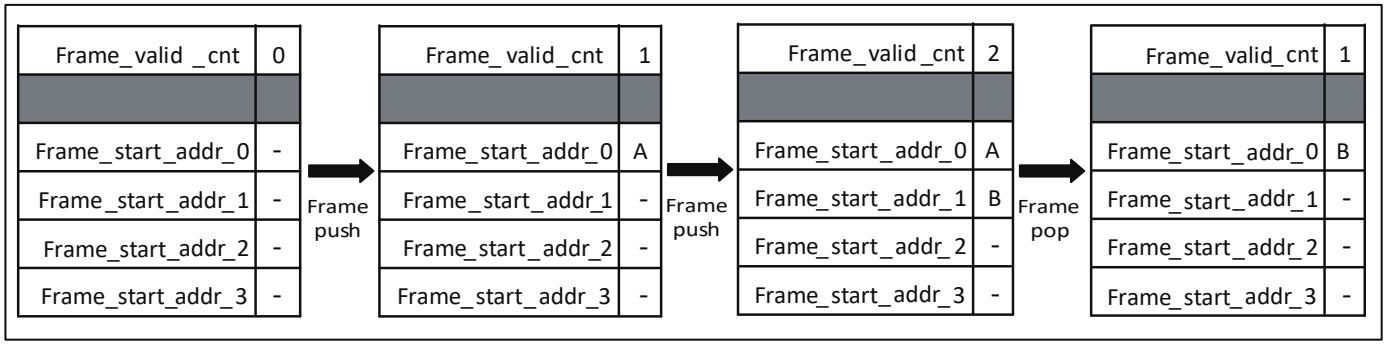


Fig. 19.2: FIFO framework

19.3.9 Support a variety of interrupt information (can be configured independently of the switch)

- Normal interrupt * a count value n can be set, and an interrupt will be issued every time n images are written
- Memory Interrupt * When the remaining memory space is less than one frame size, and the used memory is overwritten, an interrupt is issued, as shown in the following figure:

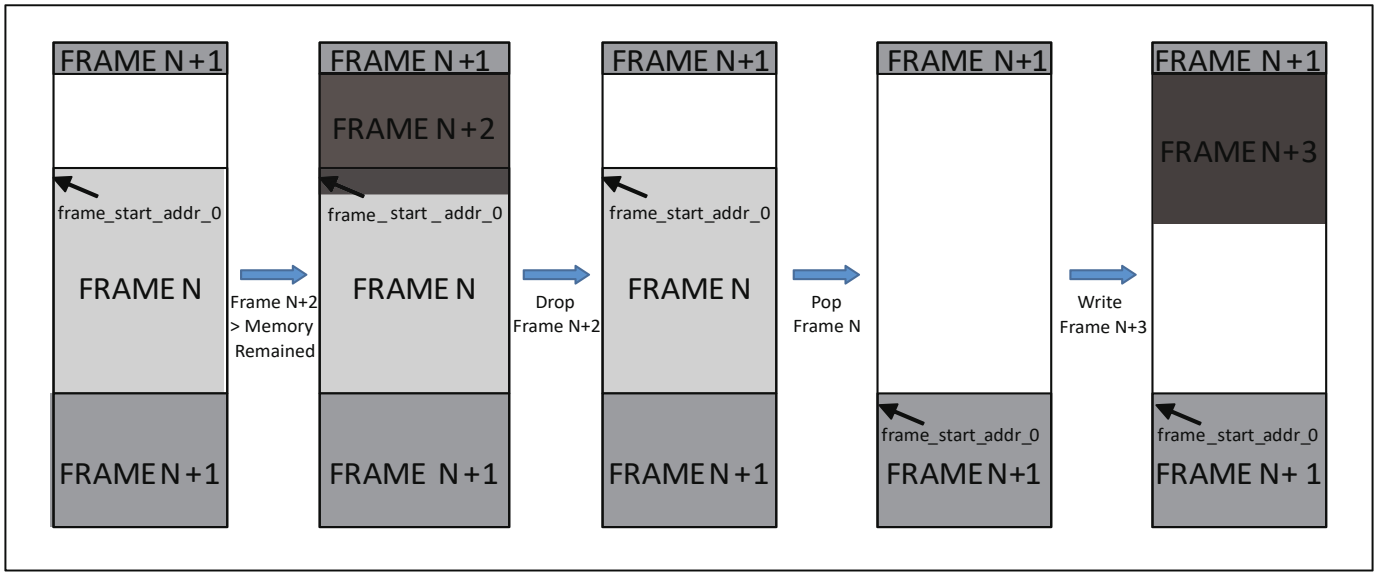


Fig. 19.3: Memory

- Frame interrupt * when there are more than 4 groups of unprocessed images and no more image information can be stored, an interrupt is issued
- FIFO interrupt * when the bus is too late to write to the memory, causing the FIFO to overflow, an interrupt is issued
- Hsync interrupt * when the number of pixels of a line in a frame of image is not equal to the set value (the line sync signal integrity test fails), an interrupt is issued
- Vsync interrupt * when the total number of lines in a frame of image is not equal to the set value (frame sync signal integrity check fails), an interrupt is issued

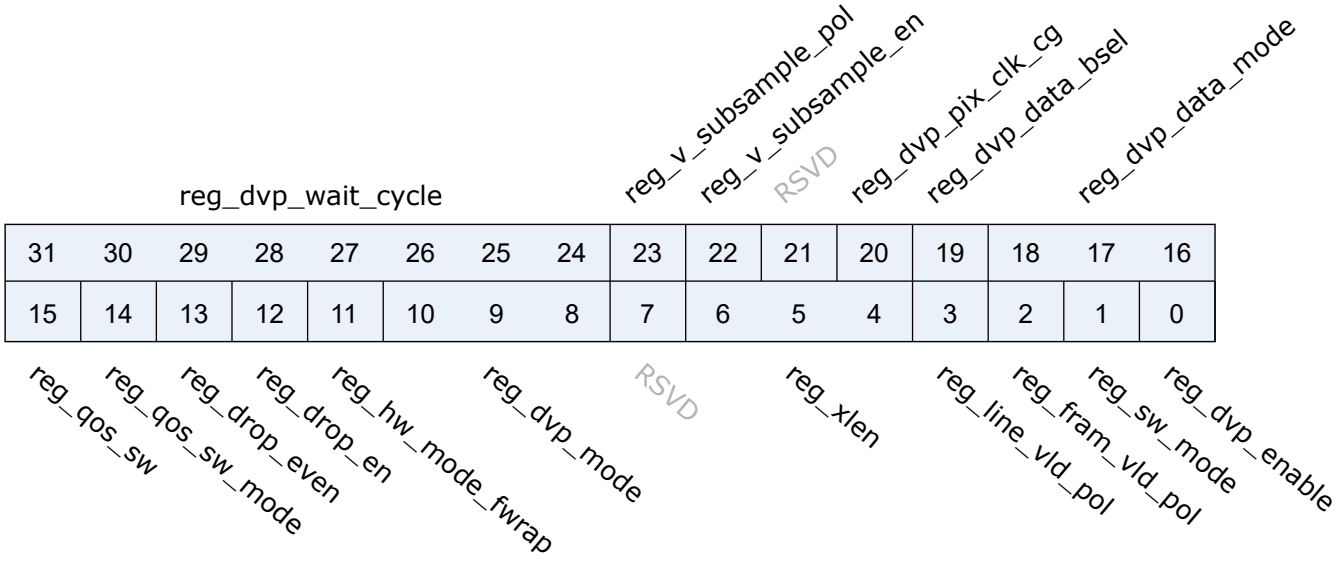
19.4 Register description

Name	Description
cam_dvp2axi_configue	Camera configure
cam_dvp2axi_addr_start	Start of write address
cam_dvp2axi_mem_bcnt	Burst count of memory
cam_dvp_status_and_error	Interrupt and bus status

Name	Description
cam_dvp2axi_frame_bcnt	Byte count of frame
cam_dvp_frame_fifo_pop	Interrupt clear and pop
cam_dvp2axi_frame_vld	Frame valid in period
cam_dvp2axi_frame_period	Frame period
cam_dvp2axi_misc	RGB565 and RGBA8888 setting
cam_dvp2axi_hsync_crop	Hsync crop
cam_dvp2axi_vsync_crop	Vsync crop
cam_dvp2axi_fram_exm	Total valid count
cam_frame_start_addr0	Start address of frame0
cam_frame_start_addr1	Start address of frame1
cam_frame_start_addr2	Start address of frame2
cam_frame_start_addr3	Start address of frame3
cam_frame_id_sts01	ID of frame 0/1
cam_frame_id_sts23	ID of frame 2/3
cam_dvp_debug	ID latch timing

19.4.1 cam_dvp2axi_configue

Address：0x20057000



Bits	Name	Type	Reset	Description
31:24	reg_dvp_wait_cycle	r/w	8'h40	Cycles in FSM Wait mode
23	reg_v_subsample_pol	r/w	1'b0	DVP2BUS vertical sub-sampling polarity 1'b0: Odd lines are masked 1'b1: Even lines are masked
22	reg_v_subsample_en	r/w	1'b0	DVP2BUS vertical sub-sampling enable
21	RSVD			
20	reg_dvp_pix_clk_cg	r/w	1'b0	DVP pix clk gate
19	reg_dvp_data_bsel	r/w	1'b0	Byte select signal for DVP 8-bit mode, don't care if reg_dvp_data_8bit is disabled 1'b0: Select the lower byte of pix_data 1'b1: Select the upper byte of pix_data
18:16	reg_dvp_data_mode	r/w	3'b0	DVP 8-bit mode enable 3'd0: DVP pix_data is 16-bit wide 3'd1: DVP pix_data is 24-bit mode 3'd2: DVP pix_data is 24-comp-16-bit mode 3'd3: DVP pix_data is 24-exp-32-bit mode 3'd4: DVP pix_data is 8-bit wide Others - Reserved
15	reg_qos_sw	r/w	1'b0	AXI Qos software mode value
14	reg_qos_sw_mode	r/w	1'b0	AXI QoS software mode enable
13	reg_drop_even	r/w	1'b0	Only effect when reg_drop_en=1 : 1'b1 : Drop all even bytes 1'b0 : Drop all odd bytes
12	reg_drop_en	r/w	1'b0	Drop mode Enable
11	reg_hw_mode_fwrap	r/w	1'b1	DVP2BUS HW mode with frame start address wrap to reg_addr_start
10:8	reg_dvp_mode	r/w	3'd0	Image senosr mode selection: 3'd0-Vsync&Hsync 3'd1-Vsync|Hsync 3'd2-Vsync 3'd3-Hsync
7	RSVD			
6:4	reg_xlen	r/w	3'd3	burst length setting 3'd0 - Single / 3'd1 - INCR4 / 3'd2 - INCR8 3'd3 - INCR16 / 3'd5 - INCR32 / 3'd6 - INCR64
3	reg_line_vld_pol	r/w	1'b1	Image sensor line valid polarity, 1'b0 - Activel low, 1'b1 - Active high

Bits	Name	Type	Reset	Description
2	reg_fram_vld_pol	r/w	1'b1	Image sensor frame valid polarity, 1'b0 - Activel low, 1'b1 - Active high
1	reg_sw_mode	r/w	1'b0	DVP2BUS SW manual mode (don't care if reg_swap_mode is enabled)
0	reg_dvp_enable	r/w	1'b0	module enable

19.4.2 cam_dvp2axi_addr_start

Address：0x20057004

reg_addr_start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_addr_start

Bits	Name	Type	Reset	Description
31:0	reg_addr_start	r/w	32'h80000000	AXI start address

19.4.3 cam_dvp2axi_mem_bcnt

Address：0x20057008

reg_mem_burst_cnt

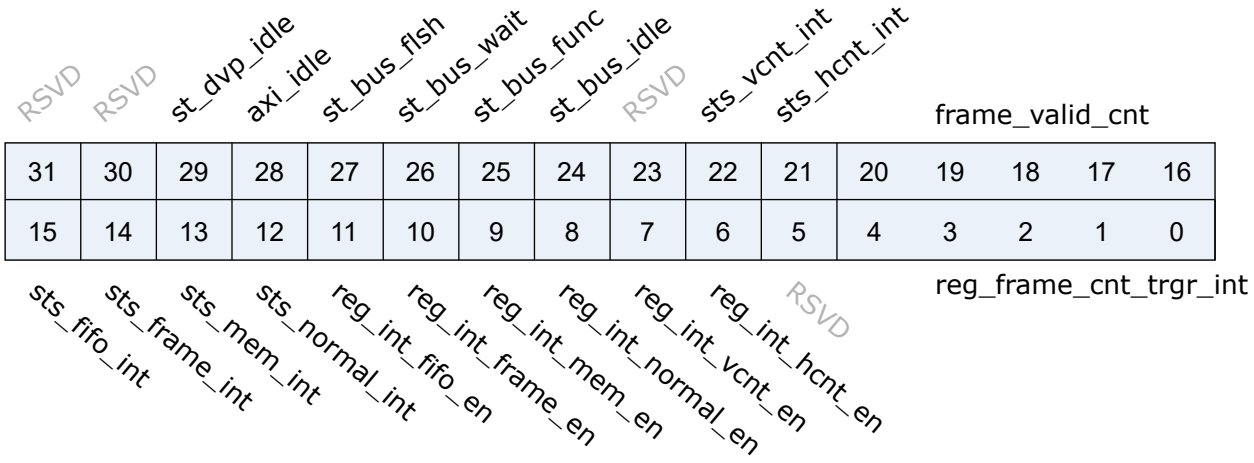
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_mem_burst_cnt

Bits	Name	Type	Reset	Description
31:0	reg_mem_burst_cnt	r/w	32'hC000	AXI burst cnt before wrap to "reg_addr_start"

19.4.4 cam_dvp_status_and_error

Address：0x2005700c



Bits	Name	Type	Reset	Description
31:30	RSVD			
29	st_dvp_idle	r	1'b1	DVP2BUS asynchronous fifo idle status
28	axi_idle	r	1'b1	DVP2BUS AHB idle status
27	st_bus_flsh	r	1'b0	DVP in flush state
26	st_bus_wait	r	1'b0	DVP in wait state
25	st_bus_func	r	1'b0	DVP in functional state
24	st_bus_idle	r	1'b1	DVP in idle state
23	RSVD			
22	sts_vcnt_int	r	1'b0	Vsync valid line count non-match interrupt status
21	sts_hcnt_int	r	1'b0	Hsync valid pixel count non-match interrupt status
20:16	frame_valid_cnt	r	5'd0	Frame counts in memory before read out in SW mode
15	sts_fifo_int	r	1'b0	FIFO OverWrite interrupt status
14	sts_frame_int	r	1'b0	Frame OverWrite interrupt status
13	sts_mem_int	r	1'b0	Memory OverWrite interrupt status
12	sts_normal_int	r	1'b0	Normal Write interrupt status
11	reg_int_fifo_en	r/w	1'b1	FIFO OverWrite interrupt enable
10	reg_int_frame_en	r/w	1'b0	Frame OverWrite interrupt enable
9	reg_int_mem_en	r/w	1'b0	Memory OverWrite interrupt enable
8	reg_int_normal_en	r/w	1'b0	Normal Write interrupt enable
7	reg_int_vcnt_en	r/w	1'b0	Vsync valid line count match interrupt enable

Bits	Name	Type	Reset	Description
6	reg_int_hcnt_en	r/w	1'b0	Hsync valid pixel count match interrupt enable
5	RSVD			
4:0	reg_frame_cnt_trgr_int	r/w	5'd0	Frame to issue interrupt at SW Mode

19.4.5 cam_dvp2axi_frame_bcnt

Address：0x20057010

reg_frame_byte_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_frame_byte_cnt

Bits	Name	Type	Reset	Description
31:0	reg_frame_byte_cnt	r/w	32'h7e90	Single Frame byte cnt(Need pre-calculation)

19.4.6 cam_dvp_frame_fifo_pop

Address：0x20057014

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	reg_int_vcnt_clr	reg_int_hcnt_clr	reg_int_fifo_clr	reg_int_frame_clr	reg_int_mem_clr	reg_int_normal_clr	RSVD	RSVD	RSVD	rfifo_pop

Bits	Name	Type	Reset	Description
31:10	RSVD			
9	reg_int_vcnt_clr	w1p	1'd0	Interrupt clear
8	reg_int_hcnt_clr	w1p	1'd0	Interrupt clear
7	reg_int_fifo_clr	w1p	1'd0	Interrupt clear

Bits	Name	Type	Reset	Description
6	reg_int_frame_clr	w1p	1'd0	Interrupt clear
5	reg_int_mem_clr	w1p	1'd0	Interrupt clear
4	reg_int_normal_clr	w1p	1'd0	Interrupt clear
3:1	RSVD			
0	rfifo_pop	w1p	1'b0	Write this bit will trigger fifo pop

19.4.7 cam_dvp2axi_frame_vld

Address：0x20057018

reg_frame_n_vld

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_frame_n_vld

Bits	Name	Type	Reset	Description
31:0	reg_frame_n_vld	r/w	32'hffff_-ffff	Bitwise frame valid in period

19.4.8 cam_dvp2axi_frame_period

Address：0x2005701c

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

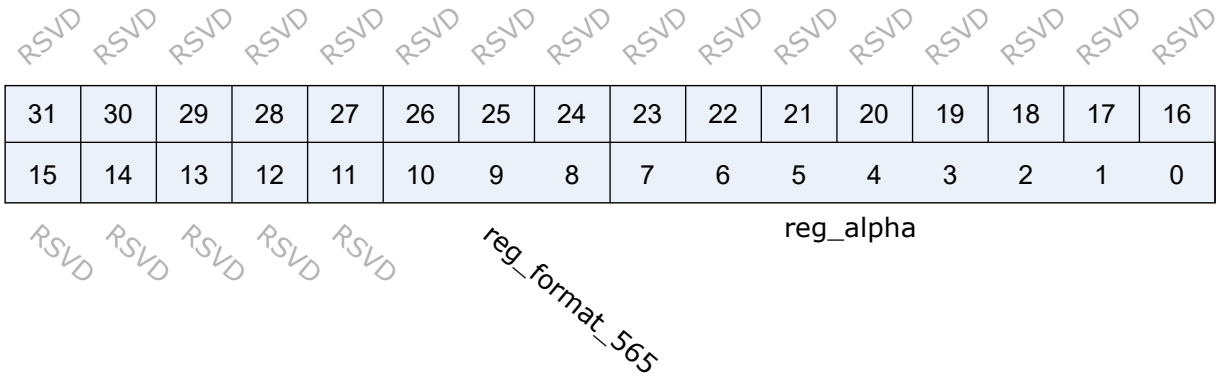
RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

reg_frame_period

Bits	Name	Type	Reset	Description
31:5	RSVD			
4:0	reg_frame_period	r/w	5'h0	Frame period cnt. (EX. Set this register 0, the period is 1)

19.4.9 cam_dvp2axi_misc

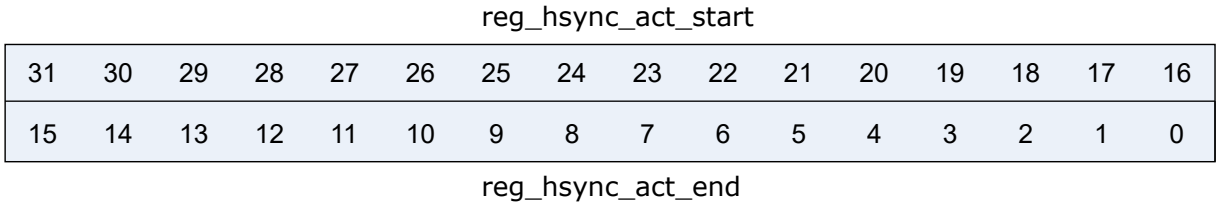
Address：0x20057020



Bits	Name	Type	Reset	Description
31:11	RSVD			
10:8	reg_format_565	r/w	3'd0	Only work when reg_dvp_data_mode=2 (24-comp-16-bit mode) 3'd0: B2(5)B1(6)B0(5) 3'd1: B1(5)B2(6)B0(5) 3'd2: B2(5)B0(6)B1(5) 3'd3: B0(5)B2(6)B1(5) 3'd4: B1(5)B0(6)B2(5) 3'd5: B0(5)B1(6)B2(5)
7:0	reg_alpha	r/w	8'h0	Only work when "reg_dvp_data_mode==2'd3(DVP pix_-data is 24-exp-32-bit mode)" The value of [31:24]

19.4.10 cam_dvp2axi_hsync_crop

Address：0x20057030



Bits	Name	Type	Reset	Description
31:16	reg_hsync_act_start	r/w	16'h0	Valid hsync start cnt
15:0	reg_hsync_act_end	r/w	16'hFFFF	Valid hsync end cnt

19.4.11 cam_dvp2axi_vsync_crop

Address：0x20057034

reg_vsync_act_start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_vsync_act_end

Bits	Name	Type	Reset	Description
31:16	reg_vsync_act_start	r/w	16'h0	Valid vsync start cnt
15:0	reg_vsync_act_end	r/w	16'hFFFF	Valid vsync end cnt

19.4.12 cam_dvp2axi_fram_exm

Address：0x20057038

reg_total_vcnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_total_hcnt

Bits	Name	Type	Reset	Description
31:16	reg_total_vcnt	r/w	16'h0	Total valid line count in a frame
15:0	reg_total_hcnt	r/w	16'h0	Total valid pix count in a line

19.4.13 cam_frame_start_addr0

Address：0x20057040

frame_start_addr_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_0

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_0	r	32'd0	DVP2BUS PIC 0 Start address

19.4.14 cam_frame_start_addr1

Address：0x20057048

frame_start_addr_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_1

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_1	r	32'd0	DVP2BUS PIC 1 Start address

19.4.15 cam_frame_start_addr2

Address：0x20057050

frame_start_addr_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_2

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_2	r	32'd0	DVP2BUS PIC 2 Start address

19.4.16 cam_frame_start_addr3

Address：0x20057058

frame_start_addr_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_3

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_3	r	32'd0	DVP2BUS PIC 3 Start address

19.4.17 cam_frame_id_sts01

Address：0x20057060

frame_id_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_id_0

Bits	Name	Type	Reset	Description
31:16	frame_id_1	r	16'd0	DVP2BUS PIC 1 ID
15:0	frame_id_0	r	16'd0	DVP2BUS PIC 0 ID

19.4.18 cam_frame_id_sts23

Address：0x20057064

frame_id_3

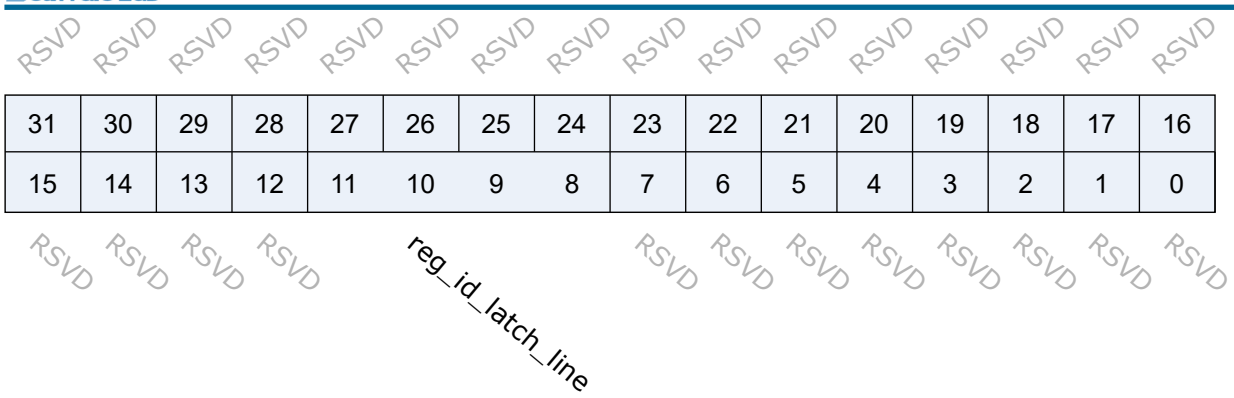
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_id_2

Bits	Name	Type	Reset	Description
31:16	frame_id_3	r	16'd0	DVP2BUS PIC 3 ID
15:0	frame_id_2	r	16'd0	DVP2BUS PIC 2 ID

19.4.19 cam_dvp_debug

Address：0x200570f0



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	reg_id_latch_line	r/w	4'd5	ID latch timing (line count)
7:32	RSVD			
31:0	RESERVED	rsvd	32'hf0f0f0f0	RESERVED

20 MJPEG

20.1 Overview

MJPEG (Motion Joint Photographic Experts Group) is a video coding format that can be accurate to frame editing and multi-layer image processing. It handles motion video sequences as continuous still images. This compression method compresses each frame individually and completely. . By compressing the original data in YCbCr format, the memory space occupied by one frame of image can be greatly reduced.

20.2 Features

- Configurable input formats including:
  - YCbCr4:2:2 plane or packed mode
  - YCbCr4:2:0 plane mode
  - YCbCr4:0:0
- Configurable input data Y, Cb, Cr sort order
- Quantization coefficient table can be freely configured
- Support software mode and linkage mode
- Support swap mode
- Support kick mode
- Reserve jpg head space and automatically add jpg tail
- Continuously cache up to 4 sets of picture information
- A variety of application interruptions are conducive to flexible use and error prompts

20.3 Function description

20.3.1 Input configuration

The YCbCr format of the input data can be selected by bit <REG_YUV_MODE> of register MJPEG_CONTROL_1, including YCbCr4:2:2 plane or packed mode, YCbCr4:2:0 plane mode and YCbCr4:0:0 grayscale image. When the YCbCr4:2:2 packing mode is selected, the arrangement order of Y, Cb and Cr can be configured through the upper 8 bits of the register MJPEG_HEADER_BYTE. When other modes are selected, the order of Cb and Cr can be set by bit <REG_ORDER_U_EVEN> of register MJPEG_CONTROL_1. The detailed configuration is shown in the figure below.

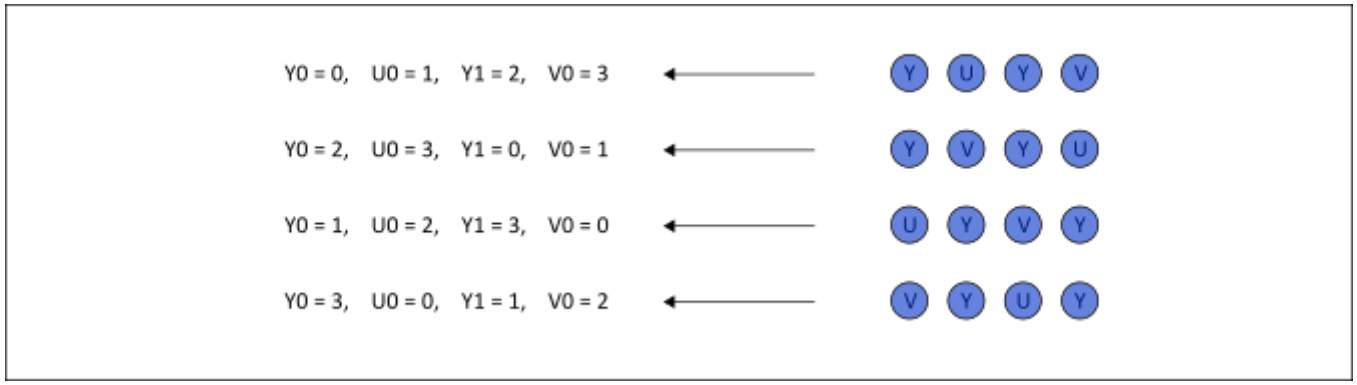


Fig. 20.1: MJPEG YUYV interleave order configuration

20.3.2 Quantization coefficient table

The quantization coefficient table can be freely configured by the user, <reg_q_0_00> to <reg_q_0_3F> represent the quantization tables of the Y component of grayscale information, and <reg_q_1_00> to <reg_q_1_3F> represent the quantization tables of the chrominance information Cb and Cr. The quantization table is arranged in the order from the upper left corner to the bottom, the first column is followed by the second column, that is, reg_q_0_00 represents the quantization value of the first row and the first column of the Y component, and reg_q_0_01 represents the quantization of the second row and the first column of the Y component. Value, reg_q_0_07 represents the quantized value of the Y component in the eighth row and first column, reg_q_0_08 represents the quantized value in the first row and second column of the Y component, and so on.

20.3.3 Software mode and link mode

When the bit <REG_MJPEG_SW_MODE> of the MJPEG_CONTROL_2 register is set to 1, the software mode is enabled. In this mode, MJPEG will read the original image data of the specified number of frames from the memory for compression. In this mode, the image data needs to be prepared in advance.

When the bit <REG_MJPEG_SW_MODE> of the MJPEG_CONTROL_2 register is cleared to 0, the linkage mode is enabled. In this mode, MJPEG will process the output of the CAM module as its input. MJPEG is processed with 8*8 data blocks as a unit. When the CAM finishes writing 8 lines of data to the memory, MJPEG will start to work, and the memory space processed by MJPEG will be released to the CAM for reuse, so that the CAM can complete the

linkage with MJPEG without the memory space of a whole picture.

20.3.4 Swap mode

When using swap mode, the MJPEG storage space will be evenly divided into two blocks, when MJPEG finishes writing one block, an interrupt will be generated to notify the software to read the data, and it will write the data to the other block. Alternate use back and forth, so as to use less than one frame of picture storage space for data processing.

20.3.5 Kick mode

The kick mode is enabled when bit <REG_SW_KICK_MODE> of the MJPEG_CONTROL_2 register is set to 1. In this mode, each time a 1 is written to the bit <REG_SW_KICK> of the MJPEG_CONTROL_2 register, a compression is started. The number of compressed lines is determined by the bit <REG_SW_KICK_HBLK> of the MJPEG_YUV_-MEM_SW register. It should be noted that the kick mode can only be used in software mode, not in linkage mode.

20.3.6 Jpg function

The jpg function will automatically reserve a certain number of bytes of space at the beginning of each frame of data and fill it with the specified data. The size of this space is set by the lower 12 bits of the register MJPEG_HEADER_-BYTE. There is 768 bytes of memory at offset 0x800 for storing the data that needs to be filled at the beginning of each frame. The user only needs to write the jpg header information into it, and this information will be included at the beginning of each frame of data generated. In addition, if the auto-fill jpg end is enabled, two bytes of data will be added at the end of each frame of data, and the value of these two bytes is 0xFFD9.

20.3.7 Cache image information

The module contains 4 groups of FIFO to record the picture address, picture size and picture ID. Whenever this module completely writes a frame to the memory, it will record the start address, image size and image ID of the frame image in this FIFO, but it should be noted that when the memory is insufficient, or there are 4 sets of FIFO When the storage is full, the module will automatically discard the information of the next picture. In the part where the picture information is taken out, the pop action can be performed through the APB interface to empty the oldest picture information. At this time, the FIFO will be automatically advanced to ensure the FIFO. Timing of internal picture information.

20.3.8 Support a variety of interrupt information (can be configured independently of the switch)

A. Normal interrupt - can be set to interrupt after writing several pictures

B. Camera Interrupt - When the speed of MJPEG processing cannot keep up with the speed of CAM module writing, and the CAM overflows, an interrupt is issued

C. Memory interrupt - when the memory is overwritten, an interrupt is issued

D. Frame interrupt - when there are more than 4 groups of unprocessed pictures, an interrupt is issued

E. Swap interrupt - when a memory block is full in swap mode, an interrupt is issued

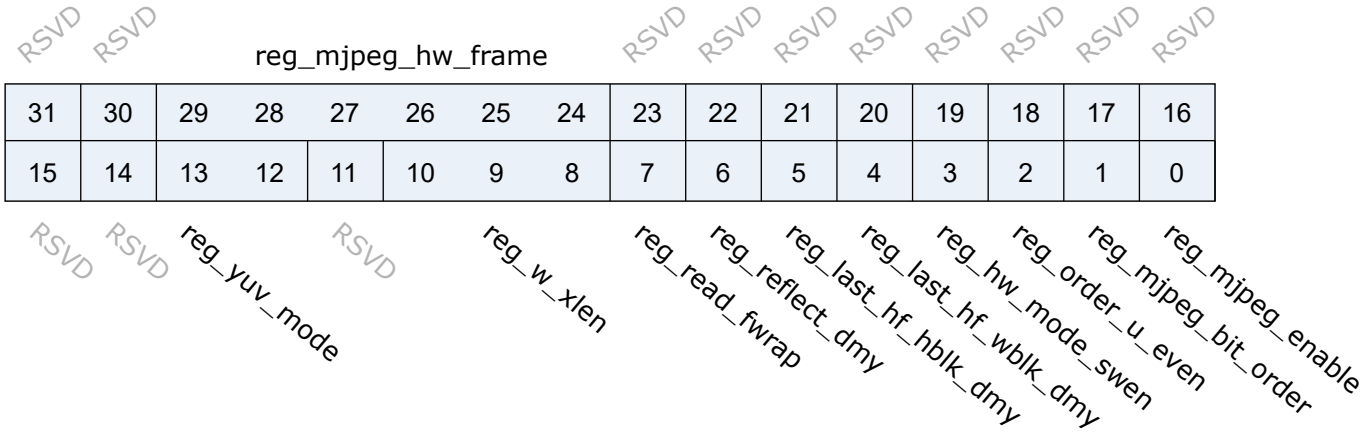
20.4 Register description

Name	Description
mjpeg_control_1	MJPEG configure
mjpeg_control_2	MJPEG software mode
mjpeg_yy_frame_addr	Memory start address of Y
mjpeg_uv_frame_addr	Memory start address of UV
mjpeg_yuv_mem	Number of Y/UV block line
jpeg_frame_addr	Memory start address of JPEG
jpeg_store_memory	Burst count of memory
mjpeg_control_3	Interrupt and bus status
mjpeg_frame_fifo_pop	Interrupt clear and pop
mjpeg_frame_size	Total block count
mjpeg_header_byte	YUV order and auto fill
mjpeg_swap_mode	Swap mode
mjpeg_swap_bit_cnt	Remain bit count
mjpeg_yuv_mem_sw	Number of kick block line
mjpeg_Y_frame_read_status_1	Y read status1
mjpeg_Y_frame_read_status_2	Y read status2
mjpeg_Y_frame_write_status	Y write status
mjpeg_UV_frame_read_status_1	UV read status1
mjpeg_UV_frame_read_status_2	UV read status2
mjpeg_UV_frame_write_status	UV write status
mjpeg_frame_w_hblk_status	Vertical block write status
mjpeg_start_addr0	Start address of frame0
mjpeg_bit_cnt0	Bit count of frame0

Name	Description
mjpeg_start_addr1	Start address of frame1
mjpeg_bit_cnt1	Bit count of frame1
mjpeg_start_addr2	Start address of frame2
mjpeg_bit_cnt2	Bit count of frame2
mjpeg_start_addr3	Start address of frame3
mjpeg_bit_cnt3	Bit count of frame3
mjpeg_q_enc	Quantize SRAM setting
mjpeg_frame_id_10	ID of frame 0/1
mjpeg_frame_id_32	ID of frame 2/3
mjpeg_debug	ID latch timing

20.4.1 mjpeg_control_1

Address：0x30021000

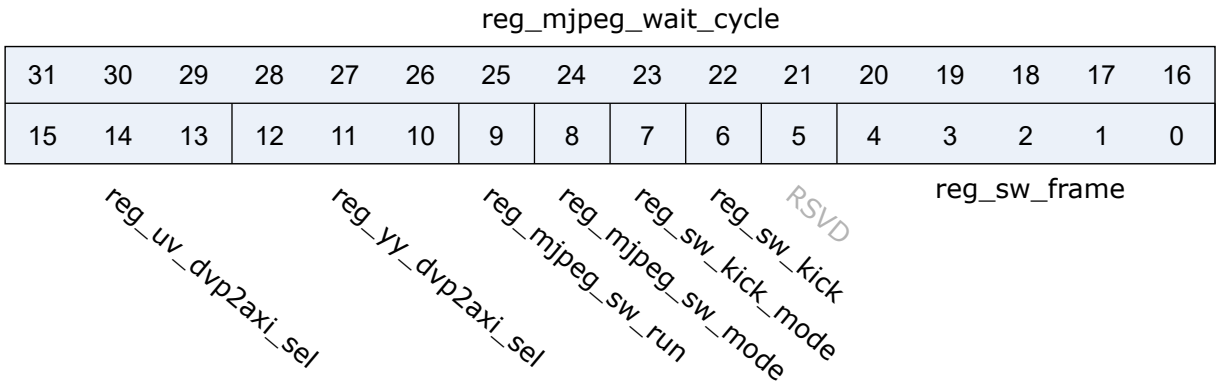


Bits	Name	Type	Reset	Description
31:30	RSVD			
29:24	reg_mjpeg_hw_frame	r/w	6'd0	Non-SW mode frame cnt 0 - Non auto stop
23:14	RSVD			

Bits	Name	Type	Reset	Description
13:12	reg_yuv_mode	r/w	2'd0	YUV format setting 2'd0 : YUV420 Planar 2'd1 : YUV400 grey scale 2'd2 : YUV422 Planar 2'd3 : INTV422
11	RSVD			
10:8	reg_w_xlen	r/w	3'd3	AXI burst length setting 3'd0: Single 3'd1: INCR4 3'd2: INCR8 3'd3: INCR16 Others: RSVD
7	reg_read_fwrap	r/w	1'b1	Only effect in non SW mode This bit determine whether YUV frame read start @ start address
6	reg_reflect_dmy	r/w	1'b0	UV dummy with relect
5	reg_last_hf_hblk_dmy	r/w	1'b0	MJPEG last half vertical block with dummy data 8'h80
4	reg_last_hf_wblk_dmy	r/w	1'b0	MJPEG last half horizational block with dummy data 8'h80
3	reg_hw_mode_swen	r/w	1'b0	Only word In hardware mode. SW enable immediately to skip waiting first frame done
2	reg_order_u_even	r/w	1'b1	1'b0: U is odd byte of UV frame / V is even byte of UV frame 1'b1: U is even byte of UV frame / V is odd byte of UV frame
1	reg_mjpeg_bit_order	r/w	1'b1	MJPEG bitstream byte order adjustment
0	reg_mjpeg_enable	r/w	1'b0	MJPEG enable

20.4.2 mjpeg_control_2

Address：0x30021004



Bits	Name	Type	Reset	Description
31:16	reg_mjpeg_wait_cycle	r/w	16'h100	Cycle count in wait state
15:13	reg_uv_dvp2axi_sel	r/w	3'd1	DVP2AXI selection for UV frame(Don't care if using SW mode)
12:10	reg_yy_dvp2axi_sel	r/w	3'd0	DVP2AXI selection for Y frame(Don't care if using SW mode)
9	reg_mjpeg_sw_run	r/w	1'b0	MJPEG SW mode run (should enable reg_mjpeg_sw_-mode before reg_mjpeg_sw_run)
8	reg_mjpeg_sw_mode	r/w	1'b0	MJPEG SW mode enable (should enable reg_mjpeg_sw_-mode before reg_mjpeg_sw_run)
7	reg_sw_kick_mode	r/w	1'b0	MJPEG SW mode hblk kick mode 1'b0: SW frame mode 1'b1: SW kick mode
6	reg_sw_kick	w1p	1'b0	MJPEG SW mode hblk kick
5	RSVD			
4:0	reg_sw_frame	r/w	5'd0	MJPEG SW mode frame count

20.4.3 mjpeg_yy_frame_addr

Address：0x30021008

reg_yy_addr_start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_yy_addr_start

Bits	Name	Type	Reset	Description
31:0	reg_yy_addr_start	r/w	32'h80000000	Memory start address of Y frame

20.4.4 mjpeg_uv_frame_addr

Address：0x3002100c

reg_uv_addr_start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_uv_addr_start

Bits	Name	Type	Reset	Description
31:0	reg_uv_addr_start	r/w	32'h80000000	Memory start address of UV frame

20.4.5 mjpeg_yuv_mem

Address：0x30021010

RSVD RSVD RSVD reg_uv_mem_hblk

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD reg_yy_mem_hblk

Bits	Name	Type	Reset	Description
31:29	RSVD			
28:16	reg_uv_mem_hblk	r/w	13'h2	Memory to store block line for UV frame
15:13	RSVD			
12:0	reg_yy_mem_hblk	r/w	13'h2	Memory to store block line for Y frame

20.4.6 jpeg_frame_addr

Address：0x30021014

reg_w_addr_start

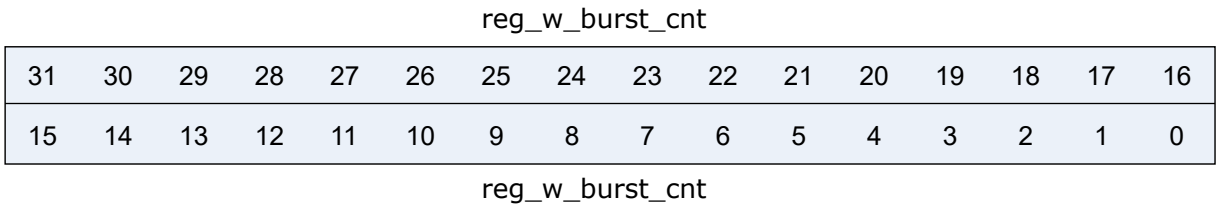
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_w_addr_start

Bits	Name	Type	Reset	Description
31:0	reg_w_addr_start	r/w	32'h80400000	Memory start address of JPEG frame

20.4.7 jpeg_store_memory

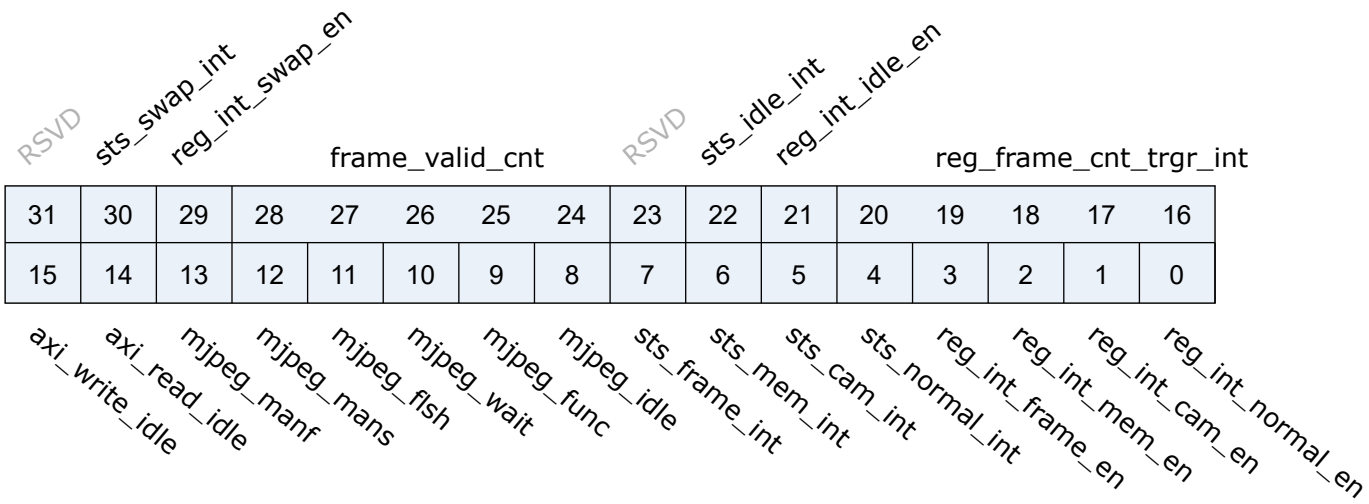
Address：0x30021018



Bits	Name	Type	Reset	Description
31:0	reg_w_burst_cnt	r/w	32'h4000	Memory to store jpeg image burst count

20.4.8 mjpeg_control_3

Address：0x3002101c

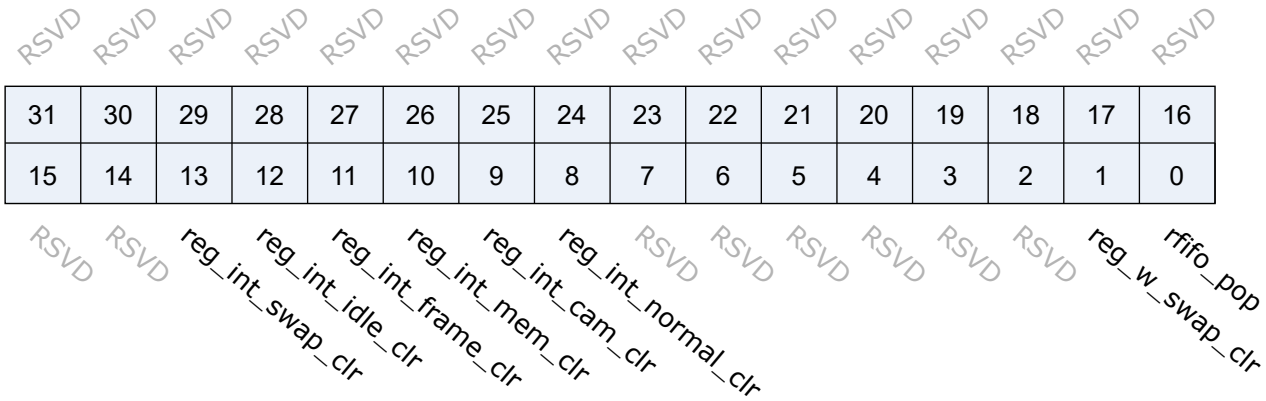


Bits	Name	Type	Reset	Description
31	RSVD			
30	sts_swap_int	r	1'b0	Swap memory block interrupt status
29	reg_int_swap_en	r/w	1'b0	Swap memory block interrupt enable
28:24	frame_valid_cnt	r	5'd0	Frame count valid in status
23	RSVD			
22	sts_idle_int	r	1'b0	Normal Write interrupt status
21	reg_int_idle_en	r/w	1'b0	Back idle interrupt enable
20:16	reg_frame_cnt_trgr_int	r/w	5'd0	Frame threshold to issue interrupt

Bits	Name	Type	Reset	Description
15	axi_write_idle	r	1'b0	AXI write bus idle state
14	axi_read_idle	r	1'b0	AXI read bus idle state
13	mjpeg_manf	r	1'b0	MJPEG in SW run state
12	mjpeg_mans	r	1'b0	MJPEG in SW set state
11	mjpeg_flsh	r	1'b0	MJPEG in flush state
10	mjpeg_wait	r	1'b0	MJPEG in wait state
9	mjpeg_func	r	1'b0	MJPEG in HW function state
8	mjpeg_idle	r	1'b1	MJPEG in idle state
7	sts_frame_int	r	1'b0	Frame OverWrite interrupt status
6	sts_mem_int	r	1'b0	Memory OverWrite interrupt status
5	sts_cam_int	r	1'b0	CAM OverWrite interrupt status (fatal)
4	sts_normal_int	r	1'b0	Normal Write interrupt status
3	reg_int_frame_en	r/w	1'b0	Frame OverWrite interrupt enable
2	reg_int_mem_en	r/w	1'b0	JPEG frame memory OverWrite interrupt enable
1	reg_int_cam_en	r/w	1'b1	YUV frame memory OverWrite interrupt enable (fatal)
0	reg_int_normal_en	r/w	1'b1	Normal Write interrupt enable

20.4.9 mjpeg_frame_fifo_pop

Address：0x30021020



Bits	Name	Type	Reset	Description
31:14	RSVD			
13	reg_int_swap_clr	w1p	1'b0	Wrire this bit with 1'b1 to trigger swap interrupt clear

Bits	Name	Type	Reset	Description
12	reg_int_idle_clr	w1p	1'b0	Wrire this bit with 1'b1 to trigger back idle interrupt clear
11	reg_int_frame_clr	w1p	1'b0	Wrire this bit with 1'b1 to trigger frame overwrite interrupt clear
10	reg_int_mem_clr	w1p	1'b0	Wrire this bit with 1'b1 to trigger memory overwrite interrupt clear
9	reg_int_cam_clr	w1p	1'b0	Wrire this bit with 1'b1 to trigger YUV overwrite interrupt clear
8	reg_int_normal_clr	w1p	1'b0	Wrire this bit with 1'b1 to trigger normal interrupt clear
7:2	RSVD			
1	reg_w_swap_clr	w1p	1'b0	Free current read memory block
0	rfifo_pop	w1p	1'b0	Write this bit with 1'b1 to trigger JPEG frame fifo pop

20.4.10 mjpeg_frame_size

Address：0x30021024

RSVD RSVD RSVD RSVD reg_frame_hblk

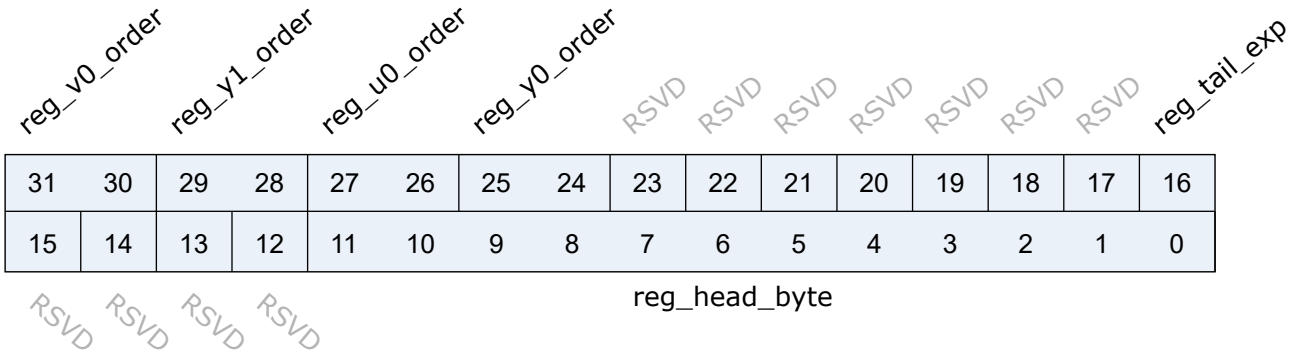
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD reg_frame_wblk

Bits	Name	Type	Reset	Description
31:28	RSVD			
27:16	reg_frame_hblk	r/w	12'd20	Frame total vertical block count
15:12	RSVD			
11:0	reg_frame_wblk	r/w	12'd15	Frame total horizational block count

20.4.11 mjpeg_header_byte

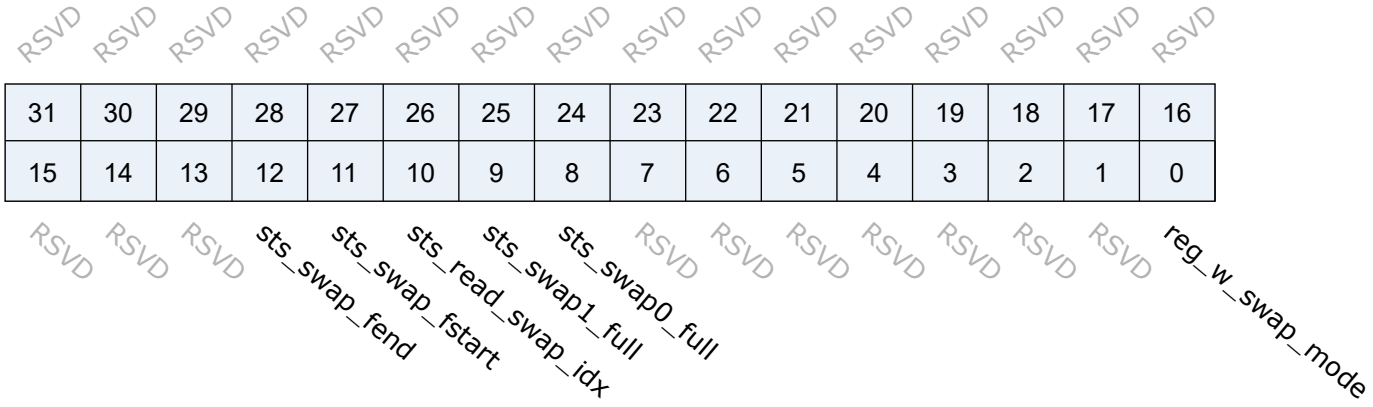
Address：0x30021028



Bits	Name	Type	Reset	Description
31:30	reg_v0_order	r/w	2'd3	V order in Interleave mode
29:28	reg_y1_order	r/w	2'd2	Y1 order in Interleave mode
27:26	reg_u0_order	r/w	2'd1	U order in Interleave mode
25:24	reg_y0_order	r/w	2'd0	Y0 order in Interleave mode
23:17	RSVD			
16	reg_tail_exp	r/w	1'b0	Auto fill tail 0xFF and 0xD9
15:12	RSVD			
11:0	reg_head_byte	r/w	12'h0	Preserve head memory space for each frame

20.4.12 mjpeg_swap_mode

Address：0x30021030



Bits	Name	Type	Reset	Description
31:13	RSVD			
12	sts_swap_fend	r	1'b0	Current read memory block is frame end
11	sts_swap_fstart	r	1'b0	Current read memory block is frame start
10	sts_read_swap_idx	r	1'b0	Current read memory block index 0: memory block0 1: memory block1
9	sts_swap1_full	r	1'b0	Memory block1 is full
8	sts_swap0_full	r	1'b0	Memory block0 is full
7:1	RSVD			
0	reg_w_swap_mode	r/w	1'b0	Enable write swap mode

20.4.13 mjpeg_swap_bit_cnt

Address：0x30021034

frame_swap_end_bit_cnt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_swap_end_bit_cnt

Bits	Name	Type	Reset	Description
31:0	frame_swap_end_bit_cnt	r	32'd0	In swap mode frame remain bit count in last block Only valid when "sts_swap_fend==1"

20.4.14 mjpeg_yuv_mem_sw

Address：0x30021038

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

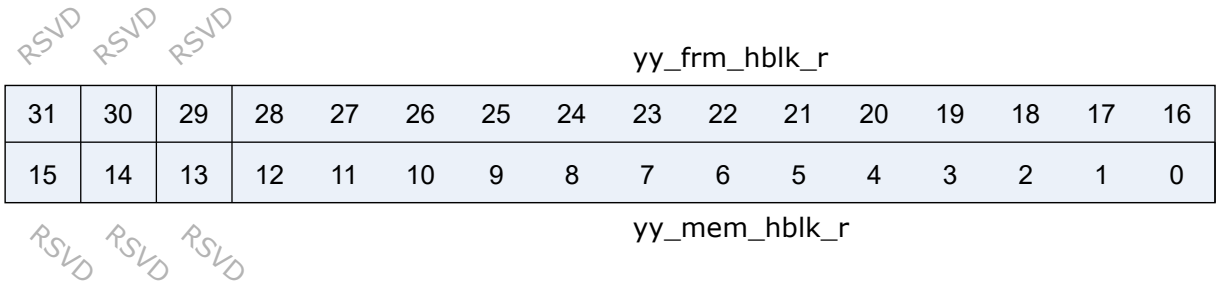
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD reg_sw_kick_hblk

Bits	Name	Type	Reset	Description
31:13	RSVD			
12:0	reg_sw_kick_hblk	r/w	13'h2	Memory to store block line for frame on SW kick

20.4.15 mjpeg_Y_frame_read_status_1

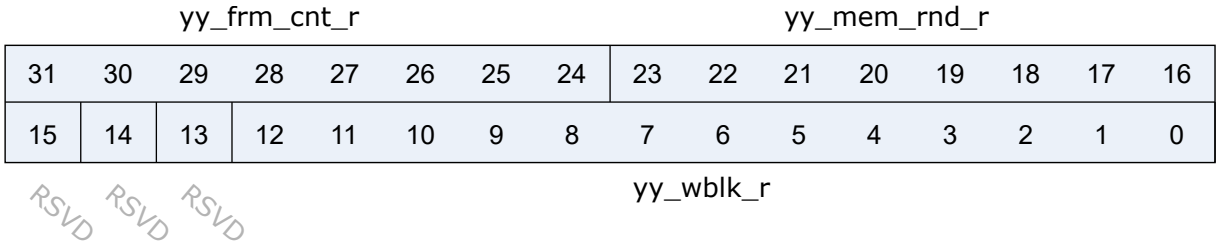
Address：0x30021040



Bits	Name	Type	Reset	Description
31:29	RSVD			
28:16	yy_frm_hblk_r	r	13'd0	Y frame veritical block read status
15:13	RSVD			
12:0	yy_mem_hblk_r	r	13'd0	Y memory veritical block read status

20.4.16 mjpeg_Y_frame_read_status_2

Address：0x30021044



Bits	Name	Type	Reset	Description
31:24	yy_frm_cnt_r	r	8'd0	Y frame read count
23:16	yy_mem_rnd_r	r	8'd0	Y memory read round
15:13	RSVD			

Bits	Name	Type	Reset	Description
12:0	yy_wblk_r	r	13'd0	Y frame horizational block read status

20.4.17 mjpeg_Y_frame_write_status

Address：0x30021048

yy_frm_cnt_w								yy_mem_rnd_w							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	yy_mem_hblk_w												

Bits	Name	Type	Reset	Description
31:24	yy_frm_cnt_w	r	8'd0	Y frame write count
23:16	yy_mem_rnd_w	r	8'd0	Y memory write round
15:13	RSVD			
12:0	yy_mem_hblk_w	r	13'd0	Y memory veritical block write status

20.4.18 mjpeg_UV_frame_read_status_1

Address：0x3002104c

RSVD	RSVD	RSVD	uv_frm_hblk_r												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	uv_mem_hblk_r												

Bits	Name	Type	Reset	Description
31:29	RSVD			
28:16	uv_frm_hblk_r	r	13'd0	UV frame veritical block read status
15:13	RSVD			
12:0	uv_mem_hblk_r	r	13'd0	UV memory veritical block read status

20.4.19 mjpeg_UV_frame_read_status_2

Address：0x30021050

uv_frm_cnt_r | uv_mem_rnd_r

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	uv_wblk_r												

Bits	Name	Type	Reset	Description
31:24	uv_frm_cnt_r	r	8'd0	UV frame read count
23:16	uv_mem_rnd_r	r	8'd0	UV memory read round
15:13	RSVD			
12:0	uv_wblk_r	r	13'd0	UV frame horizational block read status

20.4.20 mjpeg_UV_frame_write_status

Address：0x30021054

uv_frm_cnt_w | uv_mem_rnd_w

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	uv_mem_hblk_w												

Bits	Name	Type	Reset	Description
31:24	uv_frm_cnt_w	r	8'd0	UV frame write count
23:16	uv_mem_rnd_w	r	8'd0	UV memory write round
15:13	RSVD			
12:0	uv_mem_hblk_w	r	13'd0	UV memory veritical block write status

20.4.21 mjpeg_frame_w_hblk_status

Address：0x30021058

RSVD RSVD RSVD uv_frm_hblk_w

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD yy_frm_hblk_w

Bits	Name	Type	Reset	Description
31:29	RSVD			
28:16	uv_frm_hblk_w	r	13'd0	UV frame vertical block write status
15:13	RSVD			
12:0	yy_frm_hblk_w	r	13'd0	YY frame vertical block write status

20.4.22 mjpeg_start_addr0

Address：0x30021080

frame_start_addr_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_0

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_0	r	32'd0	MJPEG FRAME0 Start address

20.4.23 mjpeg_bit_cnt0

Address：0x30021084

frame_bit_cnt_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_bit_cnt_0

Bits	Name	Type	Reset	Description
31:0	frame_bit_cnt_0	r	32'd0	MJPEG FRAME0 total bit count

20.4.24 mjpeg_start_addr1

Address：0x30021088

frame_start_addr_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_1

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_1	r	32'd0	MJPEG FRAME1 Start address

20.4.25 mjpeg_bit_cnt1

Address：0x3002108c

frame_bit_cnt_1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_bit_cnt_1

Bits	Name	Type	Reset	Description
31:0	frame_bit_cnt_1	r	32'd0	MJPEG FRAME1 total bit count

20.4.26 mjpeg_start_addr2

Address：0x30021090

frame_start_addr_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_2

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_2	r	32'd0	MJPEG FRAME2 Start address

20.4.27 mjpeg_bit_cnt2

Address：0x30021094

frame_bit_cnt_2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_bit_cnt_2

Bits	Name	Type	Reset	Description
31:0	frame_bit_cnt_2	r	32'd0	MJPEG FRAME2 total bit count

20.4.28 mjpeg_start_addr3

Address：0x30021098

frame_start_addr_3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_start_addr_3

Bits	Name	Type	Reset	Description
31:0	frame_start_addr_3	r	32'd0	MJPEG FRAME3 Start address

20.4.29 mjpeg_bit_cnt3

Address：0x3002109c

frame_bit_cnt_3

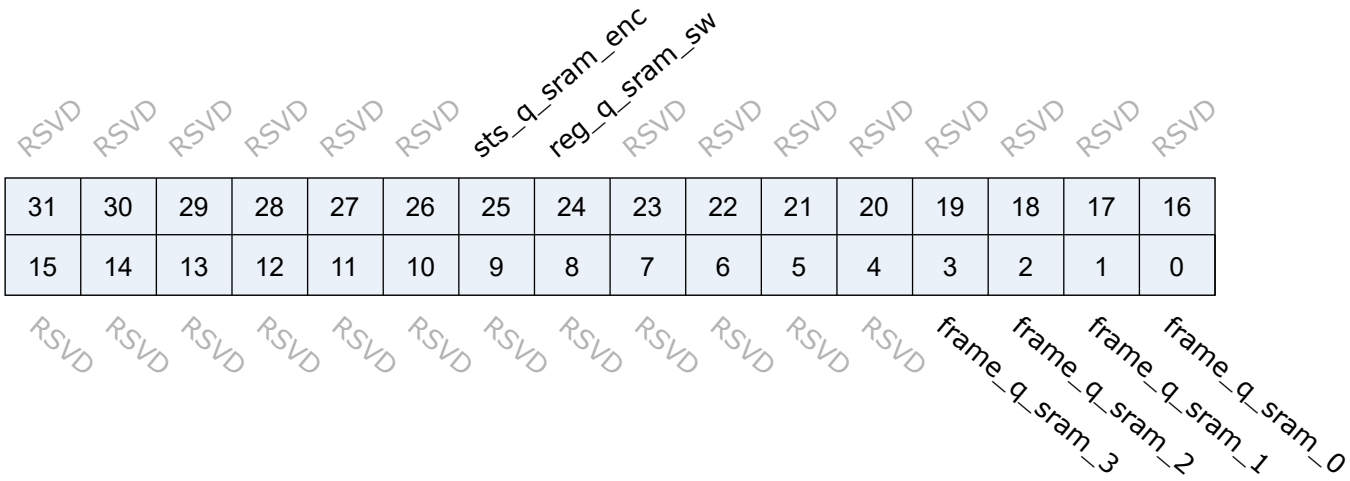
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_bit_cnt_3

Bits	Name	Type	Reset	Description
31:0	frame_bit_cnt_3	r	32'd0	MJPEG FRAME3 total bit count

20.4.30 mjpeg_q_enc

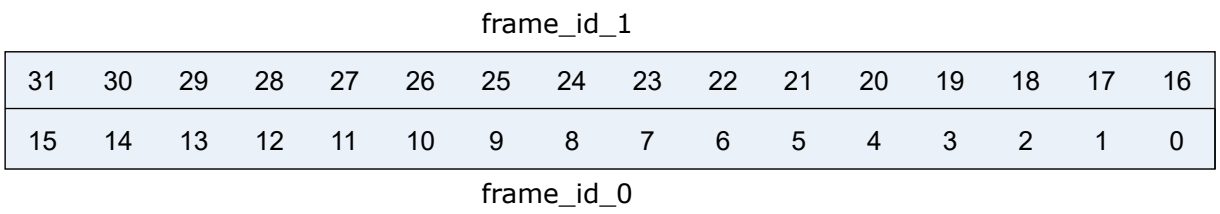
Address：0x30021100



Bits	Name	Type	Reset	Description
31:26	RSVD			
25	sts_q_sram_enc	r	1'd0	Current Quantize SRAM selection for Encode
24	reg_q_sram_sw	w1p	1'd0	Quantize SRAM Switch
23:4	RSVD			
3	frame_q_sram_3	r	1'd0	MJPEG FRAME3 Quantize SRAM selection
2	frame_q_sram_2	r	1'd0	MJPEG FRAME2 Quantize SRAM selection
1	frame_q_sram_1	r	1'd0	MJPEG FRAME1 Quantize SRAM selection
0	frame_q_sram_0	r	1'd0	MJPEG FRAME0 Quantize SRAM selection

20.4.31 mjpeg_frame_id_10

Address：0x30021110



Bits	Name	Type	Reset	Description
31:16	frame_id_1	r	16'd0	JPEG PIC 1 ID
15:0	frame_id_0	r	16'd0	JPEG PIC 0 ID

20.4.32 mjpeg_frame_id_32

Address：0x30021114

frame_id_3

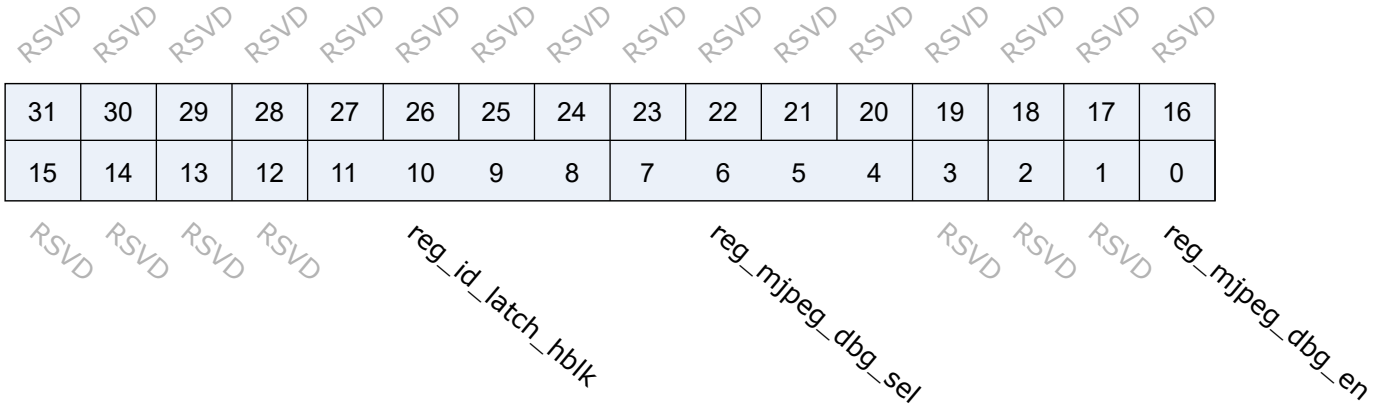
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

frame_id_2

Bits	Name	Type	Reset	Description
31:16	frame_id_3	r	16'd0	JPEG PIC 3 ID
15:0	frame_id_2	r	16'd0	JPEG PIC 2 ID

20.4.33 mjpeg_debug

Address：0x300211f0



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	reg_id_latch_hblk	r/w	4'd1	ID latch timing (hblk count)
7:4	reg_mjpeg_dbg_sel	r/w	4'd0	MJPEG debgu flag selection
3:1	RSVD			

Bits	Name	Type	Reset	Description
0	reg_mjpeg_dbg_en	r/w	1'b0	MJPEG debgu flag enable

21 DBI

21.1 Overview

The module also integrates the Quad Serial Peripheral Interface (QSPI) display interface. QSPI with four data lines is an extension of the SPI interface and it greatly improves the transmission efficiency.

21.2 Features

- Additional support for QSPI mode, with the following features: * A single transmission includes a one-byte command phase, an adjustable address phase of one to three bytes, and an optional data phase. * Command, address and data reading and writing all support 1-wire /4-wire mode, which can be combined as needed.
- Except for QSPI mode, a single transmission supports optional command and data phases, and the QSPI interface has another address phase.
- The data phase can be set to normal mode in bytes and pixel mode in pixels: * The data phase (normal mode) is used to transmit configuration data. it can write up to 256 bytes and read up to 8 bytes at a time. * The data phase (pixel mode) is used to transmit pixel data, and the data size is in pixels. It can write 2 to the power of 24 pixels at a time and is unreadable.
- The input pixel formats support RGB and YUV444, in which RGB supports eight memory arrangements:
- YUV444 supports six memory arrangements: * YUVN8888 * VUYN8888 * NYUV8888 * NVUY8888 * YUV888 * VUY888
- Hardware transcoding from YUV444 to RGB565/666/888
- The CS signal pull-up release condition can be configured

21.3 Function description

The basic block diagram of the DBI module is shown in the figure.

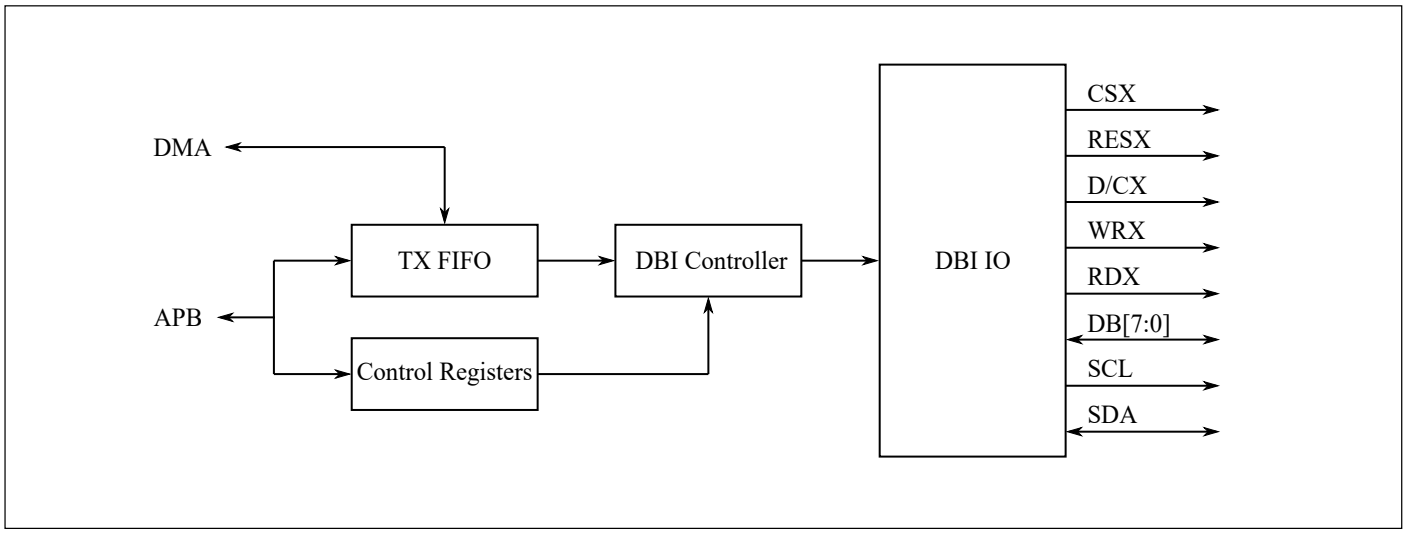


Fig. 21.1: DBI basic block diagram

21.3.1 DBI Type B

21.3.1.1 Write timing

The DBI Type B mode has an 8-bit parallel data line. The sequence of writing data from the MCU to the display module is shown in the following figure:

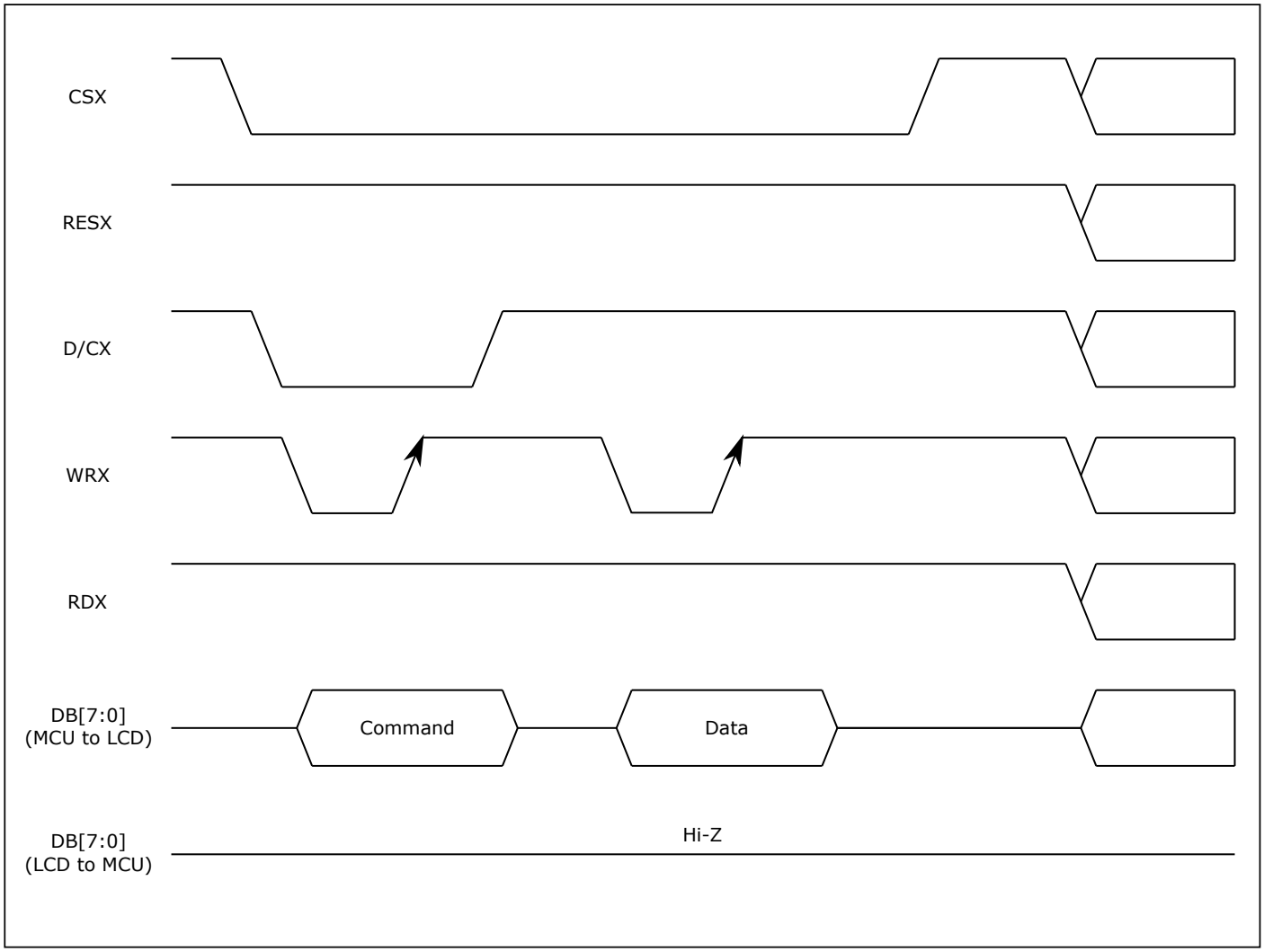


Fig. 21.2: Write timing

- CSX: Chip select signal. When the signal is low, the display module is selected, and when it is high, the display module will ignore all other interface signals
- RESX: External reset signal. When the signal is low, the display module is reset
- D/CX: Data/Command selection signal. When the signal is 0, the DB[7:0] bit is the command; when the signal is 1, the DB[7:0] bit is the RAM data or command parameter
- WRX: Parallel data write strobe signal. This signal is driven high to low during the write cycle and then pulled back high. The display module reads the information transmitted by the MCU at the rising edge of this signal. The signal is an asynchronous signal that can be terminated when not in use
- RDX: Parallel data read strobe signal. This signal is driven high to low and then pulled back high during the read cycle. The MCU reads the information transmitted by the display module at the rising edge of this signal. The signal is an asynchronous signal that can be terminated when not in use
- DB[7:0]: 8-bit data signal. Used to transfer commands, command parameters, or data

21.3.1.2 Read timing

The sequence of MCU reading data from the display module is shown in the following figure:

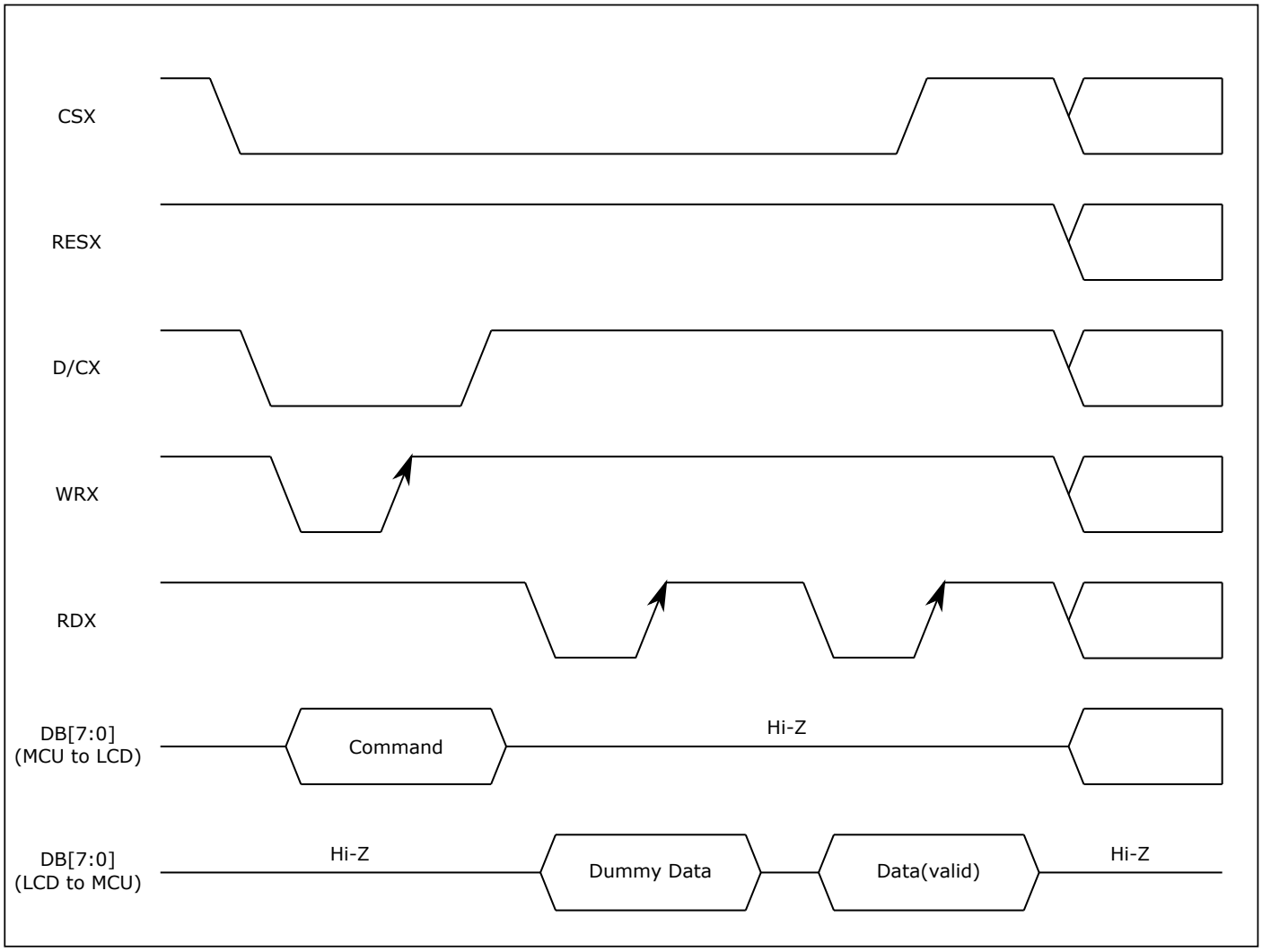


Fig. 21.3: Read timing

21.3.1.3 Output RGB565

The data output in RGB565 format is shown in the following figure:

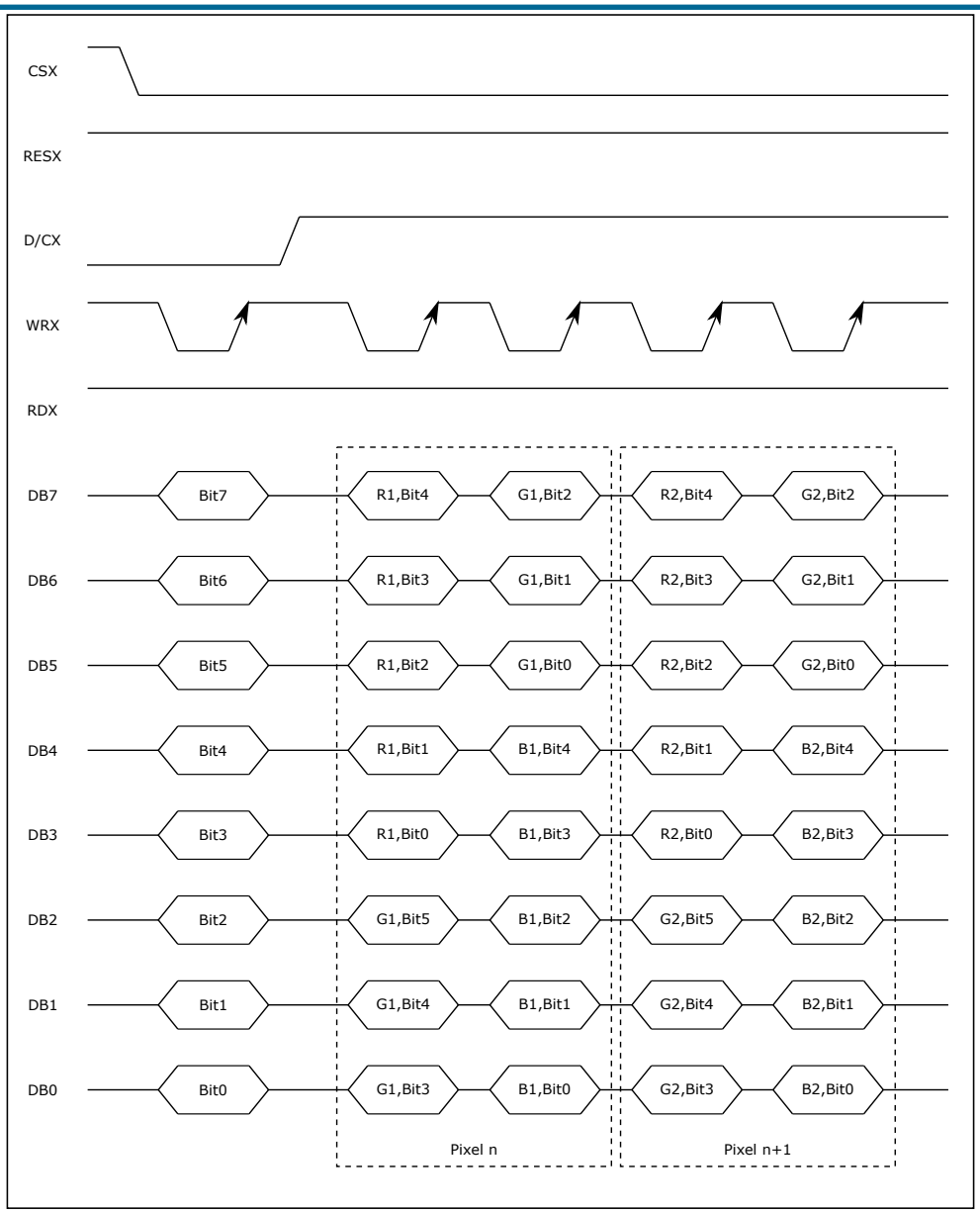


Fig. 21.4: RGB565 output

21.3.1.4 Output RGB666

The data output in RGB666 format is shown in the following figure:

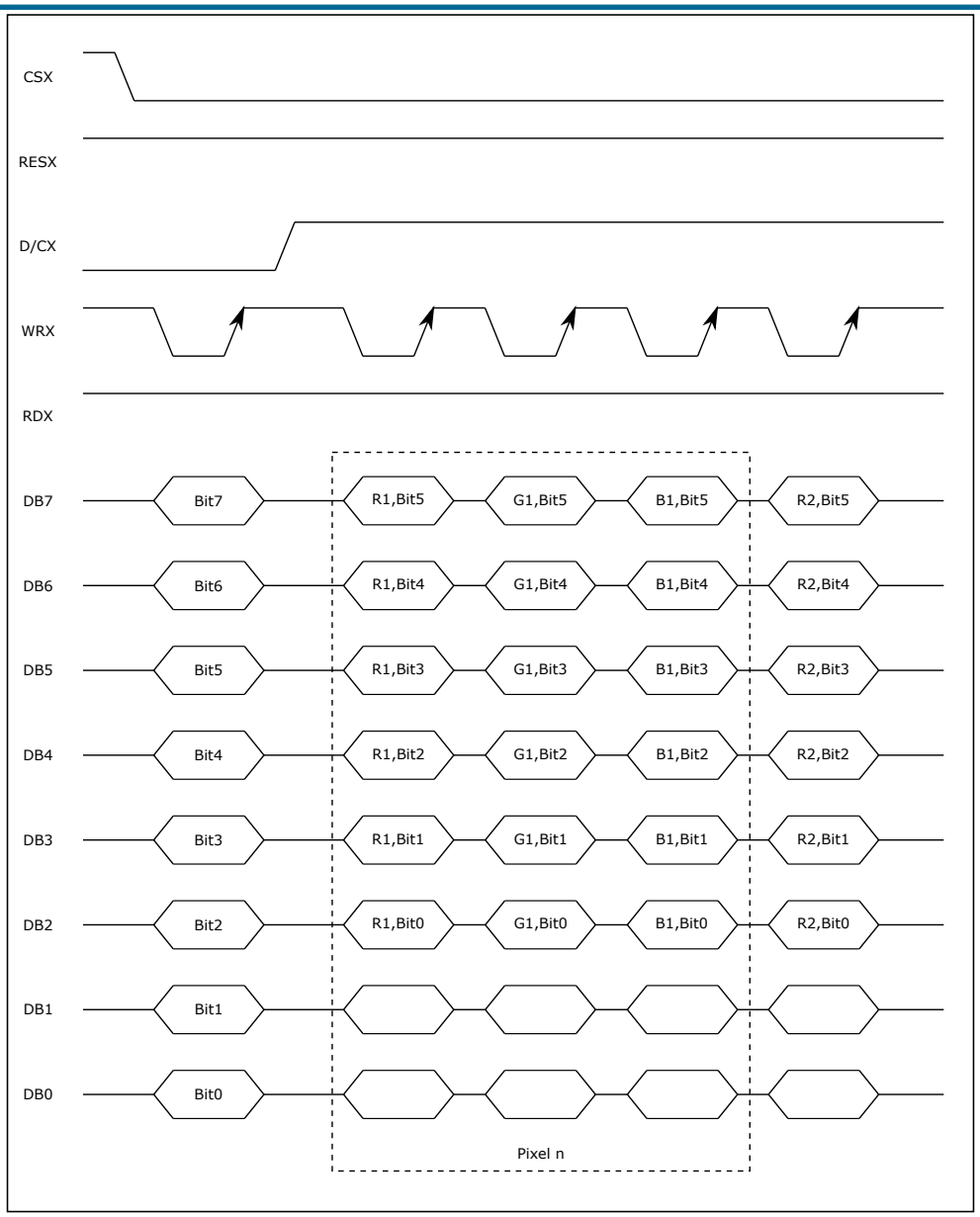


Fig. 21.5: RGB666 output

21.3.1.5 Output RGB888

The data output in RGB888 format is shown in the following figure:

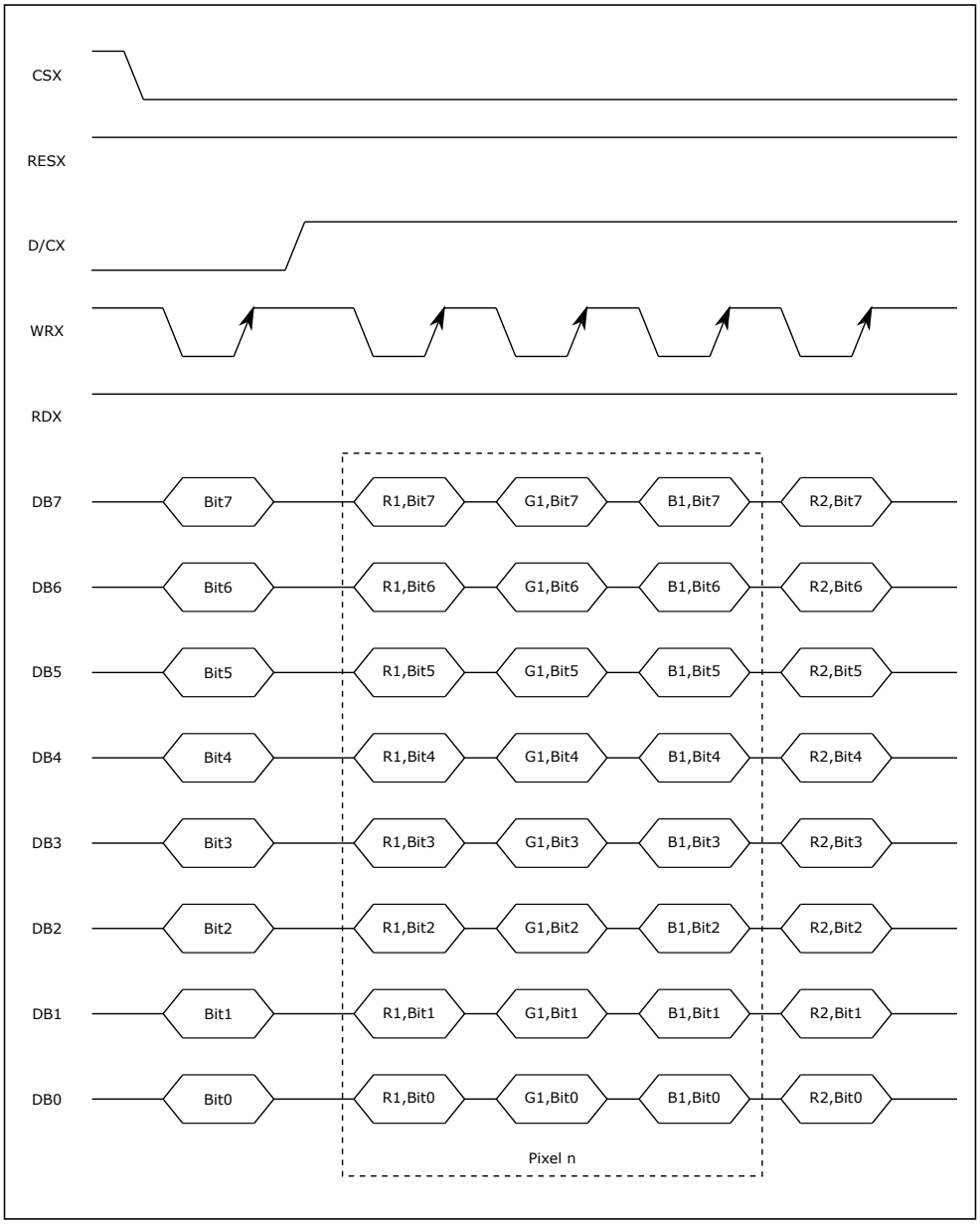


Fig. 21.6: RGB888 output

21.3.2 DBI Type C 3-Line

21.3.2.1 Write timing

The DBI Type C mode is a 3-wire 9-bit serial interface. The sequence of writing data from the MCU to the display module is shown in the following figure:

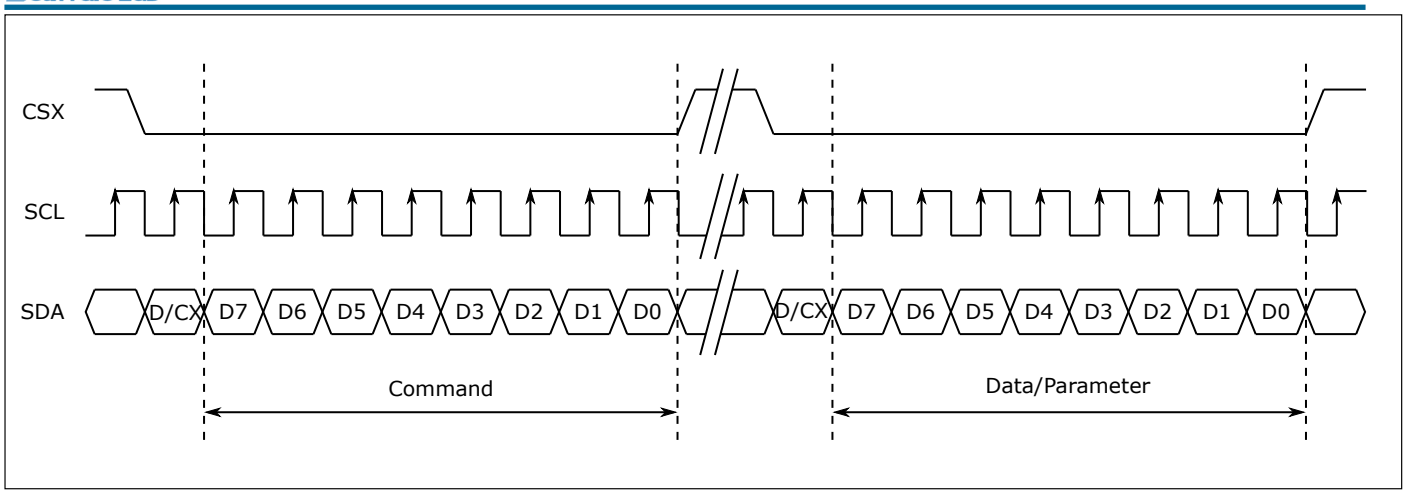


Fig. 21.7: Write timing

- CSX: Chip select signal. When the signal is low, the display module is selected, and when it is high, the display module will ignore all other interface signals;
- SCL: Serial clock input signal. Used to provide a clock signal for data transmission.
- SDA: Serial data input/output signal. In a write operation, the serial data packet contains a D/CX (Data/Command) select bit and a transfer byte. If the D/CX bit is low, the transmitted byte is a command; if the D/CX bit is high, the transmitted byte is display data or command parameters.

21.3.2.2 Read timing

The sequence of MCU reading data from the display module is shown in the following figure:

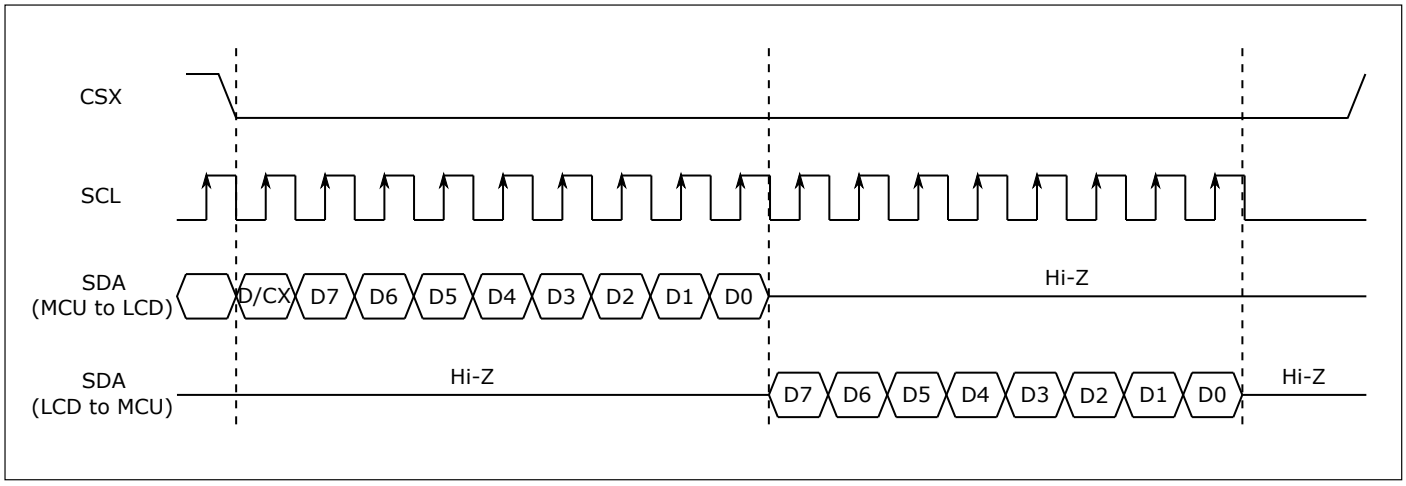


Fig. 21.8: Read timing

21.3.2.3 Output RGB565

The data output in RGB565 format is shown in the following figure:

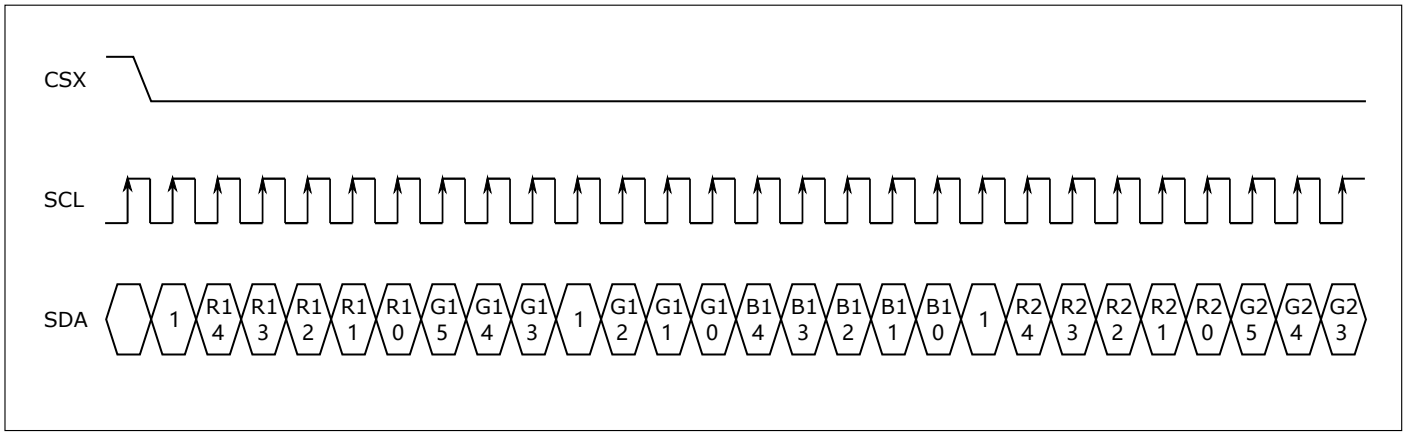


Fig. 21.9: RGB565 output

21.3.2.4 Output RGB666

The data output in RGB666 format is shown in the following figure:

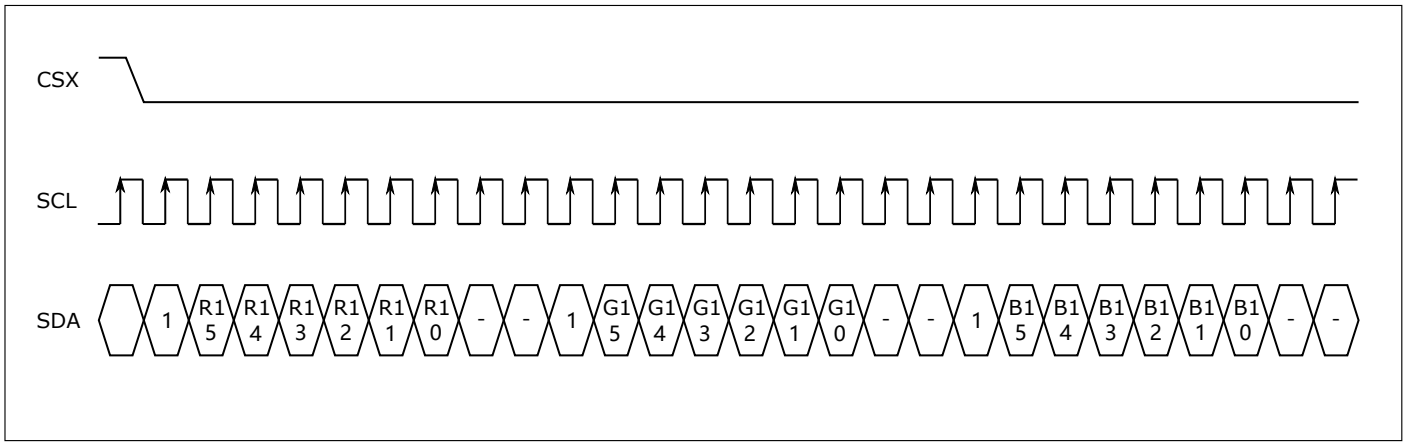


Fig. 21.10: RGB666 output

21.3.2.5 Output RGB888

The data output in RGB888 format is shown in the following figure:

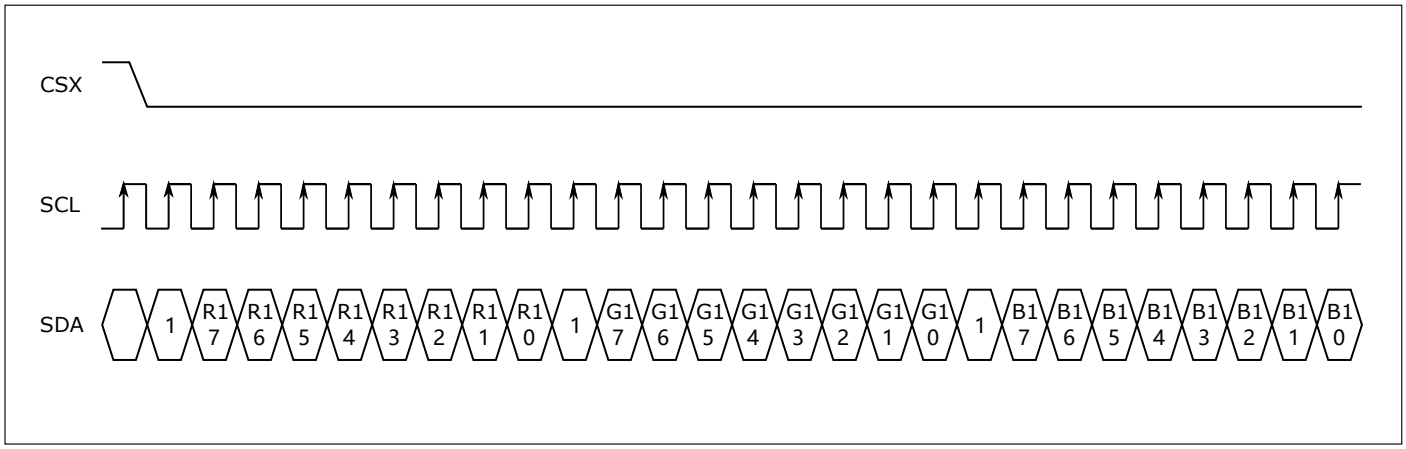


Fig. 21.11: RGB888 output

21.3.3 DBI Type C 4-Line

21.3.3.1 Write timing

The DBI Type C mode is a 4-wire 8-bit serial interface. The sequence of writing data from the MCU to the display module is shown in the following figure:

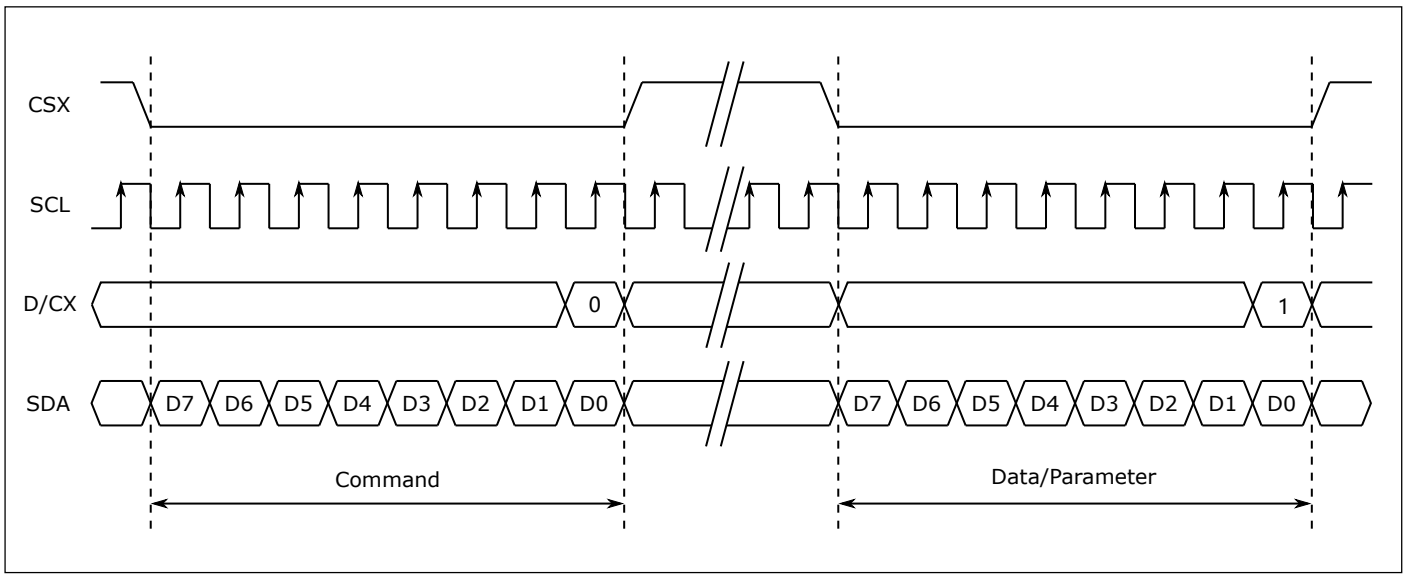


Fig. 21.12: Write timing

- CSX: Chip select signal. When the signal is low, the display module is selected, and when it is high, the display module will ignore all other interface signals

- D/CX: Data/Command selection signal. If the signal is low, the information transmitted by SDA is a command; if the signal is high, the information transmitted by SDA is display data or command parameters
- SCL: Serial clock input signal. Used to provide a clock signal for data transmission
- SDA: Serial data input/output signal. Used to transfer commands, command parameters, or data

21.3.3.2 Read timing

The sequence of MCU reading data from the display module is shown in the following figure:

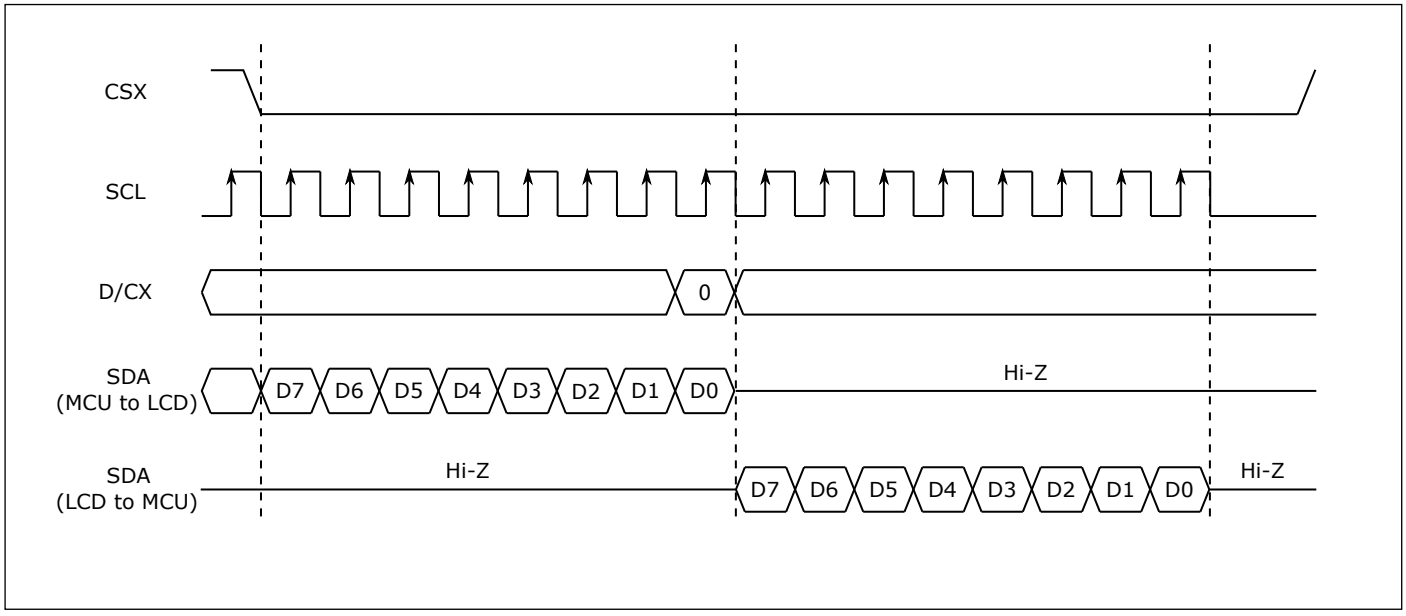


Fig. 21.13: Read timing

21.3.3.3 Output RGB565

RGB565 格式数据输出如下图所示：

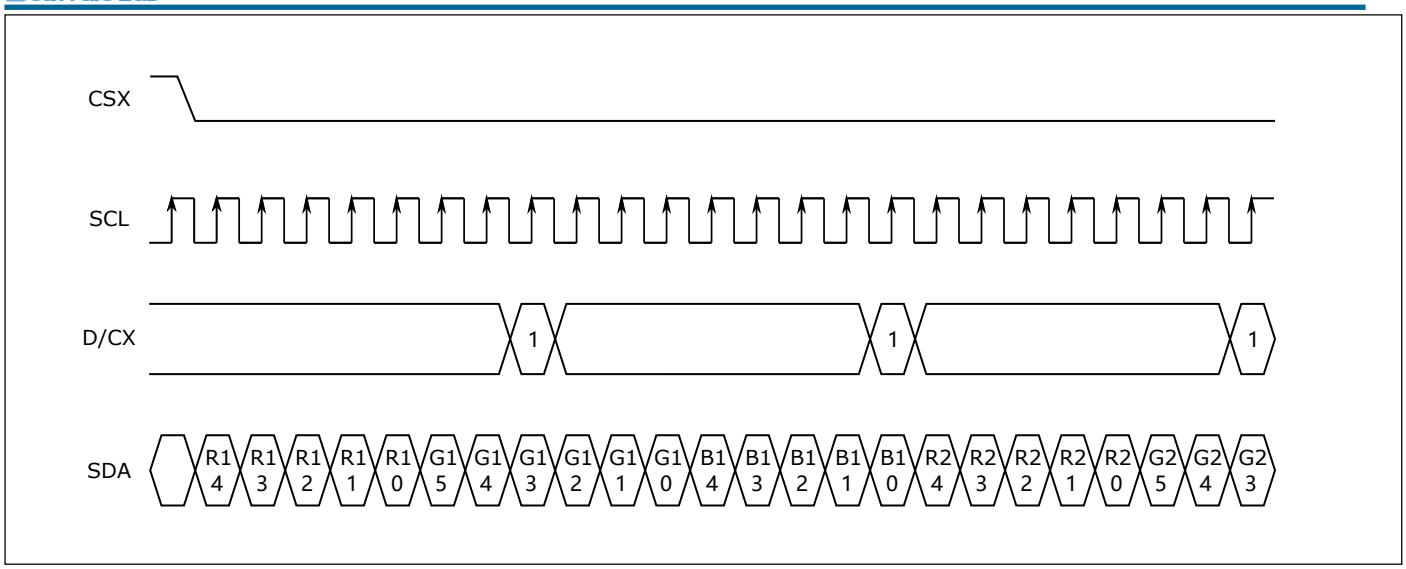


Fig. 21.14: RGB565 output

21.3.3.4 Output RGB666

The data output in RGB666 format is shown in the following figure:

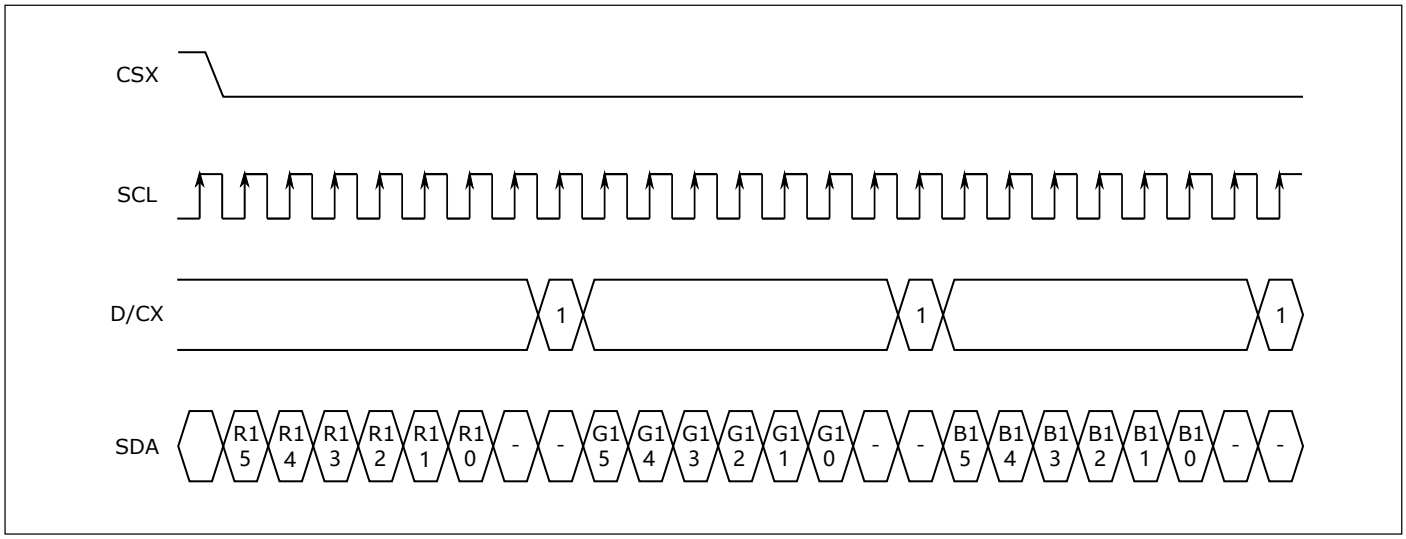


Fig. 21.15: RGB666 output

21.3.3.5 Output RGB888

The data output in RGB888 format is shown in the following figure:

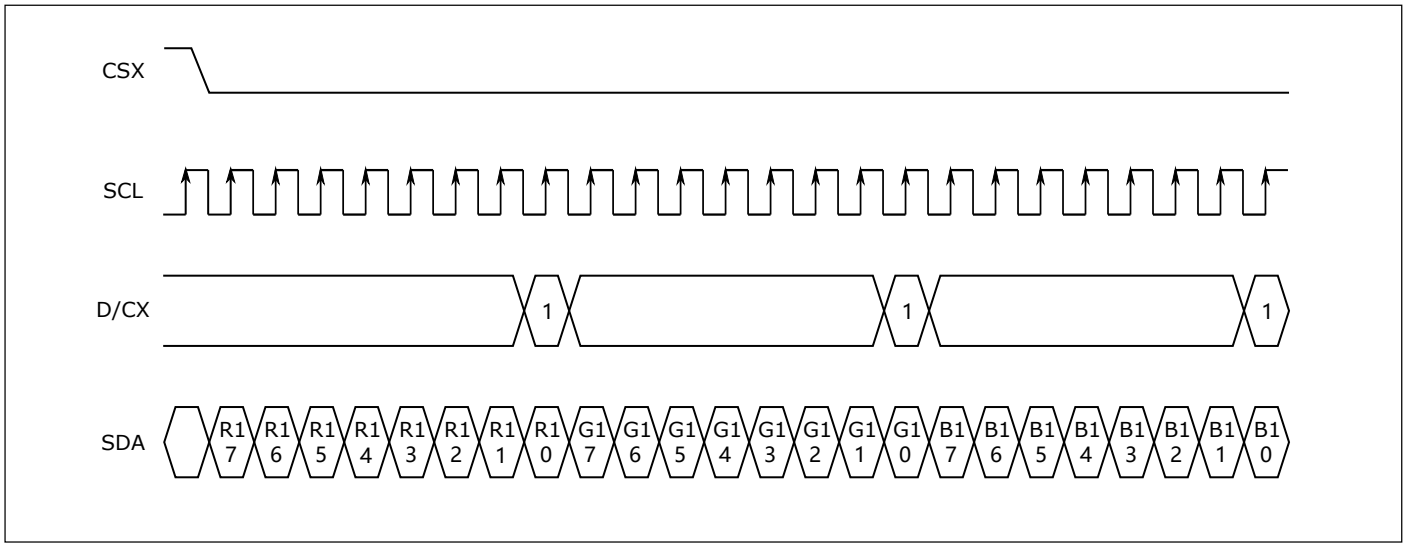


Fig. 21.16: RGB888 output

21.3.4 QSPI

21.3.4.1 Write timing

Taking 1-wire command, 1-wire 3-byte address, and 1-wire data pattern as examples, the timing of MCU writing data to the Display Module is shown as follows:

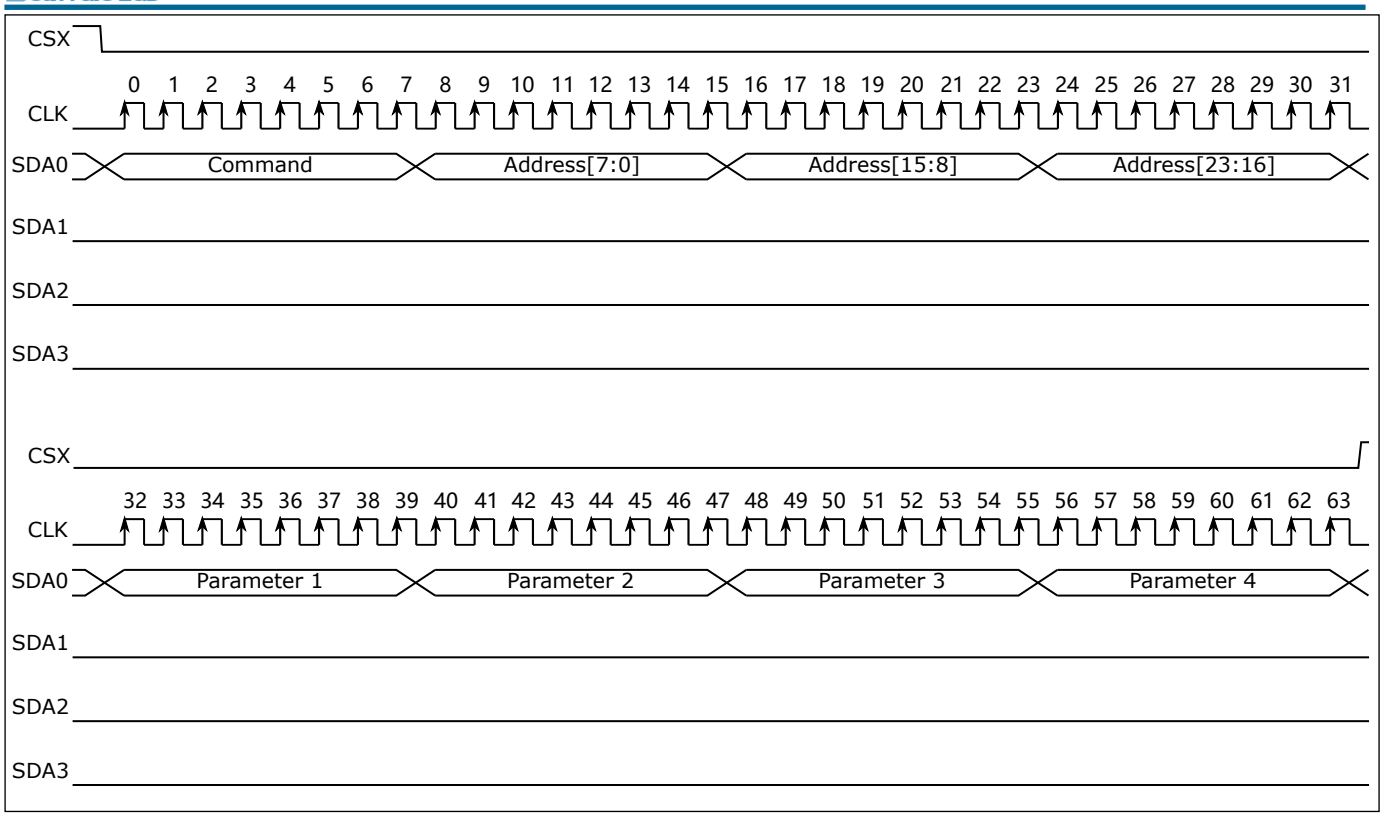


Fig. 21.17: Write timing

- CSX: chip select signal. When the signal is Low, the display module is selected; when it is High, the display module ignores all other interface signals.
- CLK: clock input signal, which provides a clock signal for data transfer.
- SDA0: The data line 0 is used to transmit commands, addresses or data.
- SDA1: The data line 1 is used to transmit commands, addresses or data.
- SDA2: The data line 2 is used to transmit commands, addresses or data.
- SDA3: The data line 3 is used to transmit commands, addresses or data.

21.3.4.2 Read timing

Taking 1-wire command, 1-wire 3-byte address, and 1-wire data pattern as examples, the timing of MCU reading data from the Display Module is shown as follows:

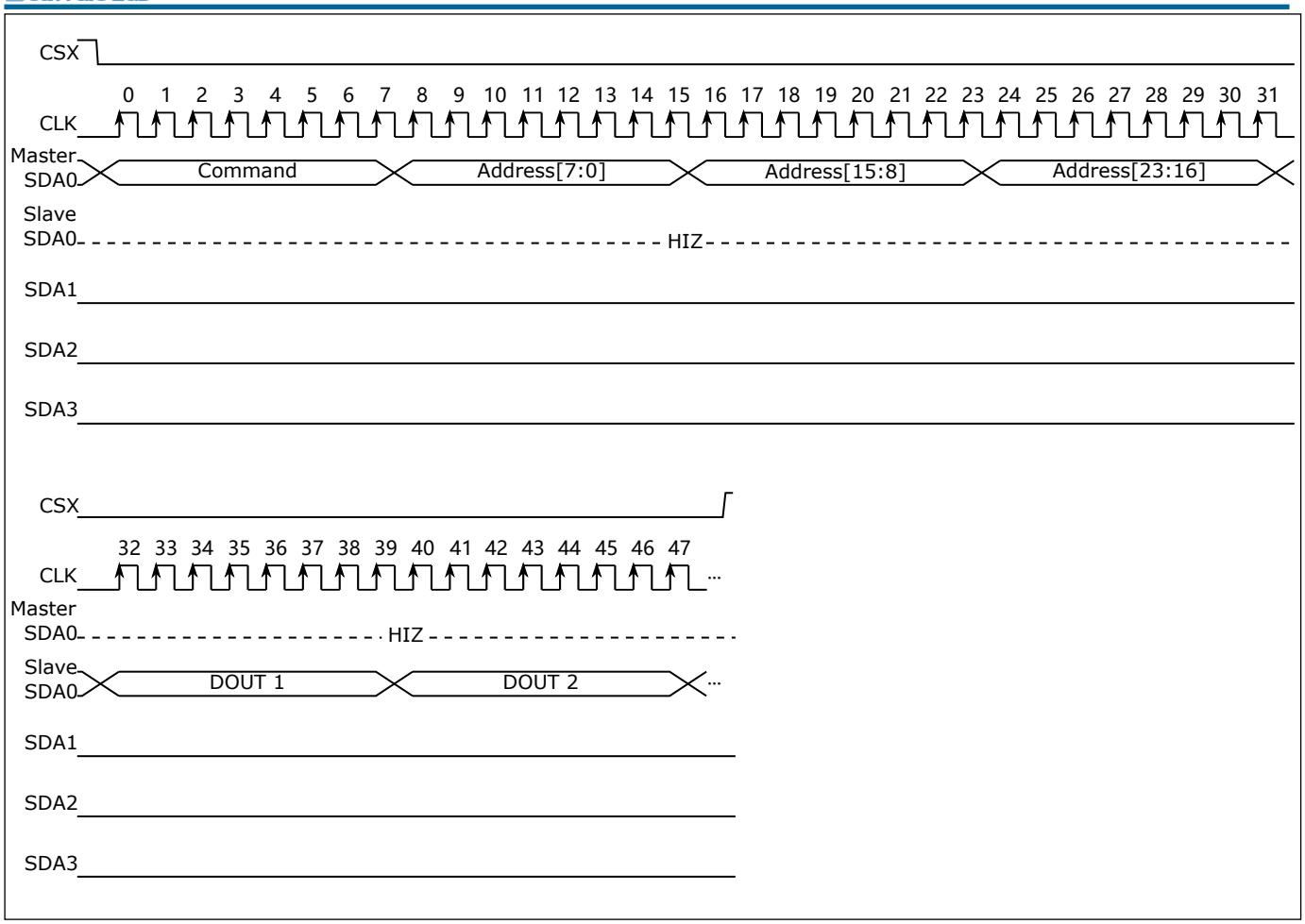


Fig. 21.18: Read timing

21.3.4.3 1-Wire Command, 1-Wire Address and 1-Wire Data Pattern

Taking the command 0x02 and 3-byte address 0x002C00/0x003C00 as example, the output of RGB565 data under 1-wire command, 1-wire address and 1-wire data pattern is shown as follows:

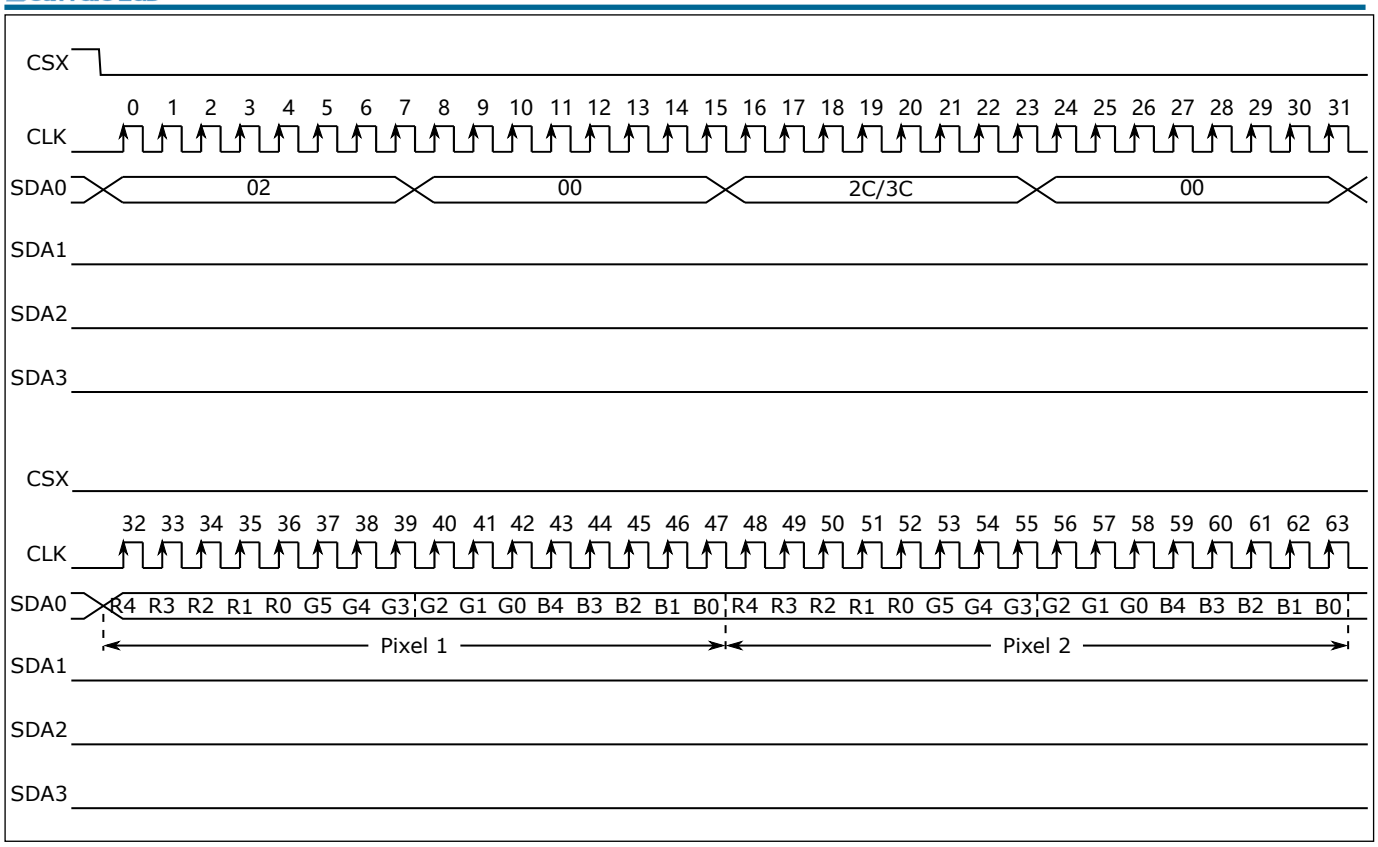


Fig. 21.19: RGB565 output

The data output in RGB666 format is shown as follows:

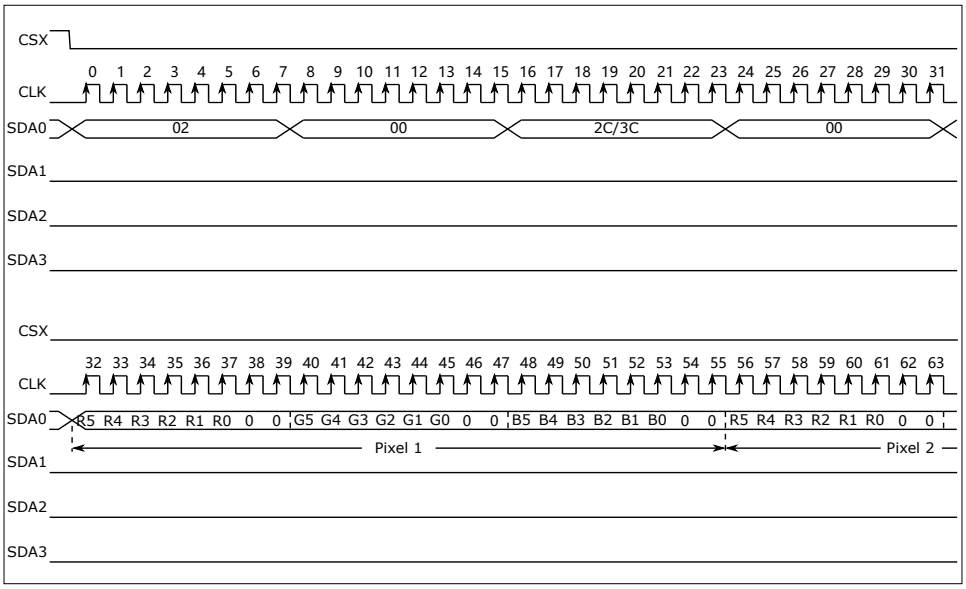


Fig. 21.20: RGB666 output

The data output in RGB888 format is shown as follows:

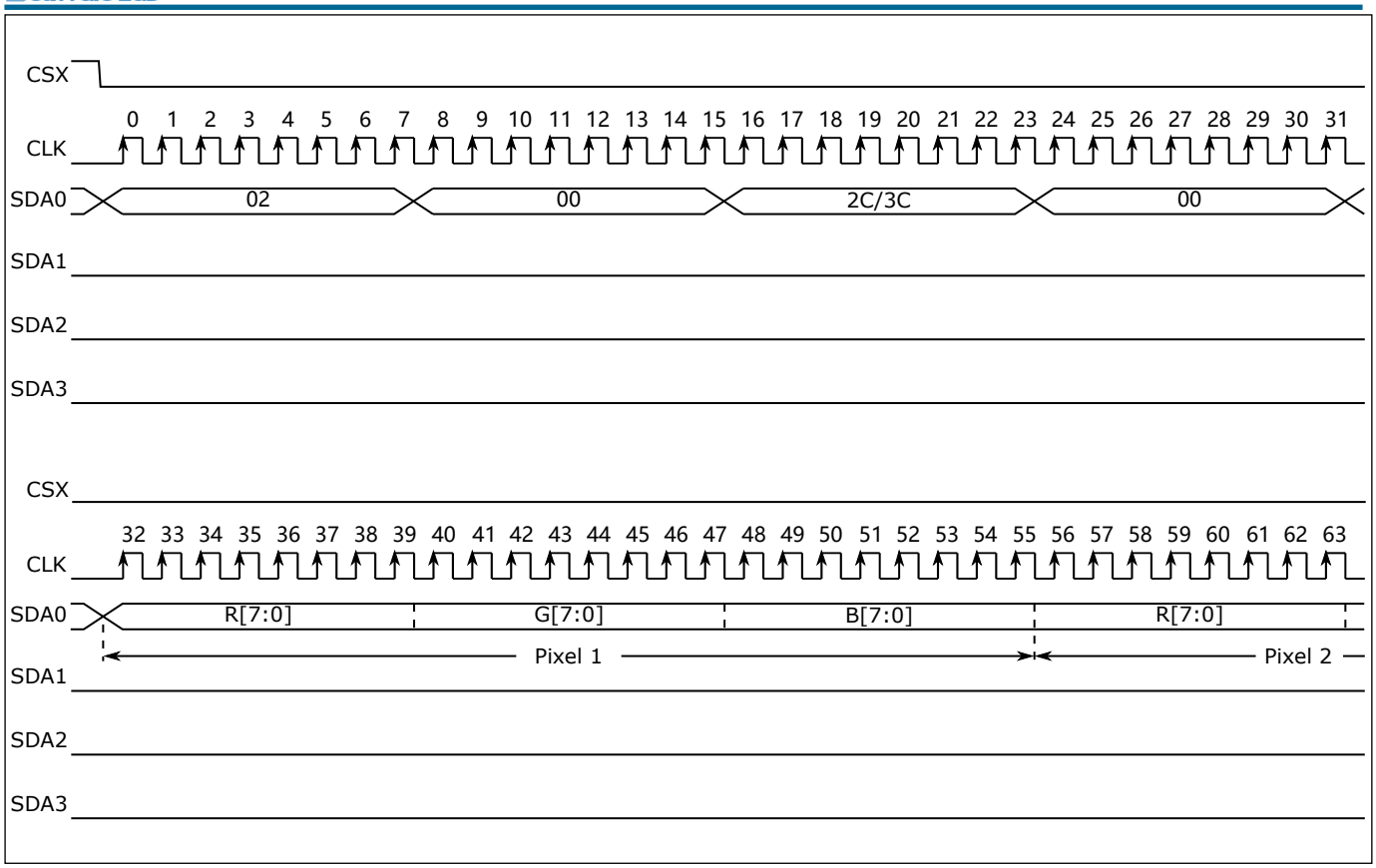


Fig. 21.21: RGB888 output

21.3.4.4 1-Wire Command, 1-Wire Address and 4-Wire Data Pattern

Taking the command 0x32 and 3-byte address 0x002C00/0x003C00 as example, the output of RGB565 data under 1-wire command, 1-wire address and 4-wire data pattern is shown as follows:

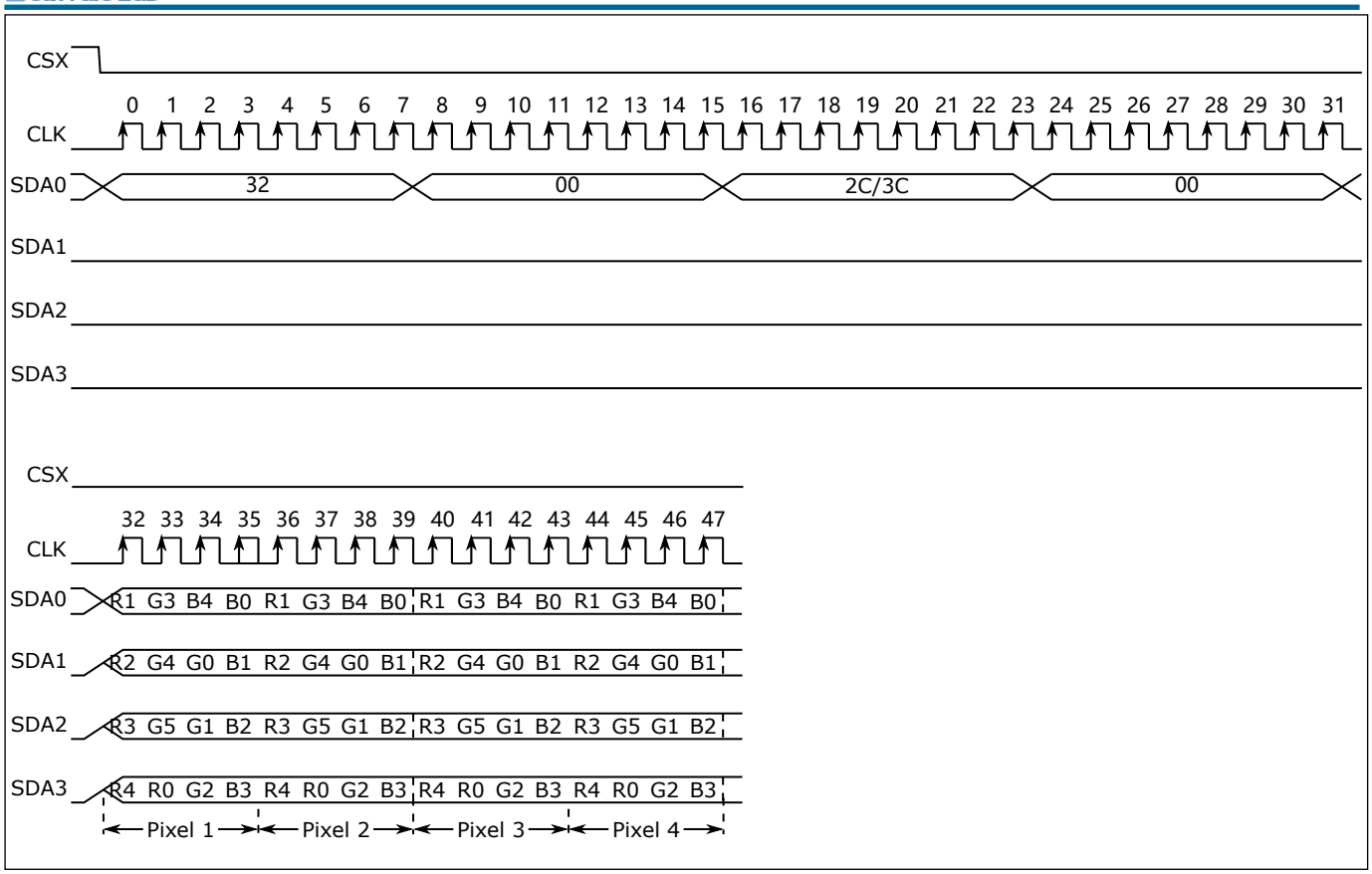


Fig. 21.22: RGB565 output

The data output in RGB666 format is shown as follows:

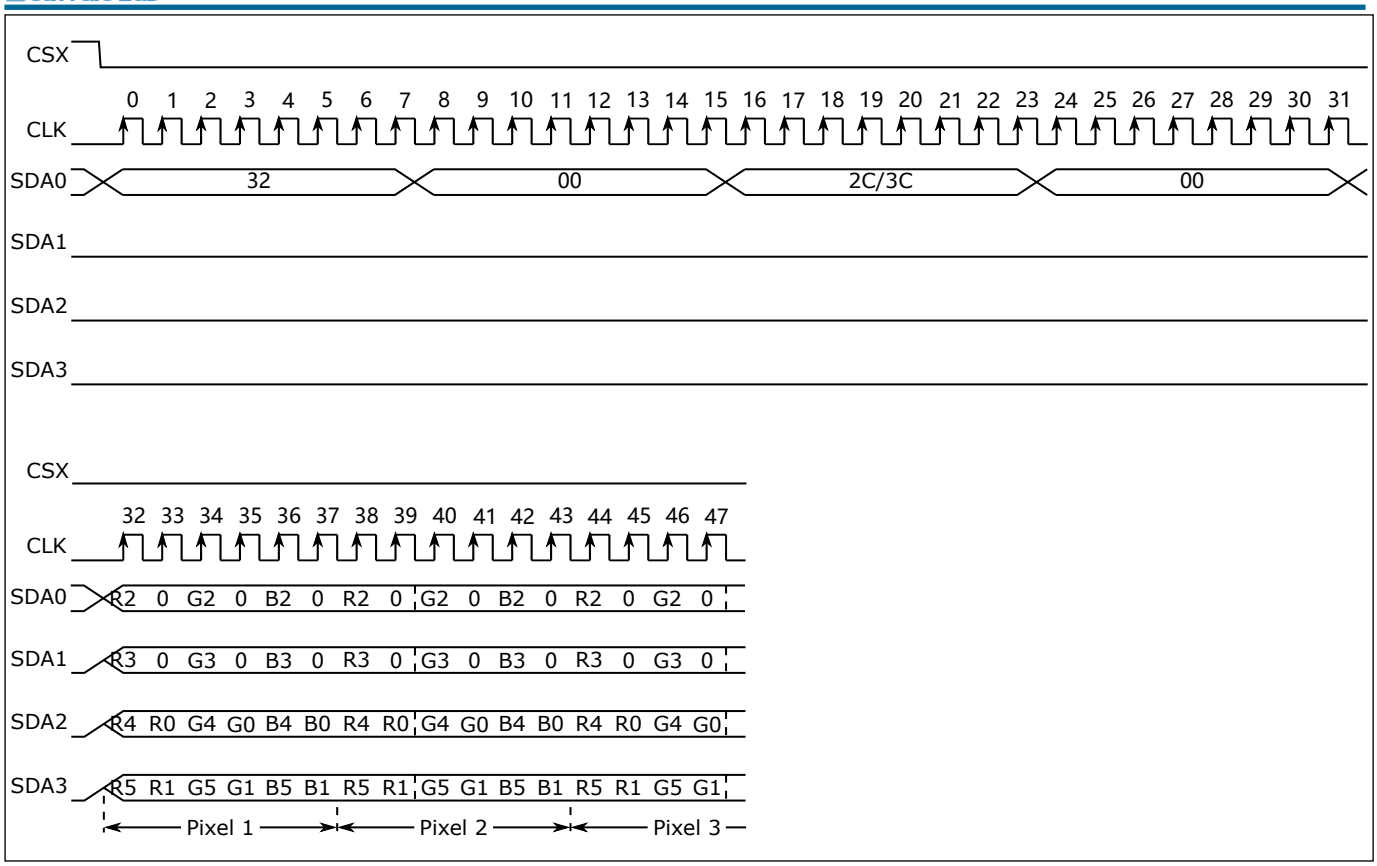


Fig. 21.23: RGB666 output

The data output in RGB888 format is shown as follows:

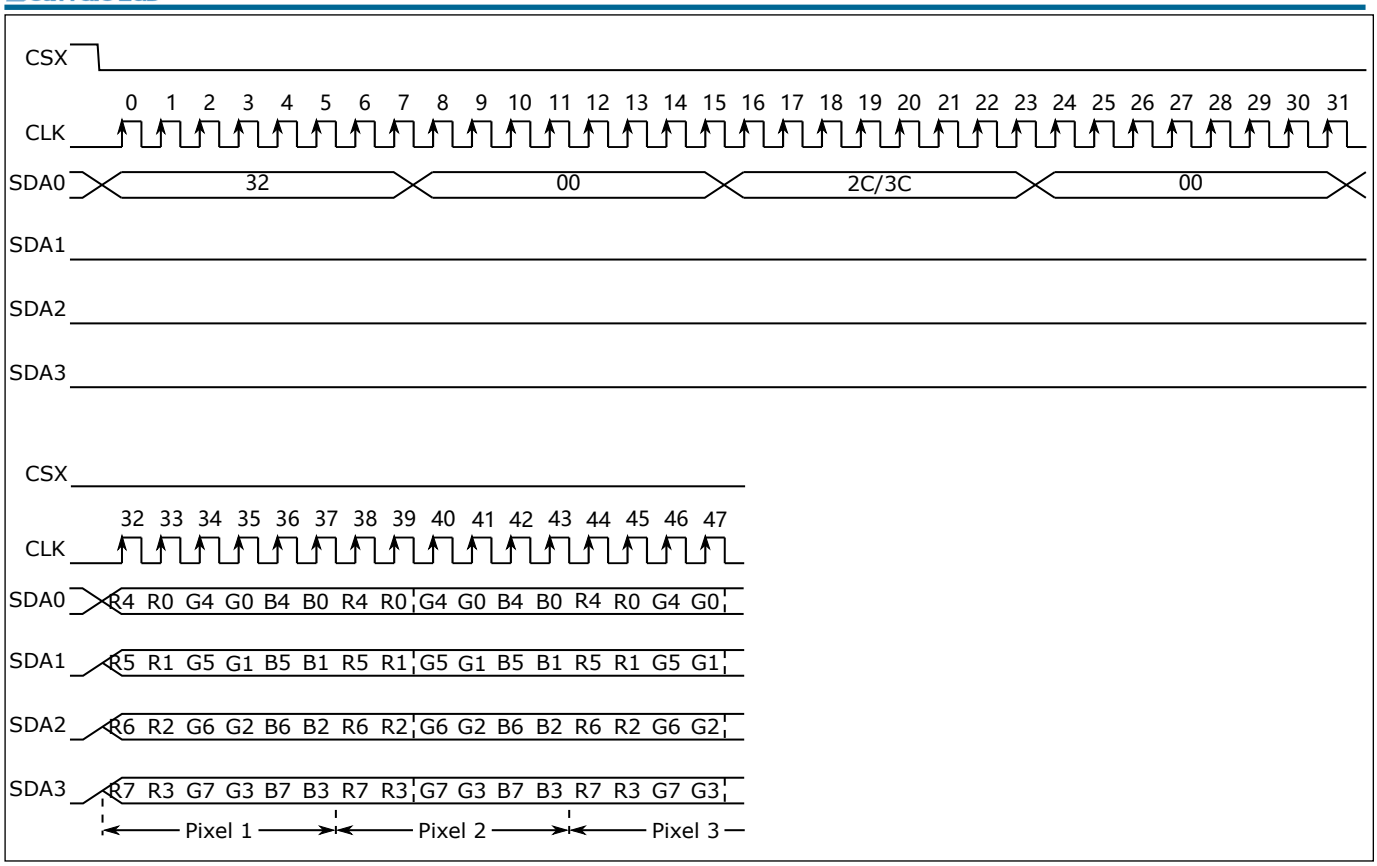


Fig. 21.24: RGB888 output

21.3.4.5 1-Wire Command, 4-Wire Address and 4-Wire Data Pattern

Taking the command 0x12 and 3-byte address 0x002C00/0x003C00 as example, the output of RGB565 data under 1-wire command, 4-wire address and 4-wire data pattern is shown as follows:

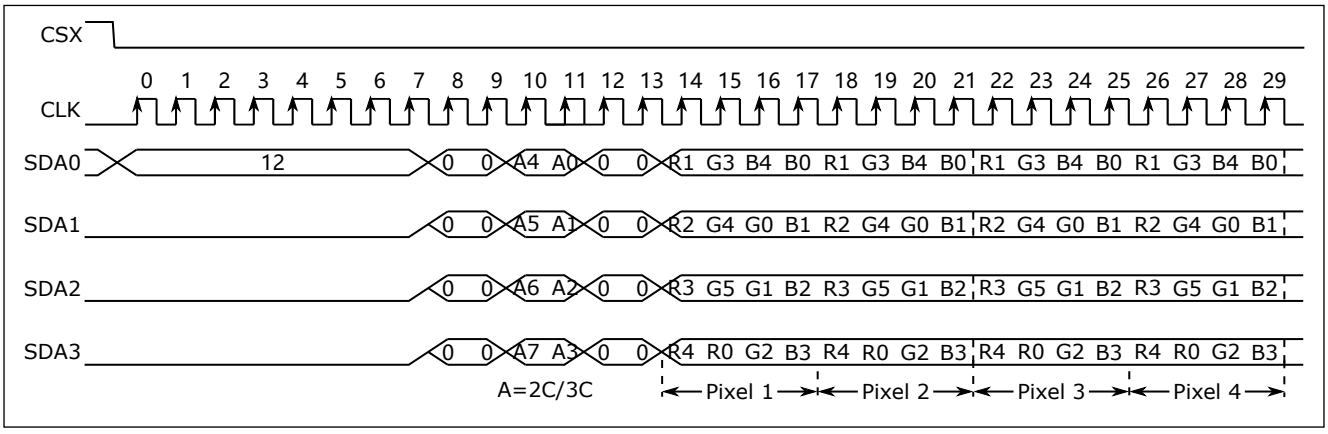


Fig. 21.25: RGB565 output

The data output in RGB666 format is shown as follows:

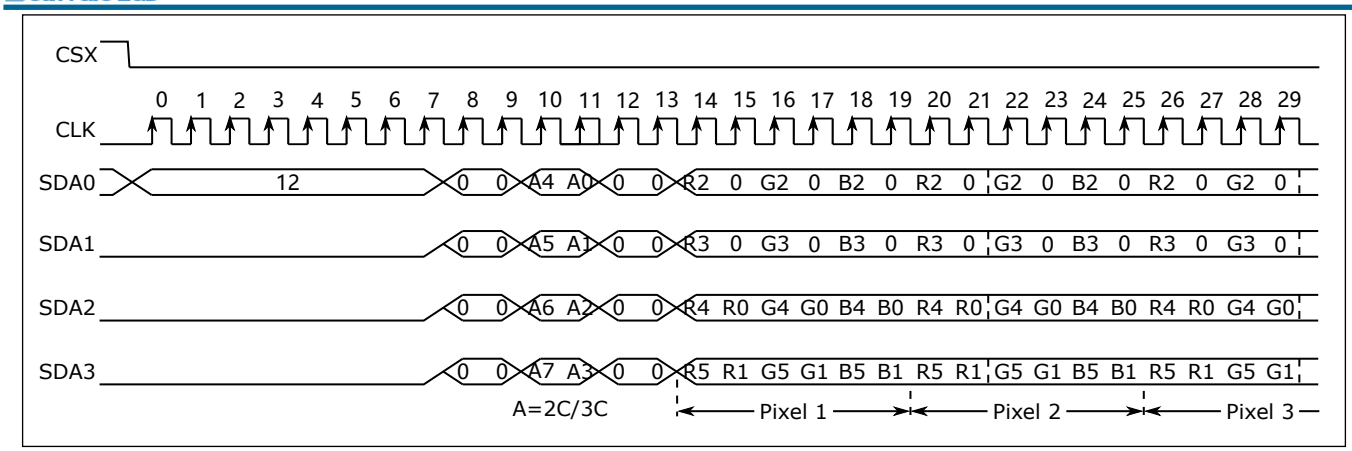


Fig. 21.26: RGB666 output

The data output in RGB888 format is shown as follows:

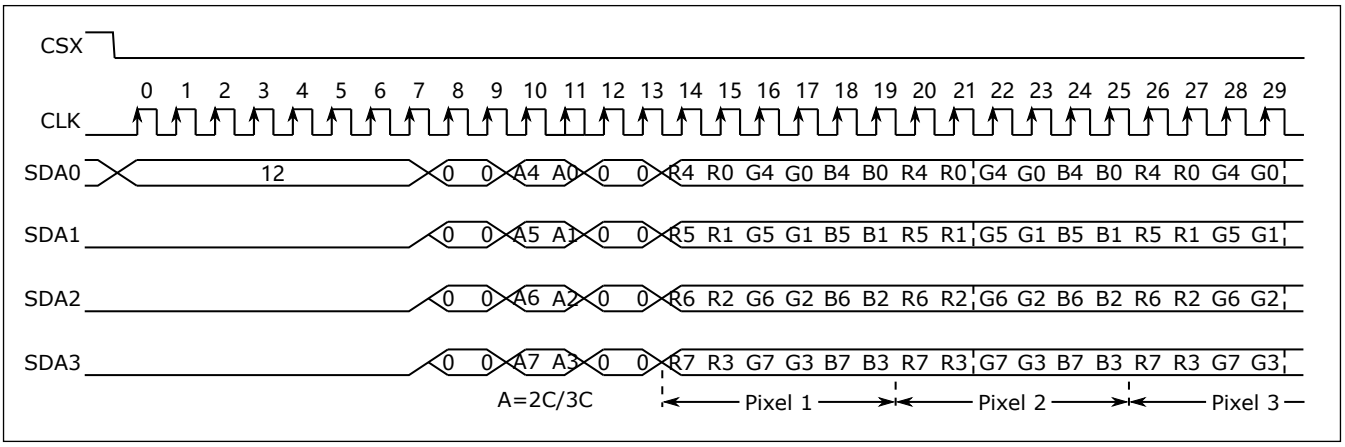


Fig. 21.27: RGB888 output

21.3.5 Input pixel format

The DBI module supports eight different input pixel RGB formats and six YUV444 formats. The RGB arrangements in the memory are shown as follows:

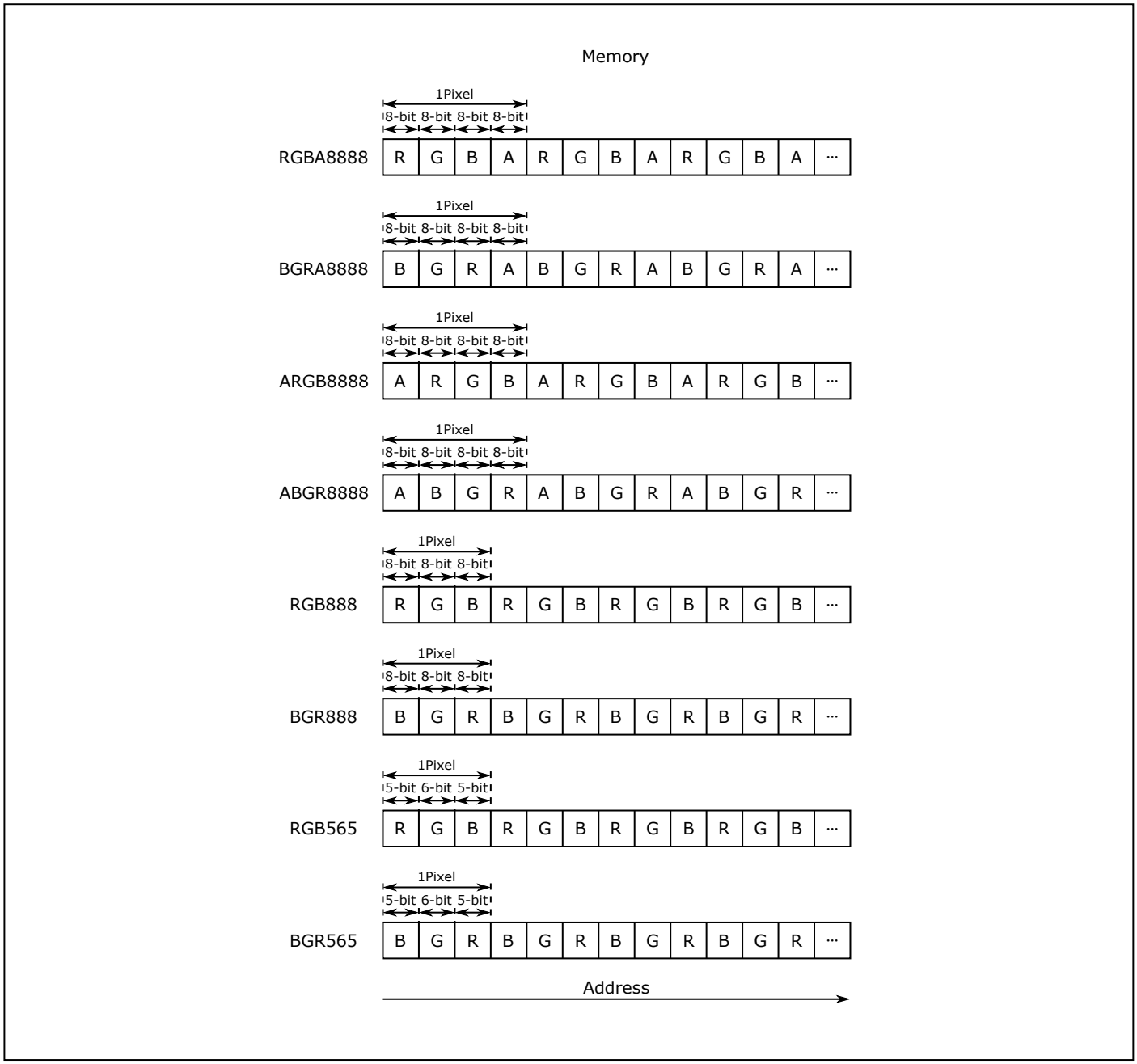


Fig. 21.28: Input Pixel RGB Format

For the YUV444 arrangement in the memory, just replace R with Y, G with U, and B with V.

21.3.6 Configuration of CS Signal Pull-up Release Condition

In Type B and Type C modes, the CS signal pull-up release condition can be configured by the CONT_EN bit in the register CONFIG. If this function cannot be enabled, the CS signal is released once after each pixel is transmitted. That is, the CS signal is pulled up every two pixels. If it is enabled, the CS signal will not be released during a single transmission. The CS signal will be pulled up and released only after this transmission is completed.

In QSPI mode, the CS signal pull-up release condition can be configured by the CS_STRETCH bit in the register CONFIG. If this function is disabled, during a single transmission, when FIFO is empty (that is, all the data in FIFO is sent out and no new data is filled in), the CS signal will be pulled up and released. If it is enabled, during a single transmission, when FIFO is empty (that is, all the data in FIFO is sent out and no new data is filled in), the CS signal will remain at a low level and wait for new data to be filled in.

21.3.7 Interrupt

DBI has the following interrupts:

- end of transfer interrupt
- TX FIFO request interrupt
- FIFO overrun error interrupt

By setting a pixel count value, the transmission end interrupt will be generated when the transmitted pixel data reaches the count value, and both Type B and Type C will generate this interrupt;

When TFICNT in DBI_FIFO_CONFIG_1 is greater than TFITH, a TX FIFO request interrupt is generated, and the interrupt flag will be automatically cleared when the condition is not met;

The FIFO overflow error interrupt will be generated when Overflow or Underflow occurs in the TX.

21.3.8 DMA

DBI TX supports DMA transfer mode, which requires setting the TX FIFO threshold through bit <TFITH> of register DBI_FIFO_CONFIG_1. When this mode is enabled, if TFICNT is greater than TFITH, a DMA TX request will be triggered. After the DMA is configured, when the DMA receives the request, it will transfer the data from the memory to the TX FIFO according to the settings.

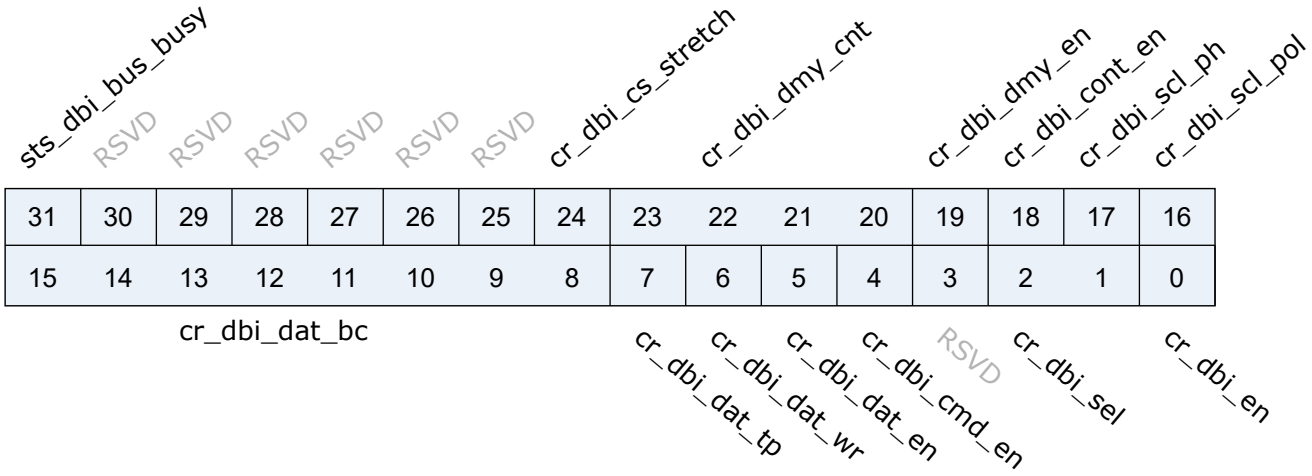
21.4 Register description

Name	Description
dbi_config	DBI configure
qspi_config	QSPI configure
dbi_pix_cnt	Pixel format and count

Name	Description
dbi_prd	DBI period
dbi_cmd	DBI command
dbi_qspi_adr	QSPI address
dbi_rdata_0	Read data 0
dbi_rdata_1	Read data 1
dbi_int_sts	Interrupt status and setting
dbi_yuv_rgb_config_0	YUV2RGB configure0
dbi_yuv_rgb_config_1	YUV2RGB configure1
dbi_yuv_rgb_config_2	YUV2RGB configure2
dbi_yuv_rgb_config_3	YUV2RGB configure3
dbi_yuv_rgb_config_4	YUV2RGB configure4
dbi_yuv_rgb_config_5	YUV2RGB configure5
dbi_fifo_config_0	FIFO format and DMA mode
dbi_fifo_config_1	FIFO threshold and available count
dbi_fifo_wdata	TX FIFO

21.4.1 dbi_config

Address：0x2000a800

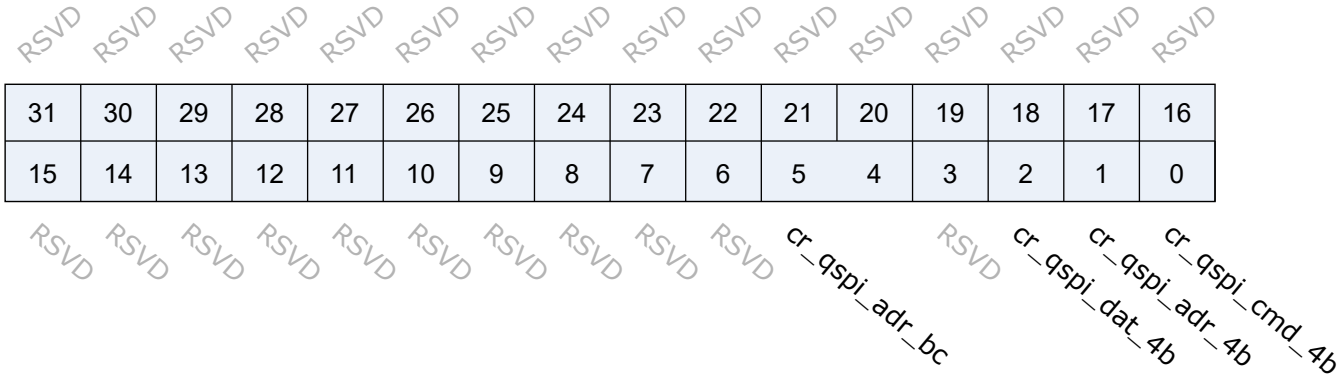


Bits	Name	Type	Reset	Description
31	sts_dbi_bus_busy	r	1'b0	Indicator of dbi bus busy
30:25	RSVD			
24	cr_dbi_cs_stretch	r/w	1'b0	Enable signal of CS-low stretch mode 1'b0: Disabled, if FIFO is empty during a pixel data transfer, CS will de-assert before FIFO is filled again and new transfer starts 1'b1: Enabled, if FIFO is empty during a pixel data transfer, CS will stay asserted while waiting for FIFO to be filled again
23:20	cr_dbi_dmy_cnt	r/w	4'd0	Dummy cycle count, unit: period defined by dbi_prd_d Effective only in Type C (Fixed to 1 dbi_prd_d period in Type B)
19	cr_dbi_dmy_en	r/w	1'b0	Enable signal of dummy cycle(s) 1'b0: Disabled, no dummy cycle(s) between command phase and data phase 1'b1: Enabled, dummy cycle(s) will be inserted between command phase and data phase Note: Don't-care if QSPI mode is selected (no dummy cycle)
18	cr_dbi_cont_en	r/w	1'b1	Enable signal of pixel data continuous transfer mode 1'b0: Disabled, CS_n will de-assert between each pixel 1'b1: Enabled, CS_n will stay asserted between each consecutive pixel Note: Don't-care if QSPI mode is selected (always in continuous mode)
17	cr_dbi_scl_ph	r/w	1'b0	SCL clock phase inverse signal
16	cr_dbi_scl_pol	r/w	1'b1	SCL clock polarity 0: SCL output LOW at IDLE state 1: SCL output HIGH at IDLE state
15:8	cr_dbi_dat_bc	r/w	8'd0	Data byte count of normal data (pixel data count is determined by cr_dbi_pix_cnt) Note: Normal read data can only be up to 8-byte (8'd7). No limit for normal write data (can be up to 256-byte using FIFO)
7	cr_dbi_dat_tp	r/w	1'b0	Data type select 1'b0: Normal data (parameter) 1'b1: Pixel data Note: Read command supports normal data only

Bits	Name	Type	Reset	Description
6	cr_dbi_dat_wr	r/w	1'b1	Data phase Read/Write select 1'b0: Read data 1'b1: Write data
5	cr_dbi_dat_en	r/w	1'b1	Data enable signal 1'b0: Data phase disabled 1'b1: Data phase enabled
4	cr_dbi_cmd_en	r/w	1'b1	Command enable signal 1'b0: No command phase 1'b1: Command will be sent Note: Don't-care if QSPI mode is selected (Command always enabled)
3	RSVD			
2:1	cr_dbi_sel	r/w	2'd0	DBI mode select 2'd0: DBI Type B 2'd1: DBI Type C, 4-wire mode 2'd2: DBI Type C, 3-wire mode 2'd3: QSPI mode
0	cr_dbi_en	r/w	1'b0	Enable signal of DBI function Asserting this bit will trigger the transaction, and should be de-asserted after finish

21.4.2 qspi_config

Address：0x2000a804

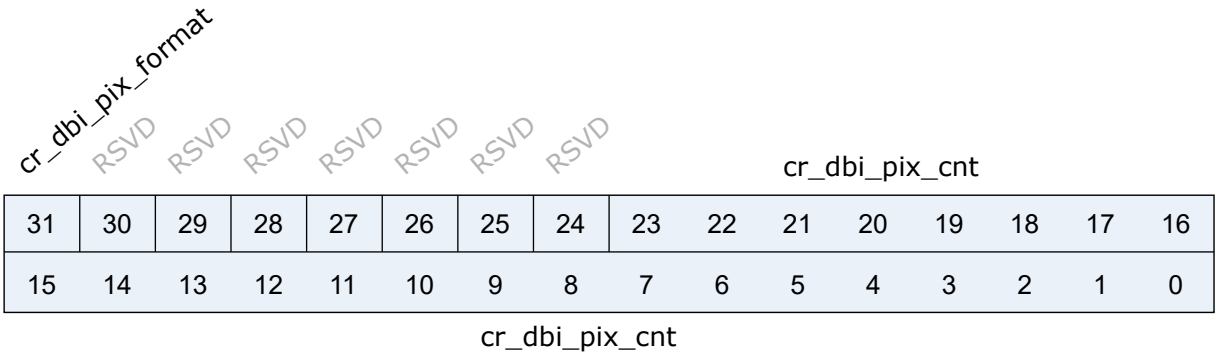


Bits	Name	Type	Reset	Description
31:6	RSVD			
5:4	cr_qspi_adr_bc	r/w	2'd2	QSPI Address byte count

Bits	Name	Type	Reset	Description
3	RSVD			
2	cr_qspi_dat_4b	r/w	1'b1	QSPI Data 4-bit (quad) mode 1'b0: Data sent/received in 1-bit mode 1'b1: Data sent/received in 4-bit mode
1	cr_qspi_adr_4b	r/w	1'b1	QSPI Address (display command) 4-bit (quad) mode 1'b0: Address sent in 1-bit mode 1'b1: Address sent in 4-bit mode
0	cr_qspi_cmd_4b	r/w	1'b0	QSPI Command 4-bit (quad) mode 1'b0: Command sent in 1-bit mode 1'b1: Command sent in 4-bit mode

21.4.3 dbi_pix_cnt

Address：0x2000a808



Bits	Name	Type	Reset	Description
31	cr_dbi_pix_format	r/w	1'b0	Pixel format 1'b0: RGB565 1'b1: RGB888/RGB666
30:24	RSVD			
23:0	cr_dbi_pix_cnt	r/w	24'h0	Pixel count Note: Should be a multiple of 2 if RGB565 is selected and a multiple of 4 if RGB888/RGB666 mode is selected

21.4.4 dbi_prd

Address：0x2000a80c

cr_dbi_prd_d_ph_1								cr_dbi_prd_d_ph_0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cr_dbi_prd_i								cr_dbi_prd_s							

Bits	Name	Type	Reset	Description
31:24	cr_dbi_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 (please refer to "Timing" tab)
23:16	cr_dbi_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (please refer to "Timing" tab)
15:8	cr_dbi_prd_i	r/w	8'd15	Length of INTERVAL between pixel data (please refer to "Timing" tab)
7:0	cr_dbi_prd_s	r/w	8'd15	Length of START/STOP condition (please refer to "Timing" tab)

21.4.5 dbi_cmd

Address：0x2000a810

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	cr_dbi_cmd							

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	cr_dbi_cmd	r/w	8'h2C	DBI Command

21.4.6 dbi_qspi_adr

Address：0x2000a814

cr_qspi_adr

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

cr_qspi_adr

Bits	Name	Type	Reset	Description
31:0	cr_qspi_adr	r/w	32'h00002C00	QSPI Address (display command) Note: LSB is sent first

21.4.7 dbi_rdata_0

Address：0x2000a818

sts_dbi_rdata_0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_dbi_rdata_0

Bits	Name	Type	Reset	Description
31:0	sts_dbi_rdata_0	r	32'h0	Data read from display IC using read command

21.4.8 dbi_rdata_1

Address：0x2000a81c

sts_dbi_rdata_1

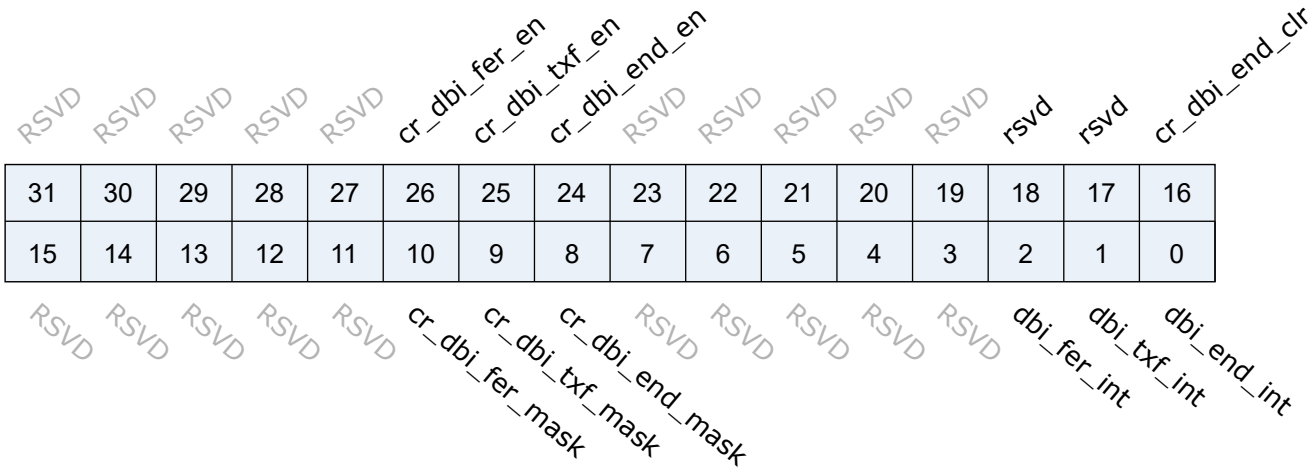
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_dbi_rdata_1

Bits	Name	Type	Reset	Description
31:0	sts_dbi_rdata_1	r	32'h0	Data read from display IC using read command

21.4.9 dbi_int_sts

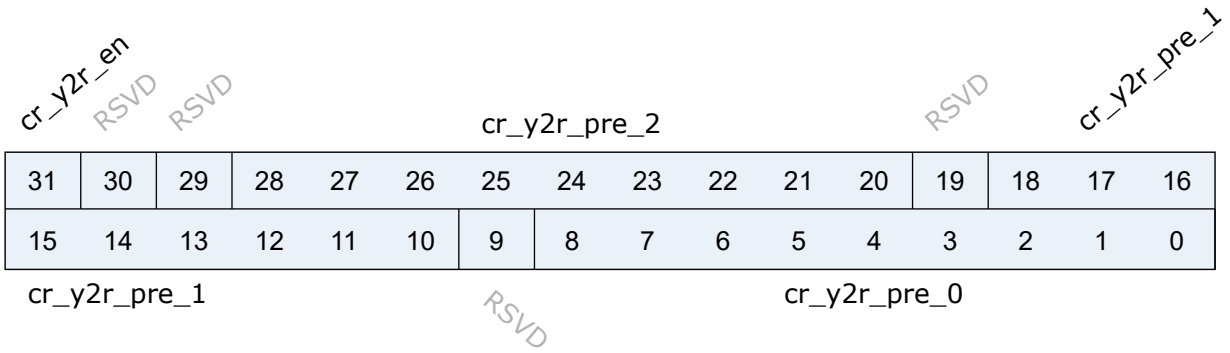
Address：0x2000a830



Bits	Name	Type	Reset	Description
31:27	RSVD			
26	cr_dbi_fer_en	r/w	1'b1	Interrupt enable of dbi_fer_int
25	cr_dbi_txf_en	r/w	1'b1	Interrupt enable of dbi_txe_int
24	cr_dbi_end_en	r/w	1'b1	Interrupt enable of dbi_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_dbi_end_clr	w1c	1'b0	Interrupt clear of dbi_end_int
15:11	RSVD			
10	cr_dbi_fer_mask	r/w	1'b1	Interrupt mask of dbi_fer_int
9	cr_dbi_txf_mask	r/w	1'b1	Interrupt mask of dbi_txe_int
8	cr_dbi_end_mask	r/w	1'b1	Interrupt mask of dbi_end_int
7:3	RSVD			
2	dbi_fer_int	r	1'b0	TX/RX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	dbi_txf_int	r	1'b1	TX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	dbi_end_int	r	1'b0	Transfer end interrupt, shared by both Type B and C mode

21.4.10 dbi_yuv_rgb_config_0

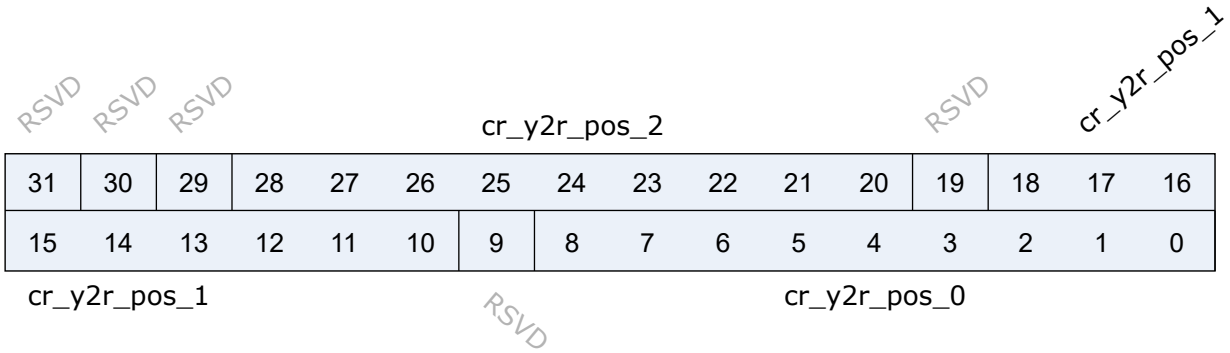
Address：0x2000a860



Bits	Name	Type	Reset	Description
31	cr_y2r_en	r/w	1'b1	Display module YUV2RGB enable signal
30:29	RSVD			
28:20	cr_y2r_pre_2	r/w	9'd0	Display module YUV2RGB "pre_offset_2" parameter
19	RSVD			
18:10	cr_y2r_pre_1	r/w	9'd0	Display module YUV2RGB "pre_offset_1" parameter
9	RSVD			
8:0	cr_y2r_pre_0	r/w	9'd0	Display module YUV2RGB "pre_offset_0" parameter

21.4.11 dbi_yuv_rgb_config_1

Address：0x2000a864

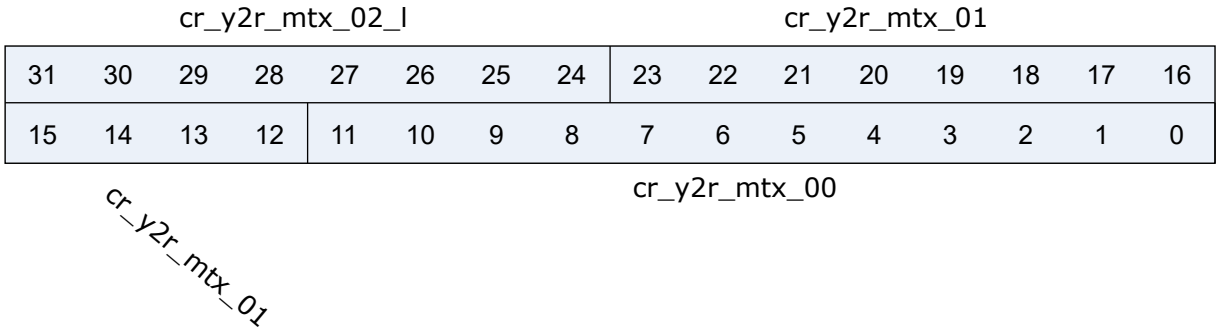


Bits	Name	Type	Reset	Description
31:29	RSVD			
28:20	cr_y2r_pos_2	r/w	9'd0	Display module YUV2RGB "post_offset_2" parameter

Bits	Name	Type	Reset	Description
19	RSVD			
18:10	cr_y2r_pos_1	r/w	9'd0	Display module YUV2RGB "post_offset_1" parameter
9	RSVD			
8:0	cr_y2r_pos_0	r/w	9'd0	Display module YUV2RGB "post_offset_0" parameter

21.4.12 dbi_yuv_rgb_config_2

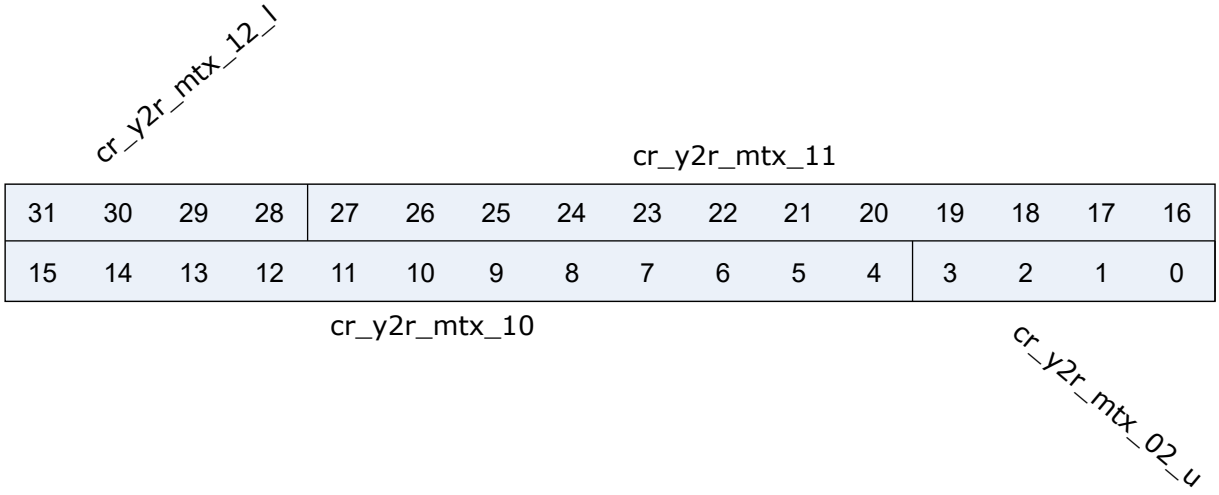
Address：0x2000a868



Bits	Name	Type	Reset	Description
31:24	cr_y2r_mtx_02_l	r/w	8'h0	Display module YUV2RGB "matrix_02" parameter lower bits
23:12	cr_y2r_mtx_01	r/w	12'h0	Display module YUV2RGB "matrix_01" parameter
11:0	cr_y2r_mtx_00	r/w	12'h0	Display module YUV2RGB "matrix_00" parameter

21.4.13 dbi_yuv_rgb_config_3

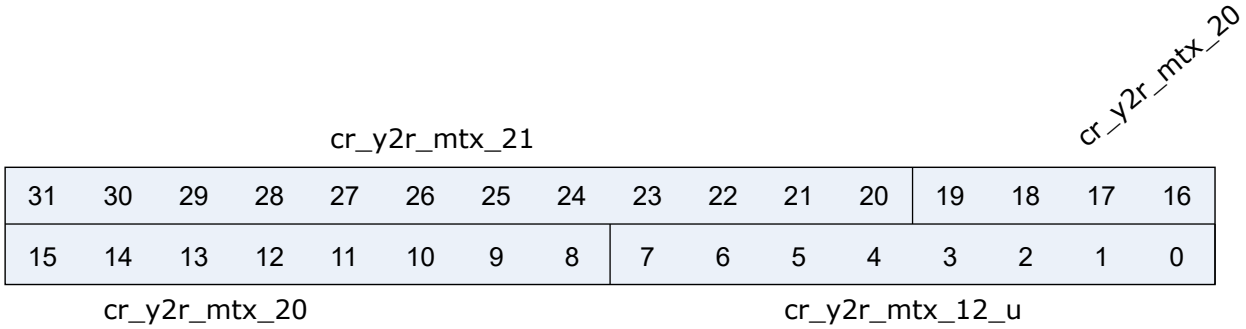
Address：0x2000a86c



Bits	Name	Type	Reset	Description
31:28	cr_y2r_mtx_12_l	r/w	4'h0	Display module YUV2RGB "matrix_12" parameter lower bits
27:16	cr_y2r_mtx_11	r/w	12'h0	Display module YUV2RGB "matrix_11" parameter
15:4	cr_y2r_mtx_10	r/w	12'h0	Display module YUV2RGB "matrix_10" parameter
3:0	cr_y2r_mtx_02_u	r/w	4'h0	Display module YUV2RGB "matrix_02" parameter upper bits

21.4.14 dbi_yuv_rgb_config_4

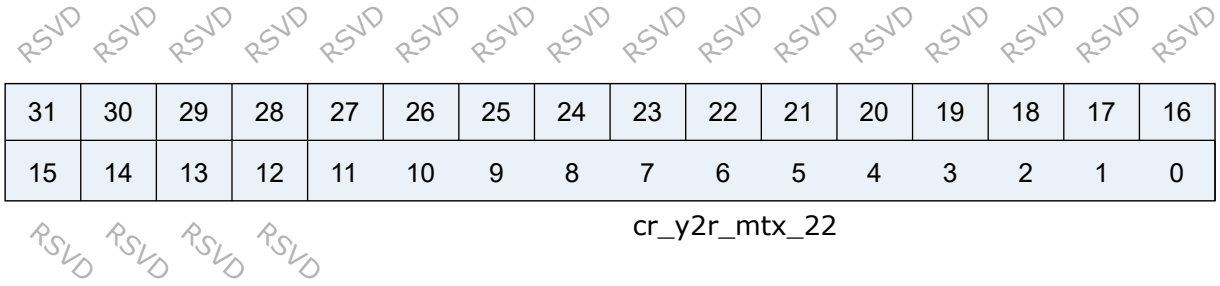
Address：0x2000a870



Bits	Name	Type	Reset	Description
31:20	cr_y2r_mtx_21	r/w	12'h0	Display module YUV2RGB "matrix_21" parameter
19:8	cr_y2r_mtx_20	r/w	12'h0	Display module YUV2RGB "matrix_20" parameter
7:0	cr_y2r_mtx_12_u	r/w	8'h0	Display module YUV2RGB "matrix_12" parameter upper bits

21.4.15 dbi_yuv_rgb_config_5

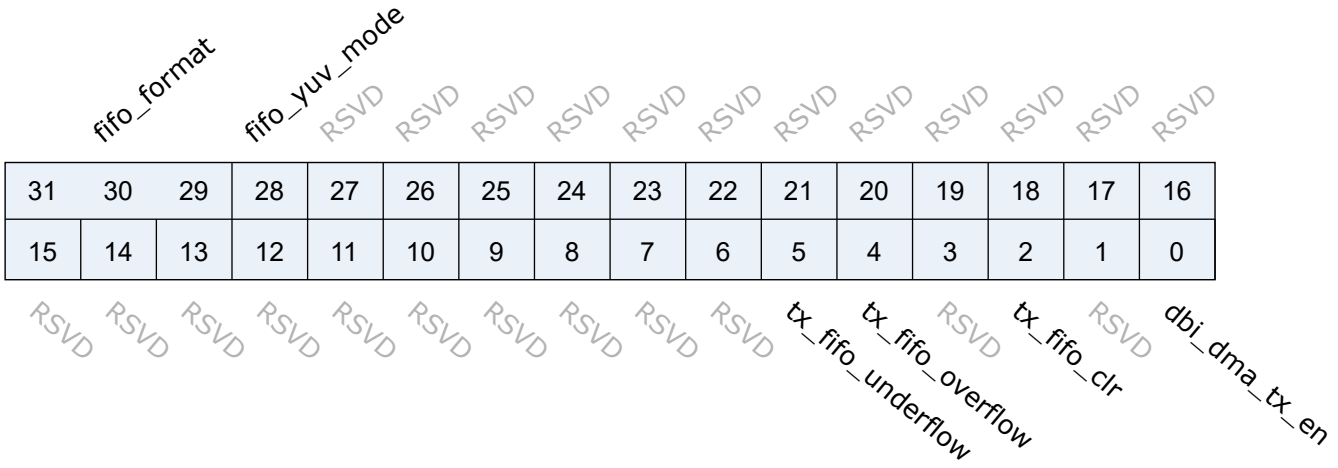
Address：0x2000a874



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:0	cr_y2r_mtx_22	r/w	12'h0	Display module YUV2RGB "matrix_22" parameter

21.4.16 dbi_fifo_config_0

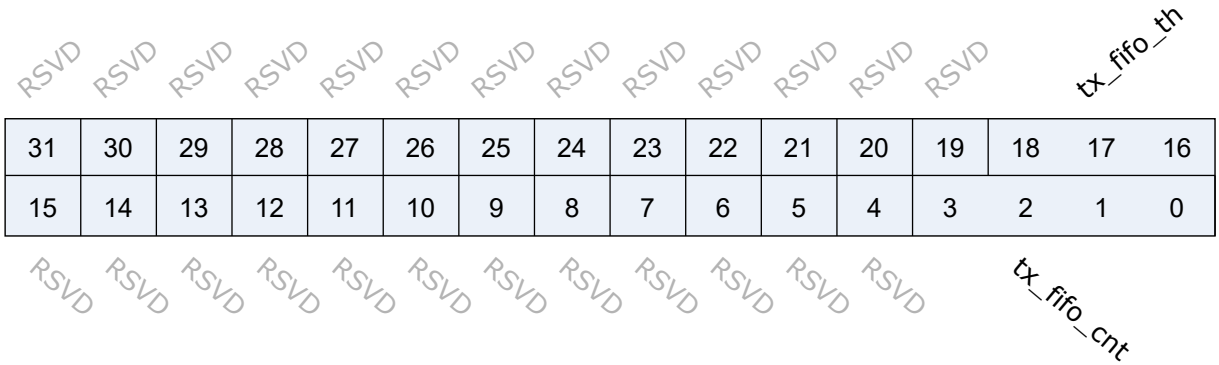
Address：0x2000a880



Bits	Name	Type	Reset	Description
31:29	fifo_format	r/w	3'd0	FIFO pixel data format (see Tab 'FIFO Format' for details) 3'd0: 8'h0, B[7:0], G[7:0], R[7:0] 3'd1: 8'h0, R[7:0], G[7:0], B[7:0] 3'd2: B[7:0], G[7:0], R[7:0], 8'h0 3'd3: R[7:0], G[7:0], B[7:0], 8'h0 3'd4: R_n[7:0], B[7:0], G[7:0], R[7:0] 3'd5: B_n[7:0], R[7:0], G[7:0], B[7:0] 3'd6: 2B[7:3], G[7:2], R[7:3] 3'd7: 2R[7:3], G[7:2], B[7:3] Note: FIFO data format does not affect normal data, which is fixed to LSB sent first Byte3[7:0], Byte2[7:0], Byte1[7:0], Byte0[7:0]
28	fifo_yuv_mode	r/w	1'b0	FIFO data YUV mode R <-> Y G <-> U/Cb B <-> V/Cr
27:6	RSVD			
5	tx_fifo_underflow	r	1'b0	Underflow flag of TX FIFO, can be cleared by tx_fifo_clr
4	tx_fifo_overflow	r	1'b0	Overflow flag of TX FIFO, can be cleared by tx_fifo_clr
3	RSVD			
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	RSVD			
0	dbi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

21.4.17 dbi_fifo_config_1

Address：0x2000a884



Bits	Name	Type	Reset	Description
31:19	RSVD			
18:16	tx_fifo_th	r/w	3'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_-fifo_cnt is less than this value
15:4	RSVD			
3:0	tx_fifo_cnt	r	4'd8	TX FIFO available count

21.4.18 dbi_fifo_wdata

Address：0x2000a888

dbi_fifo_wdata

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dbi_fifo_wdata

Bits	Name	Type	Reset	Description
31:0	dbi_fifo_wdata	w	x	Pixel data with 8 types of format (determined by fifo_format)

22 SDH

22.1 Overview

The SD Host Controller (SDH) is used to control the communication with SDIO or SD card, and all accesses are set through standard control registers.

22.2 Features

- Comply with the SD Host Controller Specification defined by the SD Association
- Suitable for SDIO/SD memory card
- Supports the standard register settings for the SDH
- Supports the DMA operation for high-speed data transfer
- Supports interrupt
- Supports data block transmission

22.3 Functional Description

22.3.1 SDH Overall Structure

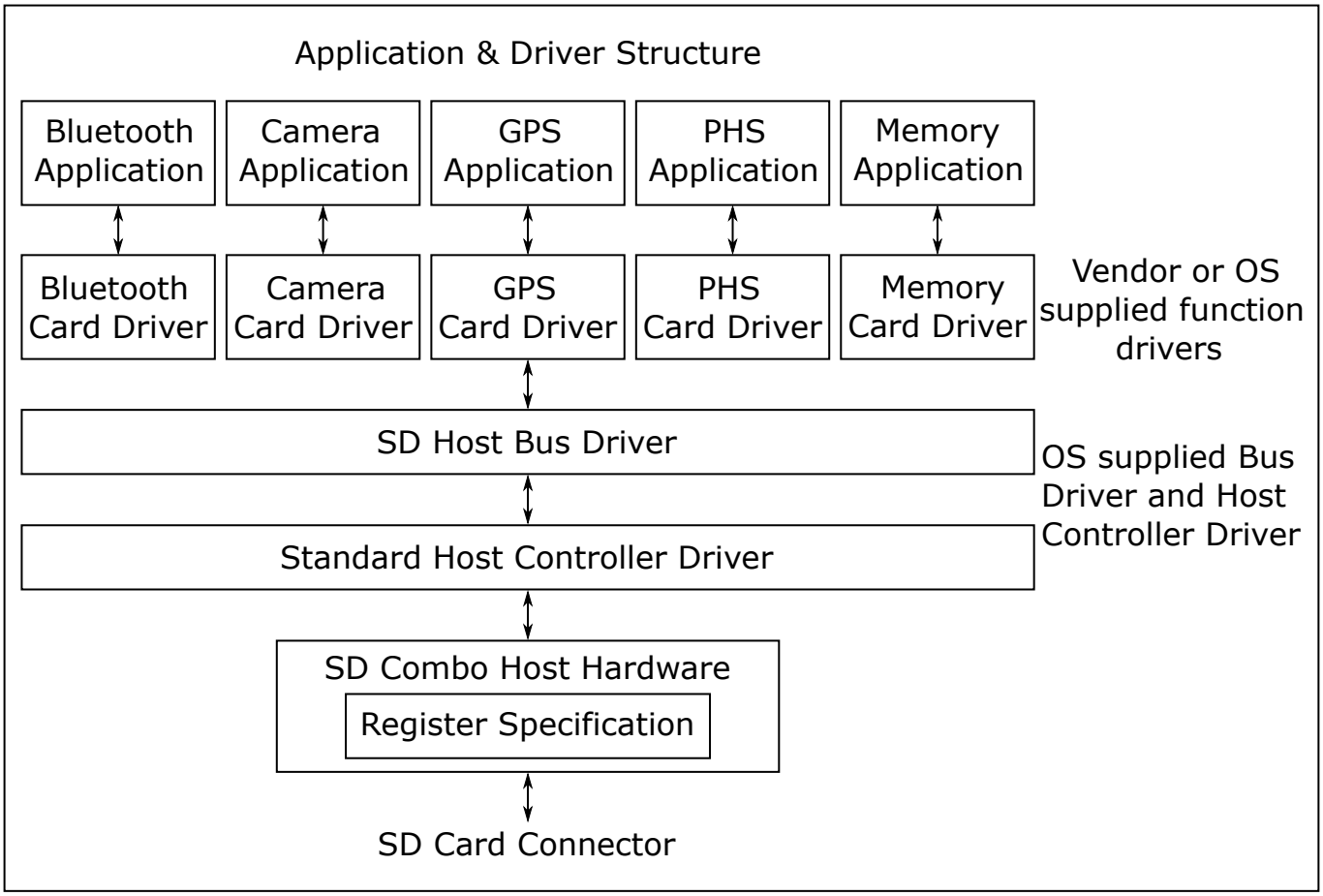


Fig. 22.1: SDH Hardware and Driver Structure

22.3.2 Register Mapping

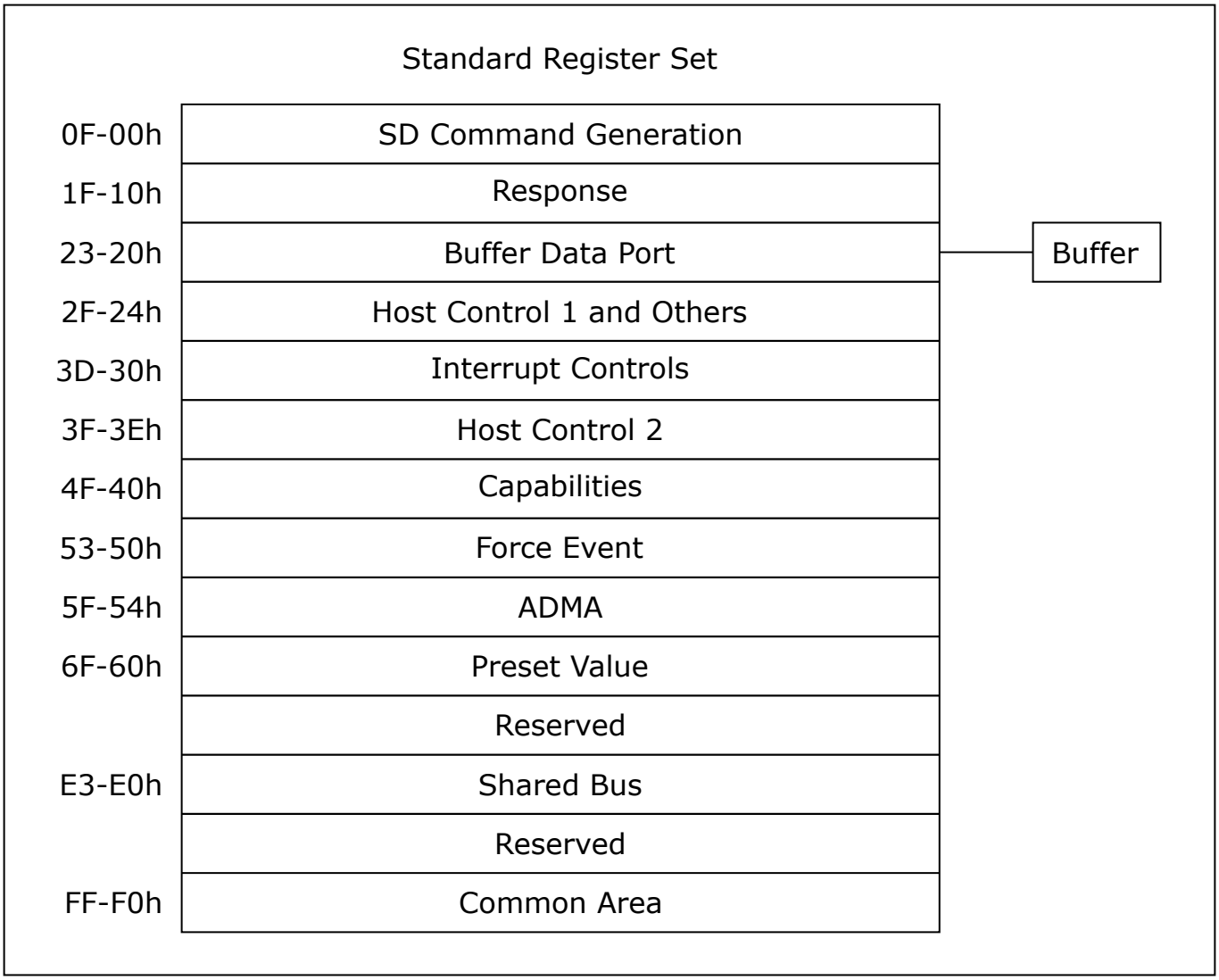


Fig. 22.2: Standard Register Mapping Classification

- SD Command Generation: parameter used to generate SD command
- Response: the response value from the card
- Buffer Data Port: the data access port of the internal buffer
- Host Control 1 and Others: current status, SD bus control, host reset, etc.
- Interrupt Controls: Interrupt Status and Interrupt Enable
- Host Control 2: vendor-specific host controller support information
- Capabilities: expansion of host control register
- Force Event: a test register that generates events through software

- ADMA: advanced DMA register
- Preset Value: preset values for clock frequency selection and driving strength selection
- Shared Bus: device control of shared bus system
- Common Area: public information area

22.3.3 Support for Multiple Card Slots

A standard register set is defined for each card slot. If the host controller has two card slots, two register sets are required. Each card slot is independently controlled. This makes it possible to support the combination of bus interface voltage, bus timing and SD clock frequency.

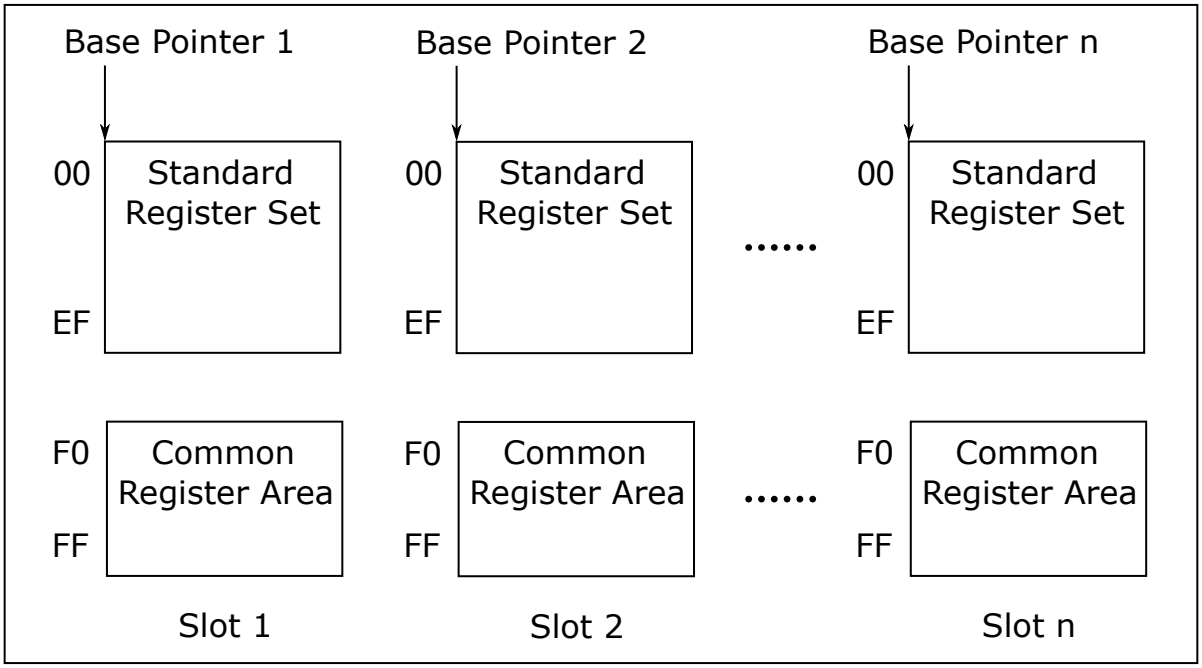


Fig. 22.3: Register Mapping of Multi-Card Slot Controller

The above figure shows the register mapping of the multi-card slot host controller. The host driver shall use PCI configuration registers or the vendor-specific methods to determine the number of card slots and the base pointer to the standard register set of each card slot. The offset from 0F0h to 0FFh is reserved for the common register area, which defines the information of card slot control and public state. The common register area can be accessed from the register set of any card slot. This allows the software to control each card slot independently, because it can access the Card Slot Interrupt Status register and the Host Controller Version register from each register set.

22.3.4 Supporting DMA

The host controller provides a "programmed I/O" method for the host driver to transfer data using the Buffer Data Port register. Optionally, host controller implementers may support data transfer using DMA. Prior to using DMA, the host driver shall confirm that both the host controller and the system bus support it (PCI bus can support DMA). DMA shall support both single block and multiple-block transfers. The host controller registers shall remain accessible for issuing non-DAT line commands during a DMA transfer execution. The result of a DMA transfer shall be the same regardless of the system bus data transfer method. The host driver can stop and restart a DMA operation by the control bits in the block gap control register. By setting Stop At Block Gap Request, a DMA operation can be stopped at block gap. By setting Continue Request, DMA operation can be restarted. If an error occurs, DMA operation shall be stopped. To abort DMA transfer, the host driver shall reset the host controller through the "Software Reset" of DAT line in the software reset register, and issue CMD12 when executing read/write commands on multiple blocks.

22.3.5 SD Command Generation

	SDMA Command	ADMA Command	CPU Data Transfer	Non-DAT Transfer
SDMA System Address /Argument 2	Yes /No	No /Auto CMD23	No /Auto CMD23	No /No
Block Size	Yes	Yes	Yes	No (Protected)
Block Count	Yes	Yes	Yes	No (Protected)
Argument 1	Yes	Yes	Yes	Yes
Transfer Mode	Yes	Yes	Yes	No (Protected)
Command	Yes	Yes	Yes	Yes

Fig. 22.4: Register for Generating SD Commands

The table above shows the register settings (offset from 000h to 00Fh in a register set) required for three types of transactions: DMA-generated transfer (SDMA or ADMA), CPU data transfer (using "programmed I/O") and non-DAT transfer. When starting a transaction, the host driver shall program these registers from 000h to 00Fh. The offset of the starting register can be calculated based on the transaction type. The last written offset shall always be 00Fh, because the high byte written to the command register will trigger the SD command. During a data transaction, the host driver shall not read the SDMA system address, block size and block count registers, unless transfer is stopped or suspended because the values are changing and unstable. To prevent data transfer from damaging registers when commands are issued, the block size, block count and transfer mode registers shall be write-protected by the host controller, and the command disable (DAT) shall be set to 1 in the current status register (this signal cannot protect the SDMA system address). When command disable (CMD) is set to 1, the host driver must not write parameter 1

and command register.

22.3.6 Suspend and Resume Mechanism

Support for Suspend/Resume can be determined by checking Suspend/Resume Support in the Capabilities register. When the SD card accepts a suspend request, the host driver saves information in the first 14 bytes registers (that is, offsets 000h-00dh) before issuing other SD commands. On resuming, the host driver restores these registers and then issues the Resume command to continue suspended operation. The SDIO card sets the DF (Resume Data Flag) in the response to the Resume command. (Since the Suspend and Resume commands are CMD52 operations, the response data is actually the Function Select Register in the CCCR.) If DF is set to 0, it means that the SDIO card cannot continue data transfer while suspended. This bit can be used to control data transfers and interrupt cycles. If the Resume Data Flag is set to 0, no more data is transferred and an interrupt cycle is started if the transaction being resumed is in 4-bit mode. If DF is set to 1, data transfers continue. It is worth noting that to use Suspend/Resume function, the SDIO card must support Suspend and Resume commands and Read Wait control.

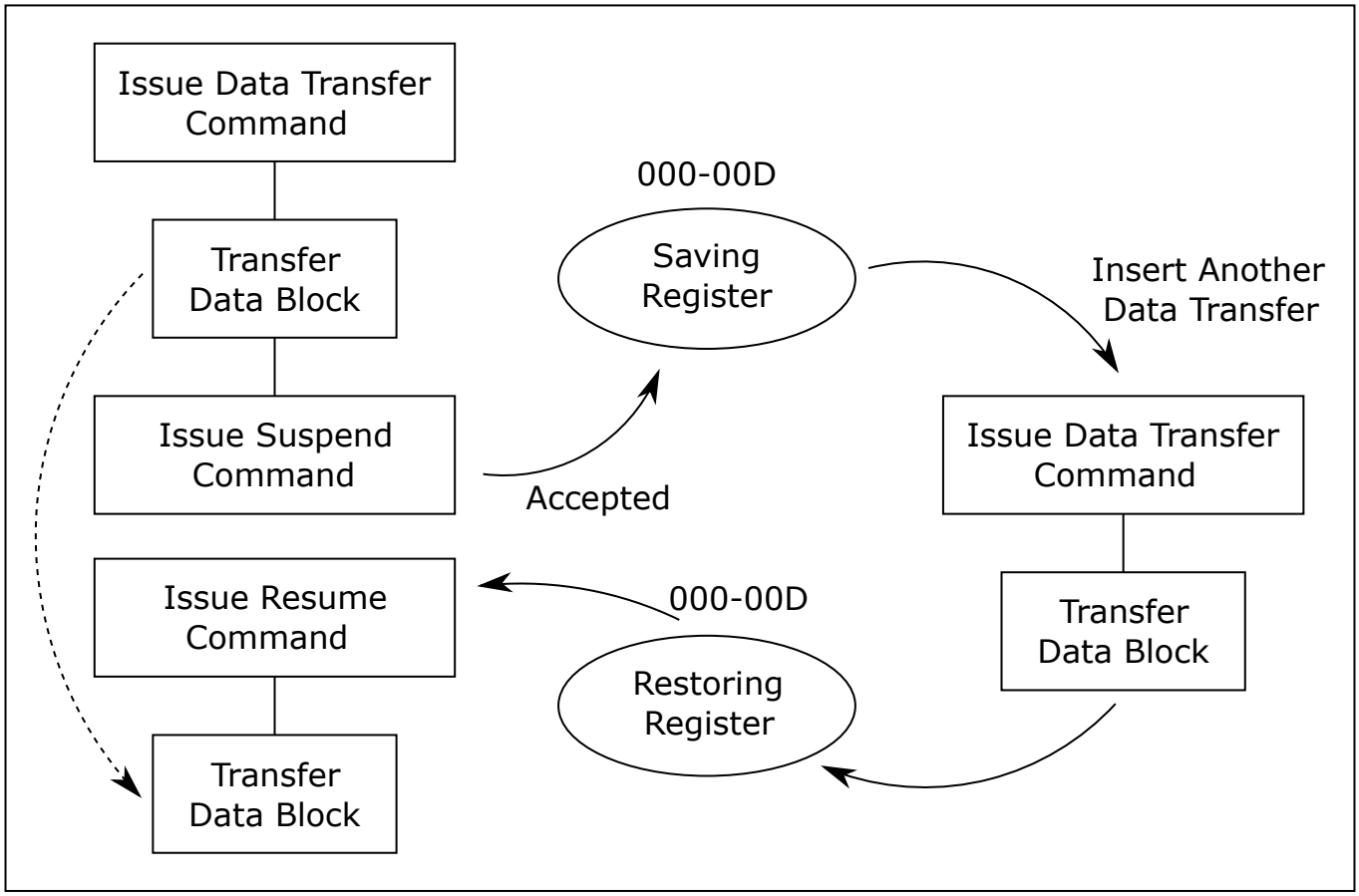


Fig. 22.5: Suspend and Resume Mechanism

22.3.7 Buffer Control

The host controller has a data buffer for data transfer. The host driver accesses internal buffer through the 32-bit Buffer Data Port register. Here are some rules for accessing the buffer.

22.3.7.1 Control of Buffer Pointer

Internally, the host controller maintains a pointer to control the data buffer. The pointer is not directly accessible by the host driver. Every time the Buffer Data Port register is accessed, the pointer is incremented depending on amount of data written to the buffer. In order to accommodate a variety of system buses, this pointer shall be implemented regardless of system bus width (8/16/32/64-bit system bus width can be supported).

22.3.7.2 Determination of Buffer Block Length

To transfer data blocks at a burst, the relationship between host controller and SD card buffer sizes is very important. The host driver shall use the same data block length for both host controller and SD card. If the controller and card buffer sizes are different, the host driver shall use the smaller one. The maximum host controller buffer size is defined by the Max Block Length field in the Capabilities register.

22.3.7.3 Dividing Large Data Transfer

The SDIO command CMD53 defines the maximum data size for transfer according to the following formula: Maximum data size = block size x block count. For example, the block size is specified by the buffer size, and the maximum value of block count is 512 (9-bit count), as specified in the command parameter CMD53. In the worst case, if the card only has a 1-byte buffer, CMD53 can be used to transfer 512 bytes of data at most (block size = 1, block count = 512). If the card does not support the multi-block mode, only one byte can be transferred. If an application or card driver wants to transfer data with a larger size, the host driver shall divide the large-size data into multiple CMD53 blocks.

22.3.8 Relationship Between Interrupt Control Registers

The host controller implements a number of interrupt sources. Interrupt sources can be enabled as interrupts or as system wakeup signals. If the interrupt source’s corresponding bit in the Normal Interrupt Status Enable or Error Interrupt Status Enable register is 1 and the interrupt becomes active, its active state is latched and made available to the host driver in the Normal Interrupt Status register or the Error Interrupt Status register. Interrupt Status shall be cleared when Interrupt Status Enable is cleared. An interrupt source with its bit set in an Interrupt Status register shall assert a system interrupt signal if its corresponding bit is also set in the Normal Interrupt Signal Enable register or the Error Interrupt Signal Enable register. Once signaled, most interrupts are cleared by writing a 1 to the associated bit in the Interrupt Status register. Card interrupts, however, shall be cleared by the card driver. If a card interrupt is generated, the host driver may clear Card Interrupt Status Enable to disable card interrupts while the card driver is

processing them. After all interrupt sources are cleared, the host driver sets it again to enable another card interrupt. Disabling the Card Interrupt Status Enable avoids generating multiple interrupts during interrupt service processing. The Wakeup Control register enables Card Interrupt, Card Insertion, or Card Removal status changes to be configured to generate a system wakeup signal. These interrupts are enabled or masked independently of the Normal Interrupt Signal Enable register. The kind of wakeup event can be read from the Normal Interrupt Status register. The interrupt signal and wakeup signal are logical ORed and shall be read from the Slot Interrupt Status register.

22.3.9 Hardware Block Diagram and Timing Part

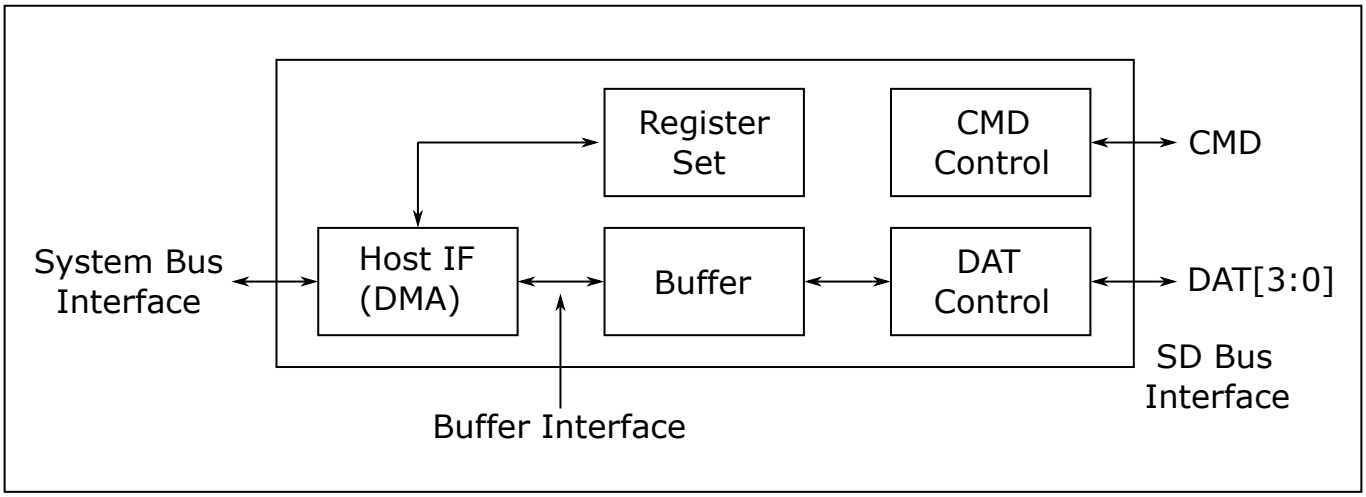


Fig. 22.6: Block Diagram of Host Controller

The host controller has two bus interfaces, the system bus interface and the SD bus interface. The host controller assumes that these interfaces are asynchronous, that is, working at different clock frequencies. The host driver is on system bus time, because it is software executed by the host controller CPU on its system clock. The SD card is on SD bus time. That is, its operation is synchronized by SDCLK. The host controller shall synchronize signals to communicate between these interfaces. Data blocks shall be synchronized at the buffer module. All status registers shall be synchronized by the system clock and maintain synchronization during output to the system interface. Control registers, which trigger SD bus transactions, shall be synchronized by SDCLK. Therefore, there will be a timing delay when propagating signals between the two interfaces. This means that the host driver cannot do real time control of the SD bus and needs to rely on the host controller to control the SD bus according to register settings. The buffer interface enables internal read and write buffers. The transfer complete interrupt status indicates completion of the read/write transfer for both DMA and non-DMA transfers. However, this timing is different between reads and writes. Read transfers shall be completed after all valid data have been transferred to the host system and are ready for the host driver to access. Write transfers shall be completed after all valid data have been transferred to the SD card and the busy state is over.

22.3.10 Auto CMD12

Multiple block transfers for SD memory require CMD12 to stop the transactions. The host controller automatically issues CMD12 when the last block transfer is completed. This feature of the host controller is called Auto CMD12. The host driver shall set Auto CMD12 Enable in the Transfer Mode register when issuing a multiple block transfer command. Auto CMD12 timing synchronization with the last data block shall be done by hardware in the host controller. Commands that do not use the DAT line can be issued during multiple block transfers. These commands are referred to using the notation CMD_wo_DAT. To prevent DAT line commands and CMD_wo_DAT commands from conflicting, the host controller shall arbitrate the timing by which each command is issued on the SD Bus. Therefore, a command might not immediately be issued after the host driver writes to the Command register. The command may be issued before or after Auto CMD12, depending on the timing. To be able to distinguish the responses of DAT line and CMD_wo_DAT commands, the Auto CMD12 response can be determined from the upper four bytes of the Response register (at offset 01Ch in the standard register set). If errors related to Auto CMD12 are detected, the host controller shall issue an Auto CMD error interrupt. The host driver can check the Auto CMD12 error status (Command Index/End bit/CRC/Timeout Error) by reading the Auto CMD Error Status register. If the Auto CMD12 was not executed, the host driver needs to recover from the CMD_wo_DAT error and issue CMD12 to stop the multiple block transfer. If the CMD_wo_DAT was not executed, the host driver can issue it again after recovering from the Auto CMD12 error. In UHS mode SDR104, the host driver shall use Auto CMD23 to stop multiple block read/write operation instead of using Auto CMD12. In other bus speed modes, if the card supports CMD23, the host driver shall use Auto CMD23 instead of using CMD12.

22.3.11 Controlling SDCLK

This table shows how SDCLK is controlled by the SD Bus Power in the Power Control register and the SD Clock Enable in the Clock Control register.

SD Bus Power	SD Clock Enable	State of SDCLK
Change 0 to 1	0	Drive Low
	1	Start Clock With Specified Period Of High
Change 1 to 0	0	Drive Low
	1	Drive Low Immediately
0	Don't Care	Drive Low
1	Change 0 to 1	Start Clock With Specified Period Of High
	Change 1 to 0	Maintains Period Of High And Then Stops Clock And Drive Low

Fig. 22.7: Controlling SDCLK by the SD Bus Power and SD Clock Enable

The clock period of SDCLK is specified by the SDCLK Frequency Select in the Clock Control register and the Base Clock Frequency For SD Clock in the Capabilities register. Since the SD card may use both clock edges, the duty

ratio of SD clock shall be average 50% (scattering within 45-55%) and the Period of High should be half of the clock period. The oscillation of SDCLK starts from driving specified Period of High. When SDCLK is stopped by the SD Clock Enable, the host controller shall stop SDCLK after driving Period of High to maintain the duty ratio of clock. When SDCLK is stopped by the SD Bus Power, the host controller shall stop SDCLK immediately (drive Low) and SD Clock Enable shall be cleared.

22.3.12 Advanced DMA

The SD Host Controller Standard Specification Version 2.00 defines the advanced DMA (ADMA) transfer algorithm. The DMA algorithm defined in the SD Host Controller Standard Specification Version 1.00 is called single DMA (SDMA). The disadvantage of SDMA is that the DMA interrupt generated at every page boundary disturbs CPU to reprogram the new system address. This SDMA algorithm develops a performance bottleneck by interrupting at each page boundary. ADMA adopts the scatter gather DMA algorithm so that higher data transfer speed is available. Before executing ADMA, the host driver can program a list of data transfers between system memory and SD card to the descriptor table. It enables ADMA to run without interrupting the host driver. In addition, ADMA supports both 32-bit system memory addressing and 64-bit system memory addressing. The 32-bit system memory addressing uses the lower 32-bit field of the 64-bit address register. ADMA is divided into ADMA1 and ADMA2. ADMA1 only supports the transfer of 4KB aligned data in the system memory. ADMA2 optimizes this restriction, so that the data of any size at any location can be transferred in the system memory. The formats of descriptor tables are different among them. In this document, the term “ADMA” refers to ADMA2.

22.3.12.1 Block Diagram of ADMA2

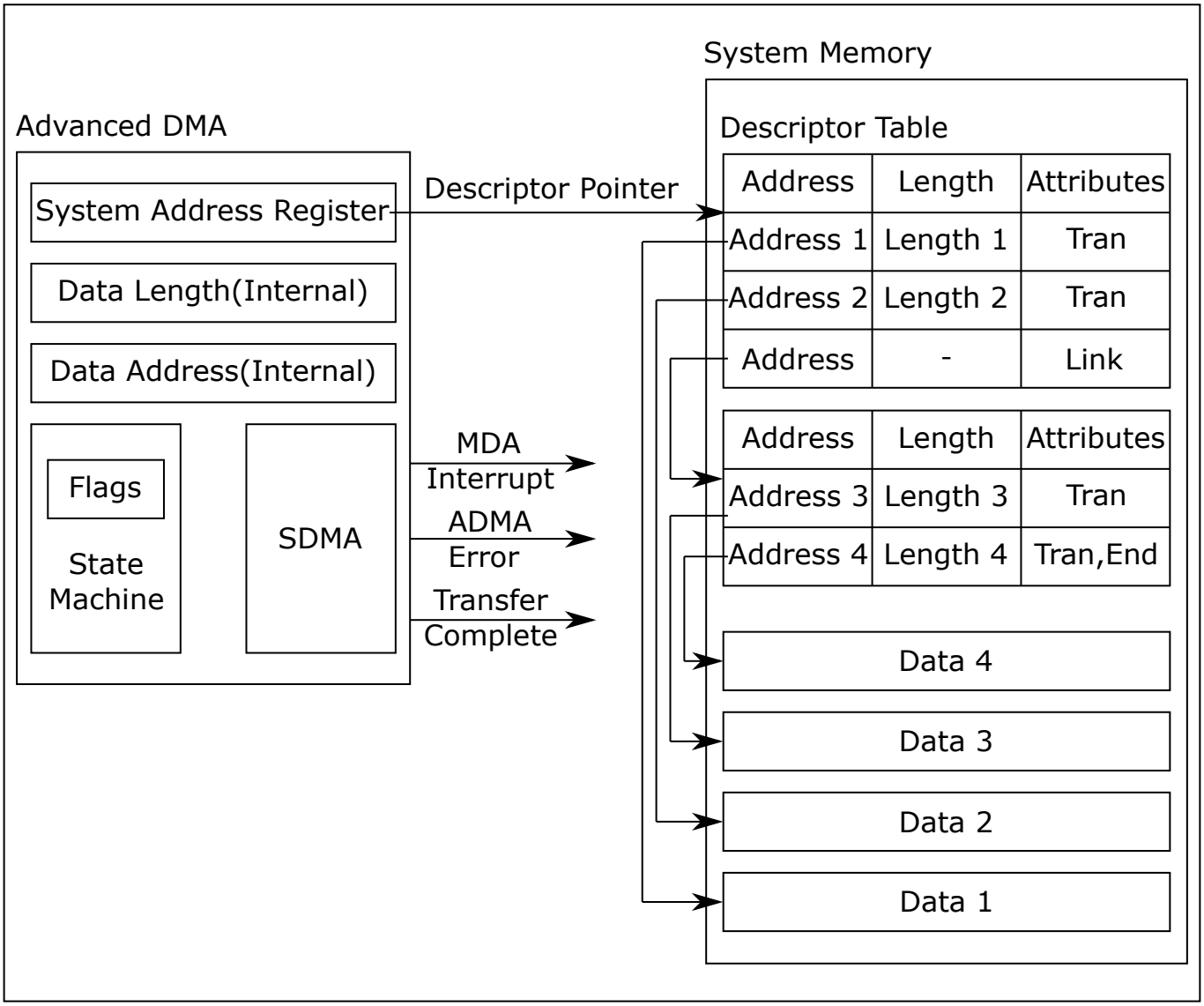


Fig. 22.8: Block Diagram of ADMA2

The descriptor table is created by the host driver in the system memory. The 32-bit address descriptor table is used for the 32-bit addressing system, and 64-bit address descriptor table is used for the 64-bit addressing system. Each descriptor line (one executable unit) consists of address, length and attribute fields. This attribute specifies the operation of the descriptor line. ADMA2 consists of SDMA, state machine and register circuit. ADMA2 uses the 64-bit advanced DMA system address register (offset 058h) as the descriptor pointer, instead of the 32-bit SDMA system address register (offset 0). Writing to the command register triggers the termination of ADMA2 transfer. ADMA2 takes a descriptor line and executes it. This process is repeated until the end of the descriptor (the attribute “end” equals to 1) is found.

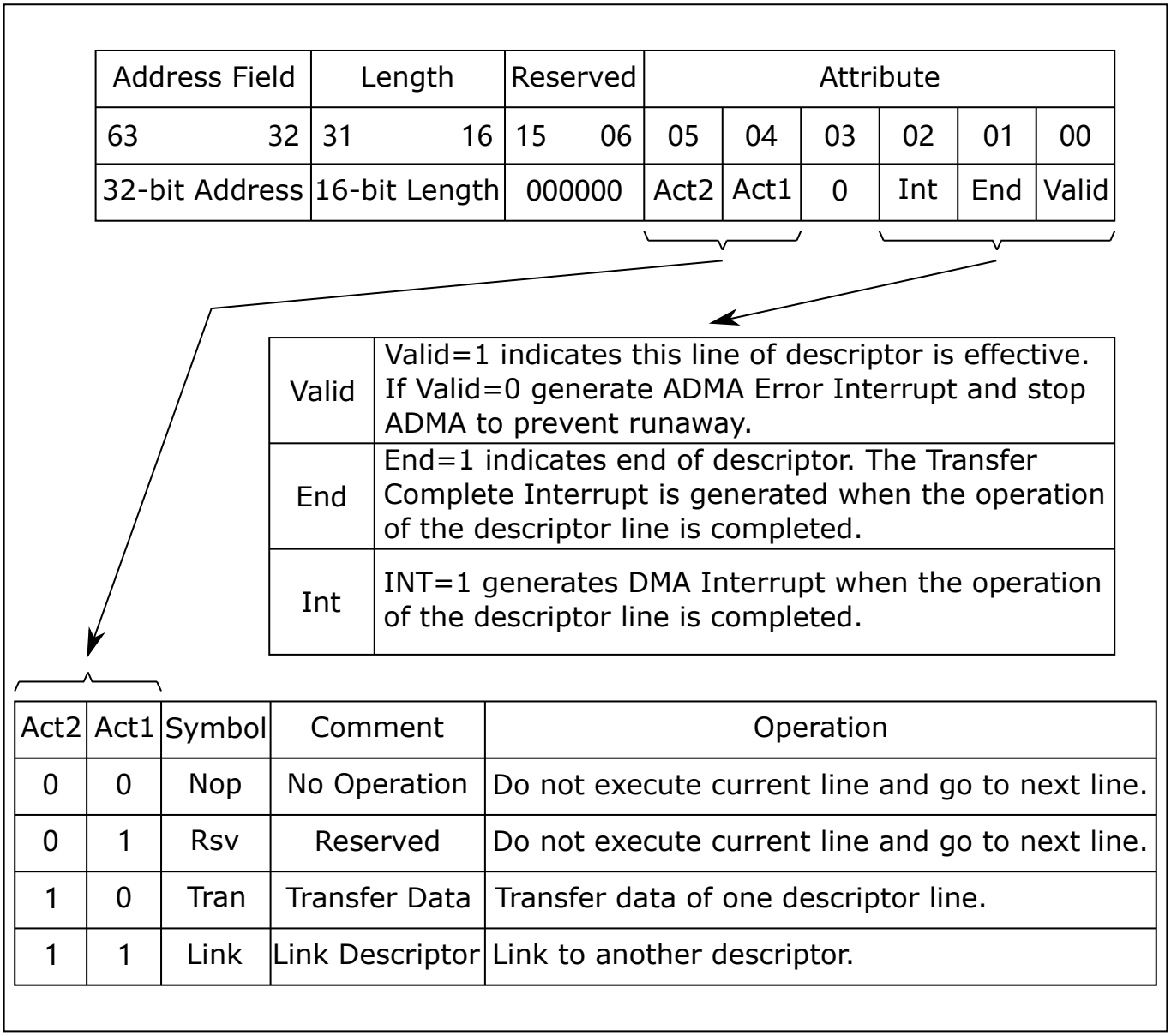
22.3.12.2 Data Address and Data Length Requirements

There are three requirements for programming descriptors:

- The minimum unit of address is 4 bytes.
- The maximum data length of each descriptor line is less than 64 KB.
- Total length = length 1 + length 2 + length 3 + ⋯+ length n = multiple of block size.

If the total length of the descriptor is not a multiple of the block size, the ADMA2 transfer may not be terminated. In this case, the transfer shall be terminated by data timeout. The block count register allows the transfer of maximum 65,535 blocks. If the ADMA2 operation is less than or equal to the transfer of 65,535 blocks, the block count register can be used. In this case, the total length of the descriptor table shall be equal to the product of the block size and the block count. If the ADMA2 operation exceeds the transfer of 65,535 blocks, the block count register shall be disabled by setting Block Count Enable to 0 in the transfer mode register. In this case, the length of data transfer is not specified by the block count but by the descriptor table. Thus, the timing of detecting the last block on the SD bus may be different, and it affects the control of read transfer activity, write transfer activity and DAT line activity in the current status register. In the case of a read operation, multiple blocks may be read. If the read operation is for the last memory area, the host driver shall ignore the OUT_OF_RANGE error.

22.3.12.3 Descriptor Table



Address Field	Length	Reserved	Attribute					
63 32	31 16	15 06	05	04	03	02	01	00
32-bit Address	16-bit Length	000000	Act2	Act1	0	Int	End	Valid

Valid	Valid=1 indicates this line of descriptor is effective. If Valid=0 generate ADMA Error Interrupt and stop ADMA to prevent runaway.
End	End=1 indicates end of descriptor. The Transfer Complete Interrupt is generated when the operation of the descriptor line is completed.
Int	INT=1 generates DMA Interrupt when the operation of the descriptor line is completed.

Act2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Do not execute current line and go to next line.
0	1	Rsv	Reserved	Do not execute current line and go to next line.
1	0	Tran	Transfer Data	Transfer data of one descriptor line.
1	1	Link	Link Descriptor	Link to another descriptor.

Fig. 22.9: 32-Bit Address Descriptor Table

The above figure shows the definition of the 32-bit address descriptor table. One descriptor line consumes 64 bits (8 bytes) of memory space. The attribute is used to control descriptors. Three action symbols are specified here. The "Nop" operation skips the current descriptor line and goes to the next line. The "Tran" operation transfers the data specified by the address and length fields. The "Link" operation is used to link two separate descriptors. The link address field points to the next descriptor table. The combination of Act2=0 and Act1=1 is reserved and defined as the same operation as "Nop". Future versions of the controller may use this field and redefine new operations. The 32-bit address is stored in the lower 32 bits of the 64-bit address register. For a 32-bit address descriptor table, the address field shall be set on the 32-bit boundary (the lower 2 bits are always set to 0).

22.3.12.4 ADMA2 State

This figure illustrates the four states of ADMA2: fetch descriptor state, change address state, transfer data state and stop ADMA state:

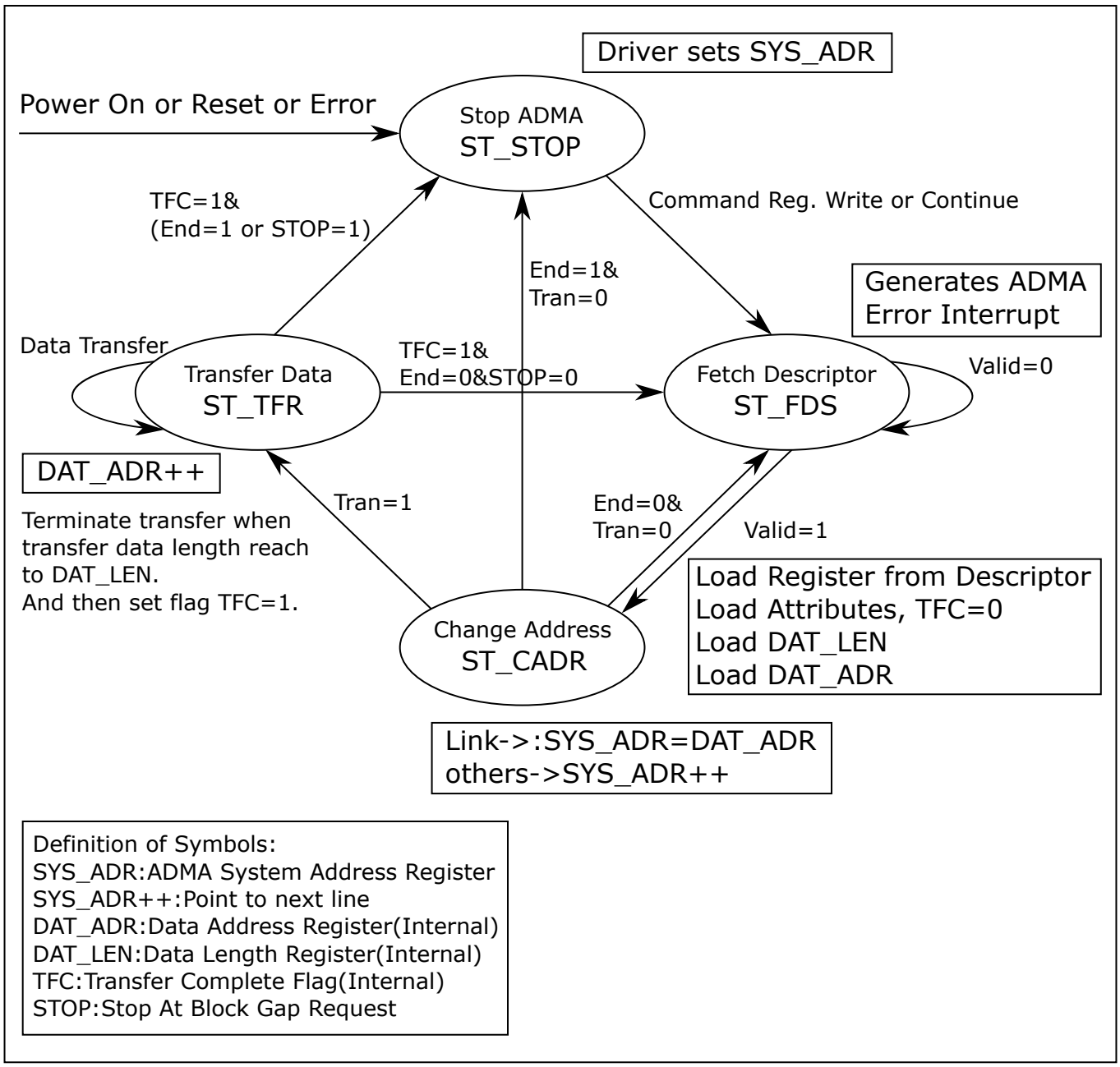


Fig. 22.10: ADMA2 States

- ST_FDS (fetch descriptor): ADMA2 fetches the descriptor line and sets the parameters in the internal register, and then switches to the ST_CADR state.
- ST_CADR (change address): The “Link” operation loads another descriptor address into the ADMA system address register. In other operations, the ADMA system address register is incremented to the next descriptor

line. If End=0, it switches to the ST_TFR state. Even if some errors occur, ADMA2 must not stop in this state.

- ST_TFR (transfer data): The data of one descriptor line is transferred between system memory and SD card. If data transfer continues (End=0), it switches to the ST_FDS state. If data transfer is completed, it switches to the ST_STOP state.
- ST_STOP (stop DMA): ADMA2 will remain in this state in these cases: after power-on reset or software reset; all the descriptor data has been transferred. It switches to the ST_FDS state if a new ADMA2 operation is started by writing to the command register.

ADMA2 does not support the Suspend/Resume function, but can use the Stop and Resume functions. When the block gap stop request in the block gap control register is set during ADMA2 operation, the block gap event interrupt will be generated when ADMA2 stops at the block gap. The host controller shall use the Read Wait or Stop SD Clock to stop the ADMA2 read operation. When ADMA2 is stopped, no SD command can be issued. Errors occurred during ADMA2 transfer may stop ADMA2 operation and generate an ADMA error interrupt. The ADMA Error Status field in the ADMA Error Status register stores the status that ADMA2 has stopped. The host driver can identify the location of the error descriptor in two ways. That is, if ADMA stops in the ST_FDS state, the ADMA system address register will point to the error descriptor line. If ADMA stops in the ST_TFR or ST_STOP state, the ADMA system address register will point to the next line of the error descriptor line. Therefore, ADMA2 must not stop in the ST_CADR state.

22.3.13 Test Register

The test register is defined for the purpose of testing. When it is difficult to intentionally generate some interrupts, you may use this function to manually generate such interrupts for driver debugging. To that end, a force event register is defined to control the Error Interrupt Status and the automatic CMD error status. It is also difficult to control card insertion and removal intentionally. The card detection signal selection and card detection test level in the host control 1 register make it possible to manually control the card inserted in the current status register and generate card insertion and card removal interrupts in the Normal Interrupt Status register.

22.3.14 Block Count

The block count command (CMD23) provides an untimed method to stop multi-block operations. The block count is set in the parameter of CMD23 to specify the transfer length of CMD18 or CMD25 after that. Automatic CMD23 is the function of automatically sending CMD23 before sending CMD18 or CMD25. This function aims to avoid performance degradation during memory access by deleting the interrupt service of CMD23. The offset 008h parameter 1 register is used for CMD18 or CMD25. Then an offset of 000h is assigned to the parameter 2 register of CMD23. The host controller does not use the parameter 2 register to calculate the data transfer length. There are two cases of data length of the host-side data transfer operation: non-ADMA and ADMA. The total length of AMDA descriptor refers to the sum of all the 16-bit data length of each ADMA descriptor line. The “block count enable” shall be disabled for ADMA. It is worth noting that the total data transfer length of the host controller shall be equal to the transfer length of the card.

22.3.15 Sampling Clock Tuning

In UHSI mode, the SD bus can run in the high clock frequency mode, and then the data window of cards on CMD and DAT[3:0] lines becomes smaller. The position of the data window varies with the implementation of the card and host system. Hence, when SDR104 or SDR50 is supported by executing the tuning program and adjusting the sampling clock (if the usage tuning of SDR50 is set to 1 in the capability register), the host controller shall support the tuning circuit. The execution tuning and sampling clock selection in the host control 2 register is used to control the tuning circuit.

22.3.16 SD Host Standard Register

22.3.16.1 SD Host Control Register Mapping

The following table summarizes the standard SD host controller register sets. The host driver needs to determine the base address of the register set by the host system specific method. The size of the register set is 256 bytes. For multiple card slot controllers, one register set is assigned to each card slot, but the register at the offset 0F0h0FFh is assigned as a common area, and these registers contain the same value from each card slot register set.

Offset	15-08 bit	07-00 bit	Offset	15-08 bit	07-00 bit
002h	SDMA System Address(High),Argument 2(High)		000h	SDMA System Address(Low),Argument 2(Low)	
006h	Block Count		004h	Block Size	
00Ah	Argument 1(High)		008h	Argument 1(Low)	
00Eh	Command		00Ch	Transfer Mode	
012h	Response1		010h	Response0	
016h	Response3		014h	Response2	
01Ah	Response5		018h	Response4	
01Eh	Response7		01Ch	Response6	
022h	Buffer Data Port1		020h	Buffer Data Port0	
026h	Present State		024h	Present State	
02Ah	Wakeup Control	Block Gap Control	028h	Power Control	Host Control 1
02Eh	Software Reset	Timeout Control	02Ch	Clock Control	
032h	Error Interrupt Status		030h	Normal Interrupt Status	
036h	Error Interrupt Status Enable		034h	Normal Interrupt Status Enable	
03Ah	Error Interrupt Signal Enable		038h	Normal Interrupt Signal Enable	
03Eh	Host Control 2		03Ch	Auto CMD Error Status	
042h	Capabilities		040h	Capabilities	
046h	Capabilities		044h	Capabilities	
04Ah	Maximum Current Capabilities		048h	Maximum Current Capabilities	
04Eh	Maximum Current Capabilities(Reserved)		04Ch	Maximum Current Capabilities(Reserved)	
052h	Force Event for Error Interrupt Status		050h	Force Event for Auto CMD Error Status	
056h	---		054h	---	ADMA Error Status
05Ah	ADMA System Address [31:16]		058h	ADMA System Address [15:00]	
05Eh	ADMA System Address [63:48]		05Ch	ADMA System Address [47:32]	
062h	Preset Value		060h	Preset Value	
066h	Preset Value		064h	Preset Value	
06Ah	Preset Value		068h	Preset Value	
06Eh	Preset Value		06Ch	Preset Value	
---	---		---	---	
0E2h	Shared Bus Control(High)		0E0h	Shared Bus Control(Low)	
---	---		---	---	
0F2h	---		0F0h	---	
---	---		---	---	
0FEh	Host Controller Version		0FCh	Slot Interrupt Status	

Fig. 22.11: SD Host Control Register Mapping

22.3.16.2 Configuration of Register Type

The configuration register field is assigned an attribute described in the following table:

Register Attribute	Description
RO	Read-only register: Register bits are read-only and cannot be altered by software or any reset operation. Writes to these bits are ignored.
ROC	Read-only status: These bits are initialized to zero at reset. Writes to these bits are ignored.
RW	Read-Write register: Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status: Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
RWAC	Read-Write, automatic clear register: The Host Driver requests a Host Controller operation by setting the bit. The Host Controllers shall clear the bit automatically when the operation of complete. Writing a 0 to RWAC bits has no effect.
HwInit	Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization, and writes to these bits are ignored.
Rsvd	Reserved. These bits are initialized to zero, and writes to them are ignored.
WO	Write-only register. It is not physically implemented register. Rather, it is an address at which registers can be written.

Fig. 22.12: Types of Registers and Register Bit Fields

22.3.16.3 Initial Value of Register

The host controller sets all registers to their initial values at power-on reset, and the default values of all other registers shall be all bits that are set to zero. The values of the capability register and the maximum current capability register depend on the host controller, and the value of the host controller version register also depends on the host controller.

22.3.16.4 Reserved Bits of a Register

“Reserved” means that this bit can be defined for future use, and it is currently set to 0. These bits shall be set to 0.

23 SDIO

23.1 Overview

Secure Digital Input and Output (SDIO) is a peripheral interface developed on the basis of the SD memory card interface. The SDIO card that is compatible with the SD memory card aims to provide high-speed data I/O and low-power mobile electronic devices.

23.2 Features

- For portable and stationary applications
- Minimal modification or no modification to the SD physical bus
- Minimal changes to memory driver software
- Extended physical form factor for special applications
- Plug and play
- Multi-functional support, including multiple I/O, combined I/O and memory
- One card supports up to seven I/O functions and one memory.
- Allow the card to interrupt the host

23.3 Functional Description

23.3.1 Definition of SDIO Signal

23.3.1.1 SDIO Card Type

There are two types of SDIO cards, full-speed and low-speed SDIO cards. In the full clock range of 0–25 MHz, the full-speed card supports SPI, 1-bit SD and 4-bit SD transfer modes. The data transfer rate of a full-speed SDIO card exceeds 100 Mb/s (10 MB/s). The low-speed SDIO card only needs SPI and 1-bit SD transfer modes and supports

the full clock range of 0–400 KHz. The low-speed card intends to provide the low-speed I/O capability with minimum hardware. It supports modem, barcode scanner, GPS receiver, and other functions.

23.3.1.2 SDIO Card Mode

Here are three signal modes defined for SD memory cards that also apply to SDIO cards. First, SPI mode, where Pin 8 is used as an interrupt pin, and all other pins and signaling protocols are the same as that specified in the SD Physical Layer Specification. Second, 1-bit SD data transfer mode, where data is only transferred on the DAT[0] pin. Pin 8 is used as an interrupt pin, and all other pins and signaling protocols are the same as that specified in the SD memory specification. Third, 4-bit SD data transfer mode, where data is transferred on all 4 data pins DAT[3:0]. In this mode, the interrupt pin must not be used exclusively, because it is used as a data transfer line. Thus, if the interrupt function is required, special timing is required to provide the interrupt. The 4-bit SD mode provides the fastest possible data transfer, up to 100 Mb/s.

23.3.1.3 Signal Pin

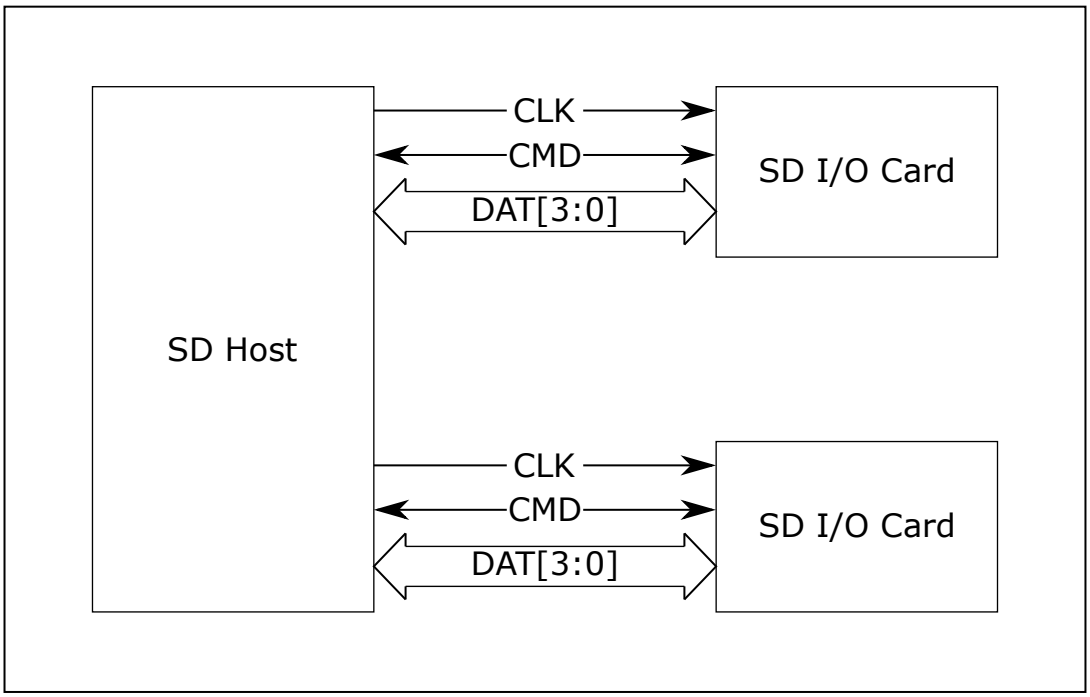


Fig. 23.1: Signal Connection of Two 4-Bit SDIO Cards

23.3.2 SDIO Card Initialization

23.3.2.1 Difference in I/O Card Initialization

The SDIO Specification specifies that when SDIO card is inserted, it shall not cause the failure of non-I/O aware host. To prevent I/O functions from being operated in non-I/O aware hosts, the flow chart of SD card recognition mode shall be changed. A new command (IO_SEND_OP_COND, CMD5) is added to replace ACMD41 and initialize SDIO through the I/O aware host. After reset or power-on, all I/O functions on the card are disabled. When CS is Low, the I/O part of the card will not perform any operation except CMD5 or CMD0. If a SD memory is installed on the card, it shall normally respond to all normal forced memory commands. I/O only cards must not respond to ACMD41, so they are initially displayed as MMC cards. I/O only cards must not respond to the CMD1 used to initialize the MMC card, so they are displayed as non-responding cards. Then the host abandons and disables the card. Therefore, the non-aware host does not receive the response from the I/O only card, and forces it into the inactive state. The following figure shows the operation of the I/O card with non-I/O aware host. The solid line is the actual path, and the dotted line part is not executed:

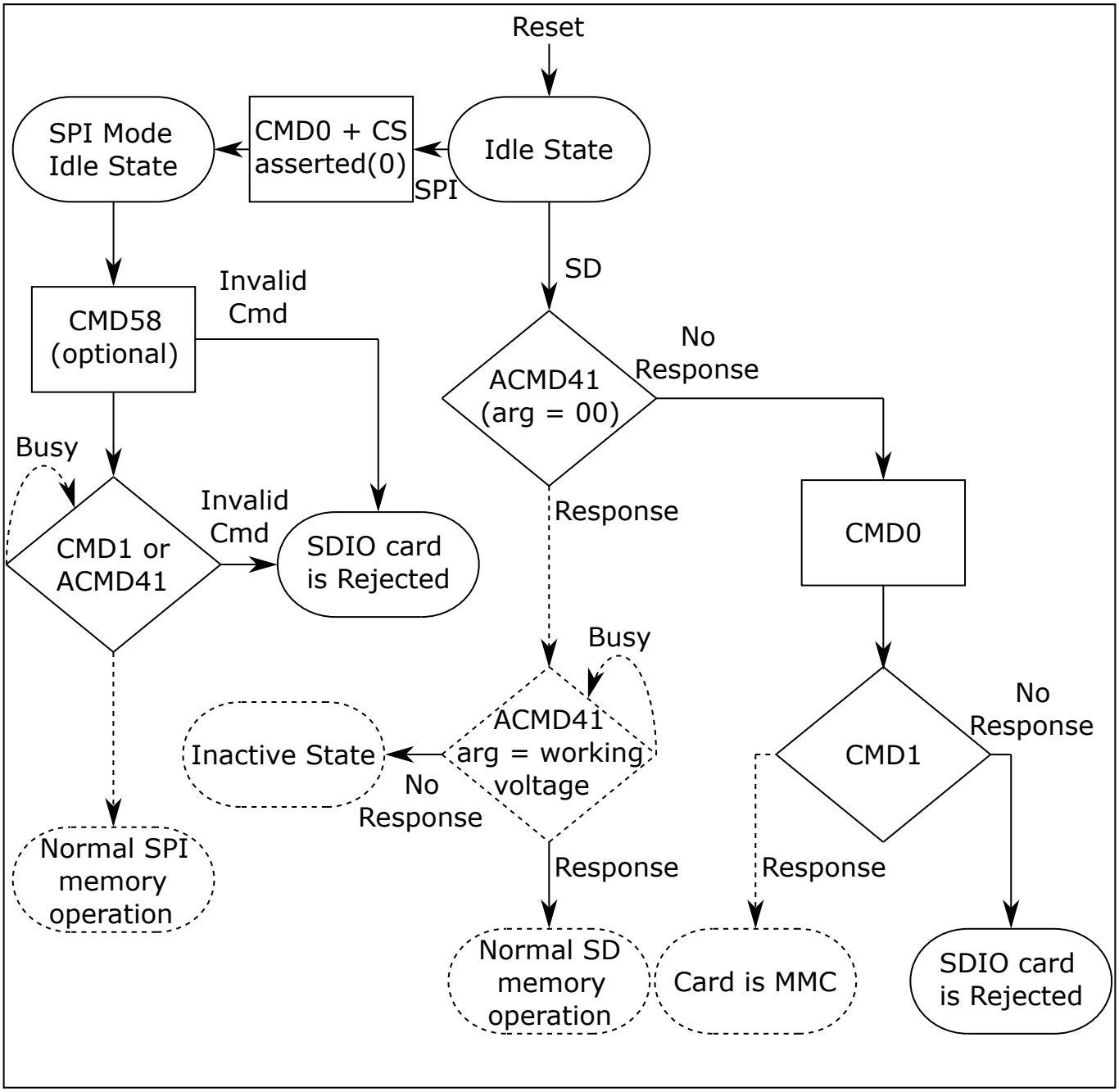


Fig. 23.2: SDIO’s Response to Non-I/O Aware Initialization

23.3.2.2 IO_SEND_OP_COND command (CMD5)

The function of the IO_SEND_OP_COND command (CMD5) for the SDIO card is similar to that of ACMD41 for the SD memory card, and it is used to query the required voltage range of the I/O card. The normal response of CMD5 is R4 in SD or SPI format. The format of CMD5 is shown as follows:

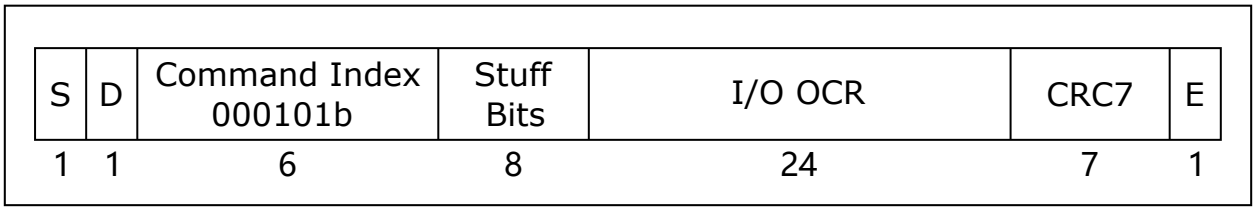


Fig. 23.3: IO_SEND_OP_COND Command

其中：

- S: Start bit, always 0.
- D: Direction, always 1, indicating transfer from host to card.
- Command Index: Identifies the CMD5 command with a value of 000101b.
- Stuff Bits: Not used, shall be set to 0.
- I/O OCR: Operation Conditions Register. The supported minimum and maximum values for VDD.
- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

23.3.2.3 IO_SEND_OP_COND Response (R4)

An SDIO card receiving CMD5 shall respond with a SDIO unique response, R4. The format of R4 for both the SD and SPI modes is:

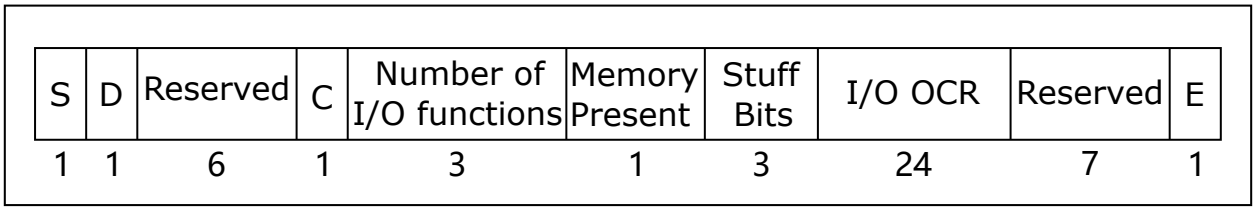


Fig. 23.4: Response R4 in SD Mode

Modified R1	C	Number of I/O functions	Memory Present	Stuff Bits	I/O OCR
8	1	3	1	3	24

Fig. 23.5: Response R4 in SPI Mode

- S: Start bit, always 0.
- D: Direction, always 0, indicating the transfer from card to host.
- Reserved: Bits reserved for future use. These bits shall be set to 1.
- C: Set to 1 if the card is ready to operate after initialization.
- Stuff Bits: Not used, shall be set to 0.
- I/O OCR: Operation Conditions Register. The supported minimum and maximum values for VDD.
- E: End bit, always 1.
- Memory Present: Set to 1 if the card also contains SD memory. Set to 0 if the card is I/O only.
- Number of I/O Functions: Indicates the total number of I/O functions supported by this card. The range is 0-7. Note that the common area present on all I/O cards at function 0 is not included in this count. The I/O functions shall be implemented sequentially beginning at function 1.
- Modified R1: The SPI R1 response byte as described in the SD Physical Layer Specification is modified for I/O as follows:

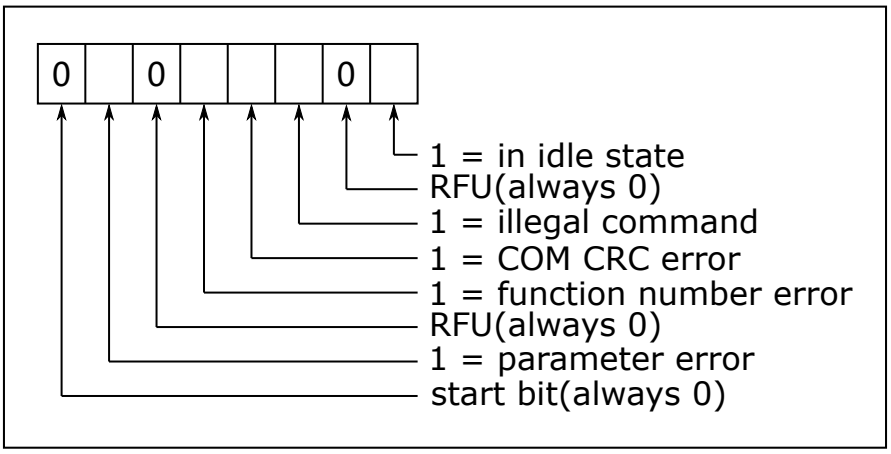


Fig. 23.6: Modified R1 Response

23.3.2.4 Special Initialization Considerations for combo Cards

The host shall be aware of some special situations when initializing a combo card (SDIO plus SD memory on the same card). This is because an implementation of the combo card could actually use two separate controllers (Memory and I/O) in the same package and share the same bus lines. It is important for the host to both detect and properly configure both parts (controllers) of a combo card in order to prevent conflicts between the SDIO and the SD memory controller. These concerns are due to the different responses to a reset (hard or soft) by the two controllers. Another issue is the value of the RCA (Relative Card Address) that exists within the Memory controller. Note that this consideration is for SD 1-bit and SD 4-bit modes only. In SPI mode, card select/de-select is accomplished using the hardware CS line rather than the RCA.

23.3.3 Differences with SD Memory Specification

23.3.3.1 SDIO Command List

The following figure shows the list of commands accepted by SD memory and SDIO cards when using the SD bus interface.

Supported Commands	Abbreviation	SDMEM System	SDIO System	Comments
CMD0	GO_IDLE_STATE	Mandatory	Mandatory	Used to change from SD to SPI mode
CMD2	ALL_SEND_CID	Mandatory		CID not supported by SDIO
CMD3	SEND_RELATIVE_ADDR	Mandatory	Mandatory	
CMD4	SET_DSR	Optional		DSR not supported by SDIO
CMD5	IO_SEND_OP_COND		Mandatory	
CMD6	SWITCH_FUNC	Mandatory	Mandatory	
CMD7	SELECT/DESELECT_CARD	Mandatory	Mandatory	
CMD9	SEND_CSD	Mandatory		CSD not supported by SDIO
CMD10	SEND_CID	Mandatory		CID not supported by SDIO
CMD12	STOP_TRANSMISSION	Mandatory		
CMD13	SEND_STATUS	Mandatory		Card Status includes only SDMEM information
CMD15	GO_INACTIVE_STATE	Mandatory	Mandatory	
CMD16	SET_BLOCKLEN	Mandatory		
CMD17	READ_SINGLE_BLOCK	Mandatory		
CMD18	READ_MULTIPLE_BLOCK	Mandatory		
CMD24	WRITE_BLOCK	Mandatory		
CMD25	WRITE_MULTIPLE_BLOCK	Mandatory		
CMD27	PROGRAM_CSD	Mandatory		CSD not supported by SDIO
CMD28	SET_WRITE_PROT	Optional		
CMD29	CLR_WRITE_PROT	Optional		
CMD30	SEND_WRITE_PROT	Optional		
CMD32	ERASE_WR_BLK_START	Mandatory		
CMD33	ERASE_WR_BLK_END	Mandatory		
CMD38	ERASE	Mandatory		
CMD42	LOCK_UNLOCK	Optional		
CMD52	IO_RW_DIRECT		Mandatory	
CMD53	IO_RW_EXTENDED		Mandatory	Block mode is optional
CMD55	APP_CMD	Mandatory		
CMD56	GEN_CMD	Mandatory		
ACMD6	SET_BUS_WIDTH	Mandatory		
ACMD13	SD_STATUS	Mandatory		
ACMD22	SEND_NUM_WR_BLOCKS	Mandatory		
ACMD23	SET_WR_BLK_ERASE_COUNT	Mandatory		
ACMD41	SD_APP_OP_COND	Mandatory		
ACMD42	SET_CLR_CARD_DETECT	Mandatory		
ACMD51	SEND_SCR	Mandatory		SCR not supported by SDIO

Fig. 23.7: Command List for SD Mode

The following figure shows the list of commands accepted by SD memory and SDIO cards when using the SPI bus interface.

Supported Commands	Abbreviation	SDMEM System	SDIO System	Comments
CMD0	GO_IDLE_STATE	Mandatory	Mandatory	Used to change from SD to SPI mode
CMD1	SEND_OP_COND	Mandatory		
CMD5	IO_SEND_OP_COND		Mandatory	
CMD6	SWITCH_FUNC	Mandatory	Mandatory	
CMD9	SEND_CSD	Mandatory		CSD not supported by SDIO
CMD10	SEND_CID	Mandatory		CID not supported by SDIO
CMD12	STOP_TRANSMISSION	Mandatory		
CMD13	SEND_STATUS	Mandatory		Card Status includes only SDMEM information
CMD16	SET_BLOCKLEN	Mandatory		
CMD17	READ_SINGLE_BLOCK	Mandatory		
CMD18	READ_MULTIPLE_BLOCK	Mandatory		
CMD24	WRITE_BLOCK	Mandatory		
CMD25	WRITE_MULTIPLE_BLOCK	Mandatory		
CMD27	PROGRAM_CSD	Mandatory		CSD not supported by SDIO
CMD28	SET_WRITE_PROT	Optional		
CMD29	CLR_WRITE_PROT	Optional		
CMD30	SEND_WRITE_PROT	Optional		
CMD32	ERASE_WR_BLK_START	Mandatory		
CMD33	ERASE_WR_BLK_END	Mandatory		
CMD38	ERASE	Mandatory		
CMD42	LOCK_UNLOCK	Optional		
CMD52	IO_RW_DIRECT		Mandatory	
CMD53	IO_RW_EXTENDED		Mandatory	Block mode is optional
CMD55	APP_CMD	Mandatory		
CMD56	GEN_CMD	Mandatory		
CMD58	READ_OCR	Mandatory		
CMD59	CRC_ON_OFF	Mandatory	Mandatory	
ACMD13	SD_STATUS	Mandatory		
ACMD22	SEND_NUM_WR_BLOCKS	Mandatory		
ACMD23	SET_WR_BLK_ERASE_COUNT	Mandatory		
ACMD41	SD_APP_OP_COND	Mandatory		
ACMD42	SET_CLR_CARD_DETECT	Mandatory		
ACMD51	SEND_SCR	Mandatory		SCR includes only SDMEM information

Fig. 23.8: Command List for SPI Mode

23.3.3.2 Unsupported SD Memory Commands

Several commands required for SD memory cards are not supported by either SDIO-only cards or the I/O portion of combo cards. Some of these commands have no use in SDIO cards such as Erase commands and thus are not supported in SDIO. Moreover, there are several commands for SD memory cards that have different commands when used with the SDIO section of a card. The following table lists these SD memory commands and the equivalent SDIO commands.

SD Memory Command	SDIO Command	Comments
CMD0	CMD52 (write to I/O reset in CCCR)	The reset command (CM DO) is only used for memory or the memory portion of Combo cards. In order to reset an I/O only card or the I/O portion of a combo card, use CMD52 to write a 1 to the RES bit in the CCCR (bit 3 of register 6). Note that in the SD mode, CMDO is only used to indicate entry into SPI mode and shall be supported. An I/O only card or the I/O portion of a combo card is not reset with CMDO
CMD12	CMD52 (write to I/O abort)	In order to CMD12. In write to the See 4.8 for abort the block transfer of data, SD memory use order to abort an I/O transaction, use CMD52 to abort register in the CCCR (bits 2:0 of register 6)
CMD16	CMD52 (write to I/O Block Length)	CMD16 sets the block length for SD memory. In order to set the block length for each I/O function, use CMD52 to write the block length in the FBR
CMD2	NONE	The CID register does not exist in an SDIO only card
CMD4	NONE	The DSR register does not exist in an SDIO only card
CMD9	NONE	The CSD register does not exist in an SDIO only card
CMD10	NONE	The CID register does not exist in an SDIO only card
CMD13	NONE	An SDIO only card or the I/O portion of a combo card does not support the same SEND_STATUS (CMD13) protocol the SD memory uses.
ACMD6	CMD52 (write to Bus_Width [1:0] in CCCR)	SET_BUS_VVIDTH is handled by a write to the CCCR.
ACMD13	NONE	The SD Status register does not exist in an SDIO only card
ACMD41	CMD5	SDIO cards and hosts use the IO SEND OP COND Command (CMD5).
ACMD42	CMD52	In the SD mode, the pull-up resistor on DAT[3] is controlled by writing to the CD Disable bit in the CCCR. For Combo Cards, this resistor is enabled unless both the memory and the I/O control registers are set to disable the resistor.
ACMD51	NONE	The SCR register does not exist in an SDIO only card
CMD17 CMD18 CMD24 CMD25	CMD53	I/O block operations use CMD53, rather than memory block read/write commands

Fig. 23.9: Unsupported SD Memory Commands

23.3.3.3 Modified R6 Response

The normal response to CMD3 by a memory card is R6, as shown in the following table:

Bit position	47	46	[45:40]	[39:8]Argument field		[7:1]	0
Width(bits)	1	1	6	16	16	7	1
Value	'0'	'0'	X	X	X	X	'1'
Description	Start bit	Direction bit	Command index ('000011')	New published RCA [31:16] of the card	[15:0] Card status	CRC7	end bit

Fig. 23.10: R6 Response of CMD3

The card status bits (8–23) are changed when CMD3 is sent to an I/O only card. In this case, the 16 bits of response shall be the SDIO-only values shown in the following table.

Bits	Identifier	Type	Value	Description	Clear Condition
15	COM_CRC_ERROR	E R	'0' = no error '1' = error	CRC check failed on previous command	B
14	ILLEGAL_COMMAND	E R	'0' = no error '1' = error	Command is illegal for card status	B
13	ERROR	E R X	'0' = no error '1' = error	A general or unknown error occurs during operation	C
12:0	Not defined. Only the SDIO card should read 0. The host should ignore these bits				

Fig. 23.11: SDIO R6 Status Bit

23.3.3.4 Reset for SDIO

In order to reset all functions within an SDIO card or the SDIO portion of a combo card, a method different than that used for SD memory is defined. The reset command (CMD0) is only used for memory or the memory portion of combo cards. In order to reset an I/O only card or the I/O portion of a combo card, use CMD52 to write a 1 to the RES bit in the CCCR (bit 3 of register 6). Note that in the SD mode, CMD0 is only used to indicate entry into SPI mode. An I/O only card or the I/O portion of a combo card is not reset by CMD0.

23.3.3.5 Bus Width

For an SD memory card, the bus width for SD mode is set using ACMD6. For an SDIO card, a write to the CCCR using CMD52 is used to select bus width. In the case of a combo card, both selection methods exist. In this case, the host shall set the bus width in both locations by issuing both the ACMD6 and the CCCR write using CMD52 with the same width before starting any data transfer.

23.3.3.6 Card Detect Resistor

SD memory and I/O cards use a pull-up resistor on DAT[3] to detect card insertion. The procedure to enable/disable this resistor is different between SD memory and SDIO. SD memory uses ACMD42 to control this resistor while SDIO uses writes to the CCCR using CMD52. In the case of a combo card, both control modes exist and shall be managed by the host. For a combo card, the resistor is enabled only when both the memory and the I/O control registers have the resistor enabled. That is, after a power on, the host shall disable the resistor by sending ACMD42 to the memory controller or a CCCR write to the SDIO controller since the resistor enable is a logical AND of the two enables. After power-up, both locations default to resistor enabled. It is worth noting that after an I/O reset, the I/O resistor enable is not changed.

23.3.3.7 Data Transfer Block Sizes

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format, while the SD memory cards are fixed in the block transfer mode. The SD Physical Layer Specification limits the block size for data transfer to powers of 2 (i.e. 512, 1024, 2048) unless using partial read and write. The SDIO Specification allows any block size from 1 byte to 2048 bytes in order to accommodate the various natural block sizes for I/O functions. It is worth noting that an SDIO card function may define a maximum block size or byte count in the CIS that is smaller than the maximum values described above.

23.3.3.8 Data Transfer Abort

A host communicating with a SD memory card uses CMD12 to abort the transfer of read or write data from/to the card. For an SDIO card, CMD12 abort is replaced by a write to the ASx bits in the CCCR. Normally, the abort is used to stop an infinite block transfer (block count=0). If an exact number of blocks are to be transferred, it is recommended that the host issue a block command with the correct block count, rather than using an infinite count and aborting the data at the correct time.

23.3.4 New I/O Read/Write Commands

Two additional data transfer instructions have been added to support I/O. IO_RW_DIRECT is a direct I/O command similar to MMC's "fast I/O" command. IO_RW_EXTENDED allows fast access with byte or block addresses. Both commands are in class 9 (I/O Commands).

23.3.4.1 IO_RW_DIRECT Command (CMD52)

The IO_RW_DIRECT is the simplest means to access a single register within the 128K of register space in any I/O function, including the common I/O area (CIA). This command reads or writes 1 byte using only 1 command/response pair. A common use is to initialize registers or monitor status values for I/O functions. This command is the fastest means to read or write single I/O registers, as it requires only a single command/response pair. The command structure is shown as follows:

S	D	Command Index 110100b	R/W flag	Function Number	RAW flag	Stuff	Register Address	Stuff	Write Data or Stuff Bits	CRC7	E
1	1	6	1	3	1	1	17	1	8	7	1

Fig. 23.12: IO_RW_DIRECT Command

- S: Start bit, always 0.
- D: Direction, always 1, indicating transfer from host to card.
- Command Index: Identifies the "IO_RW_DIRECT" command with a value of 110100b.
- R/W Flag: This bit determines the direction of the I/O operation. If this bit is 0, this command shall read data from the SDIO card at the address specified by the Function Number and the Register Address to the host. The data byte is returned in the response, R5. If this bit is set to 1, the command shall write the bytes in the Write Data field to the I/O location addressed by the Function Number and the Register Address. If the RAW flag is 0, then the data in the register that was written shall be read and that value returned in the response.
- RAW Flag: The Read after Write flag. If this bit is set to 1 and the R/W flag is set to 1, then the command shall read the value of the register after the write. This is useful for allowing writing to the control register and reading

the state of the same address. If this bit is cleared, the value returned in the R5 response shall be the same as the write data in the command. If this bit is set, the data field of the R5 response shall contain the value read from the addressed register after the write operation.

- Function Number: The number of the function within the I/O card you wish to read or write. Note that function 0 selects the common I/O area (CIA).
- Register Address: This is the address of the byte of data inside of the selected function to read or write. There are 17 bits of address available so the register is located within the first 128K (131,072) addresses of that function.
- Write Data/Stuff Bits: For a direct write command (R/W=1), this is the byte that is written to the selected address. For a direct read (R/W=0), this field is not used and shall be set to 0.
- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

23.3.4.2 IO_RW_DIRECT Response (R5)

The SDIO card's response to CMD52 shall be in one of two formats. If the communication between card and host is in the 1-bit or 4-bit SD mode, the response shall be in a 48-bit response (R5). If the operation was a read command, the data being read is returned as an 8-bit value. In addition, 15 bits of status information is returned. The format of the response is as follows:

S	D	Command Index 110100b	Stuff	Response Flags Bit 7-------0	Read or Write Data	CRC7	E
1	1	6	16	8	8	7	1

Fig. 23.13: IO_RW_DIRECT Response in SD Mode

- S: Start bit, always 0.
- D: Direction, always 0, indicating the transfer from card to host.
- Command Index: Identifies the "IO_RW_DIRECT" command with a value of 110100b.
- Stuff Bits: Not used, shall be set to 0.
- Response Flags: 8 bits of flag data indicating the status of the SDIO card.
- Read or Write Data: For an I/O write (R/W=1) with the RAW Flag set (RAW=1), this field shall contain the value read from the addressed register after the write of the data contained in the command. Note that in this case, the read-back data may not be the same as the data written to the register, depending on the design of the hardware. For an I/O write with the RAW bit=0, the SDIO function shall not do a read after write operation, and the data in this field shall be identical to the data byte in the write command. For an I/O read (R/W=0), the actual value read

from that I/O location is returned in this field.

- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

If the communication is using the SPI mode, the response shall be a 16-bit R5 response. If the operation was a read command, the data being read is returned as an 8-bit value. In addition, 8 bits of status information is returned in a SPI R1 response byte. The format of the response is as follows:

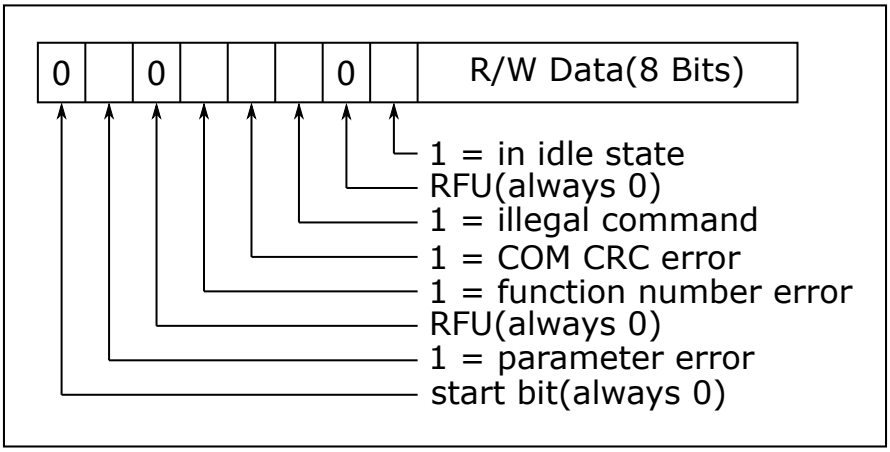


Fig. 23.14: IO_RW_DIRECT Response in SPI Mode

Note: The read/write (R/W) data is identical to the read/write data described for the SD R5 response. Parameter error status in SPI mode corresponds to OUT_OF_RANGE and ERROR in the SD mode response. In the case of CMD53, Data Error Token shall also be used to indicate OUT_OF_RANGE and ERROR.

23.3.4.3 IO_RW_EXTENDED Command (CMD53)

In order to read and write multiple I/O registers with a single command, a new command, IO_RW_EXTENDED is defined. This command is included in command class 9 (I/O Commands). This command allows the reading or writing of a large number of I/O registers with a single command. Since this is a data transfer command, it provides the highest possible transfer rate. The IO_RW_EXTENDED command is shown as follows:

S	D	Command Index 110101b	R/W flag	Function Number	Block Mode	OP Code	Register Address	Byte/Block Count	CRC7	E
1	1	6	1	3	1	1	17	9	7	1

Fig. 23.15: IO_RW_EXTENDED Command

- S: Start bit, always 0.
- D: Direction, always 1, indicating transfer from host to card.

- Command Index: Identifies the “IO_RW_EXTENDED” command with a value of 110101b.
- R/W Flag: This bit determines the direction of the I/O operation. If this bit is 0, this command reads data from the SDIO card at the address specified by the Function Number and the Register Address to the host. The read data shall be returned on the DAT[x] lines. If this bit is set to 1, the command shall write the bytes from the DAT[x] lines to the I/O location addressed by the Function Number and the Register Address.
- Function Number: The number of the function within the I/O card you wish to read or write. Note that function 0 selects the common I/O area (CIA).
- Block Mode: (Optional) this bit, if set to 1, indicates that the read or write operation shall be performed on a block basis, rather than the normal byte basis. If this bit is set, the byte/block count value shall contain the number of blocks to be read/written. The block size for functions 1-7 is set by writing the block size to the I/O block size register in the FBR. The block size for function 0 is set by writing to the FN0 block size register in the CCCR. Card and host support of the block I/O mode is optional. The host can determine if a card supports block I/O by reading the Card supports MBIO bit (SMB) in the CCCR. The block size used when Block Mode = 1 and the maximum byte count per command used when Block Mode = 0 can be read from the CIS in the tuple TPLFE_MAX_BLK_SIZE on a per-function basis.
- OP code: Defines the read/write operation. 0 is used to read or write multiple bytes of data to/from a fixed address. 1 is used to read or write multiple bytes of data to/from an incremental address.
- Register Address: Start Address of I/O register to read or write. Range is [0~0x1FFF].
- Byte/Block Count: If the command is operating on bytes (Block Mode = 0), this field contains the number of bytes to read or write. A value of 0X000 shall cause 512 bytes to be read or written.
- CRC7: 7 bits of CRC data.
- E: End bit, always 1.

23.3.5 SDIO Card Internal Operation

I/O access differs from memory in that the registers can be written and read individually and directly without a FAT file structure or the concept of blocks (although block access is supported). These registers allow access to the I/O data, control of the I/O function, report on status or transfer I/O data to the host. SD memory relies on the concept of a fixed block length with commands reading/writing multiples of these fixed size blocks. I/O may or may not have fixed block lengths and the read size may be different from the write size. Because of this, I/O operations may be based on either a length (byte count) or a block size.

23.3.5.1 Overview

Each SDIO card may have from 1 to 7 functions plus one built-in memory function. A function is a self-contained I/O device. I/O functions may be identical or completely different from each other. All I/O functions are organized as a collection of registers. There is a maximum of 131,072 (2^17) registers possible for each I/O function. These registers and their individual bits may be Read Only (RO), Write Only (WO) or Read/Write (R/W). These registers can be 8, 16 or 32 bits wide within the card. All addressing is based on byte access. These registers can be written and/or read one at a time, multiply to the same address or multiply to an incremental address. The single R/W access is often used to initialize the I/O function or to read a single status or data value. The multiple reads to a fixed address are used to read or write data from a data FIFO register in the card. The read to incremental addresses is used to read or write a collection of data to/from a RAM area inside of the card. The following figure shows the mapping of the CIA and optional CSA space for an SDIO card.

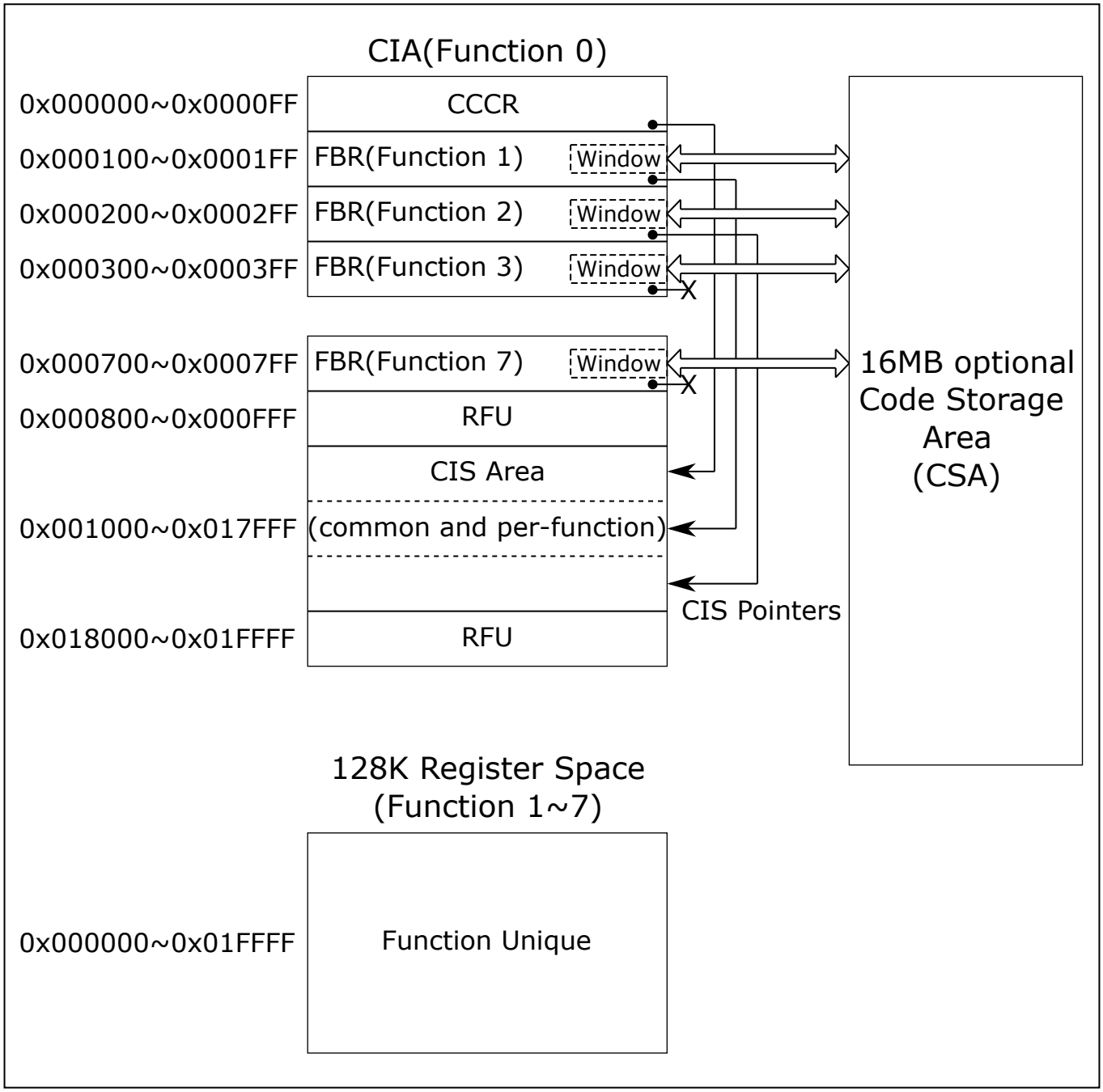


Fig. 23.16: SDIO Internal Mapping

23.3.5.2 Register Access Time

All registers in SDIO only cards and the SDIO portion of combo cards shall complete read and write data transfers in less than one second. This timeout value relates to the time for the requested data to be transferred to/from the host on the DAT[X] lines and not the timing between the command and the response. This wait time is signaled to the host by the card using “busy” for a write or delaying the start bit for a read operation. The host can use one second as the timeout value for a non-responding location.

23.3.5.3 Interrupt

All SDIO hosts shall support hardware interrupts. If a host does not support interrupts, it may have difficulties working with SDIO cards that expect fast response to interrupt conditions. Each function within an SDIO or combo card may implement interrupts as needed. The interrupt used on SDIO functions is a type commonly called “level sensitive”. Level sensitive means that any function may signal for an interrupt at any time, but once the function has signaled an interrupt, it shall not release (stop signaling) the interrupt until the cause of the interrupt is removed or commanded to do so by the host. Since there is only one interrupt line, it may be shared by multiple interrupt sources. The function shall continue to signal the interrupt until the host responds and clears the interrupt. Since multiple interrupts may be active at once, it is the responsibility of the host to determine the interrupt source(s) and deal with it as needed. This is done on the SDIO function by the use of two bits, the interrupt enable and interrupt suspend. Each function that may generate an interrupt has an interrupt enable bit. In addition, the SDIO card has a master interrupt enable that controls all functions. An interrupt shall only be signaled to the SD bus if both the function’s enable and the card’s master enable are set. The second interrupt bit is called interrupt suspend. This read-only bit tells the host which function(s) may be signaling for an interrupt. There is an interrupt suspend bit for each function that can generate interrupts. These bits are located in the CCCR area.

23.3.5.4 Suspend/Resume

Within a multi-function SDIO or a combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of Suspend/Resume. If a card supports Suspend/Resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function or memory. Once this higher-priority transfer is completed, the original transfer is re-started where it left off (resume). Support of Suspend/Resume is optional on a per-card basis If Suspend/Resume is implemented, it shall be supported by the memory of a combo card and all I/O functions except 0 (the CIA). It is worth noting that the host can suspend multiple transactions and resume them in any order desired. The I/O function 0 does not support Suspend/Resume. Any card that supports Suspend/Resume shall also support Read Wait and Direct Commands (SRW and SDC = 1) It is worth noting that Suspend/Resume is defined only for the SD 1 and 4-bit modes. It does not apply to SPI transfers.

23.3.5.5 Read Wait

Host devices built based on the SD Physical Layer Specification shall control the SDCLK to stop the read data block output from a card executing a multiple read command whenever the host cannot accept more data. During the time that the host has stopped the SDCLK, a CMD52 cannot be issued. This limitation causes a problem in that a host device built based on the SD Physical Layer Specification cannot perform the I/O command during a multiple read cycle. In order to eliminate this limitation, the SDIO Specification adds the Read Wait control to enable the host to issue CMD52 during a multiple read cycle. Read Wait uses the DAT[2] line to allow the host to signal the card to temporarily halt the sending of read data by a card. This feature is optional for an SDIO or combo card. However, if

an SDIO or combo supports Read Wait, all functions and any memory shall support Read Wait. Any card that supports Suspend/Resume shall also support Read Wait. It is worth noting that Read Wait is defined only for the SD 1 and 4-bit modes. It does not apply to SPI transfers.

23.3.5.6 CMD52 During Data Transfer

A card may accept CMD52 during data transfer if it supports Direct Commands. For both SD and SPI modes, if an error occurs during data transfer the SDIO card shall accept CMD52 to allow I/O abort and reset regardless of this bit value of the value of SDC.

23.3.5.7 Fixed Internal Mapping

The SDIO card has a fixed internal register space and a function unique area. The fixed area contains information about the card and certain mandatory and optional registers in fixed locations. The fixed locations allow any host to obtain information about the card and perform simple operations such as Enable in a common manner. The function unique area is a per-function area, which is defined either by the Application Specifications for Standard SDIO functions or by the vendor for non-standard functions.

23.3.5.8 Common I/O Area (CIA)

The CIA shall be implemented on all SDIO cards. The CIA is accessed by the host via I/O reads and writes to function 0. The registers within the CIA are provided to enable/disable the operation of the I/O function(s), control the generation of interrupts and optionally load software to support the I/O functions. The registers in the CIA also provide information about the function(s) abilities and requirements. There are three distinct register structures supported within the CIA. They are:

- Card Common Control Registers (CCCR)
- Function Basic Registers (FBR)
- Card Information Structure (CIS)

23.3.5.9 Card Common Control Registers (CCCR)

CCCR allows for quick host checking and control of an I/O card’s enable and interrupts on a per card (master) and per function basis. The bits in the CCCR are mixed Read/Write and read only. If any of the possible 7 functions are not provided on an SDIO card, the bits corresponding to unused functions shall all be read-only and read as 0. All reserved for future use bits (RFU) shall be read-only and return a value of 0. All writeable bits are set to 0 after power-up or reset. Access to the CCCR is possible even after initialization when the I/O functions are disabled. This allows the host to enable functions after initialization.

23.3.5.10 Function Basic Registers (FBR)

In addition to the CCCR, each supported I/O function has a 256-byte area used to allow the host to quickly determine the abilities and requirements of each function, enable power selection for each function and to enable software loading. The address of this area is from 0x00n00 to 0x00nFF where n is the function number (0x1 to 0x7).

23.3.5.11 Card Information Structure (CIS)

The Card Information Structure provides more complete information about the card and the individual functions. The CIS is the common area to read information about all I/O functions that exist in a card. The design is based on the PC Card16 design standardized by PCMCIA. All cards that support I/O shall have a common CIS and a CIS for each function. The CIS is accessed by reading the 0x1000 to 0x17FFF area. This area serves the card as a Common CIS and also as the storage area for each function. The common area and each function have a pointer to the start of its CIS within this memory space.

23.3.5.12 Multiple Function SDIO Cards

Multiple Function SDIO Cards shall have a separate set of configuration registers for each function on the card. Multiple Function SDIO Cards shall use a combination of a CIS common to all functions on the card and a separate function-specific CIS specific to each function on the card. The common CIS describes features that are common to all functions on the card. Each function-specific CIS describes features specific to a particular function on the SDIO Card. Functions are numbered sequentially beginning with 1. The CMD5 response indicates the total number of functions, which includes ‘dummy’ functions. The host shall iterate through the CIS entries based on the CMD5 response. The ERROR status flag of an R5 response is type “E R X”, and can indicate an error in the previous command. Since the host software needs a method to determine which function detected the error, a Multiple Function SDIO Card shall only return the R5 ERROR status flag in the subsequent command issued to the same function.

23.3.5.13 Setting Block Size with CMD53

The host sets the block size for a function’s multiple block transfers by writing to the 16-bit function I/O block size register in the FBR. The host shall not write this register using CMD53 with Block Mode set to 1. If the card detects an invalid block size before executing CMD53 with Block Mode set to 1, it shall indicate an OUT_OF_RANGE error in the current response and shall not perform data transfer. This will also stop the interrupt cycle

23.3.5.14 Bus State Diagram

The following figure shows the Bus State Diagram for an SDIO card. It shows the bus states and their relations to SDIO commands and Suspend/Resume.

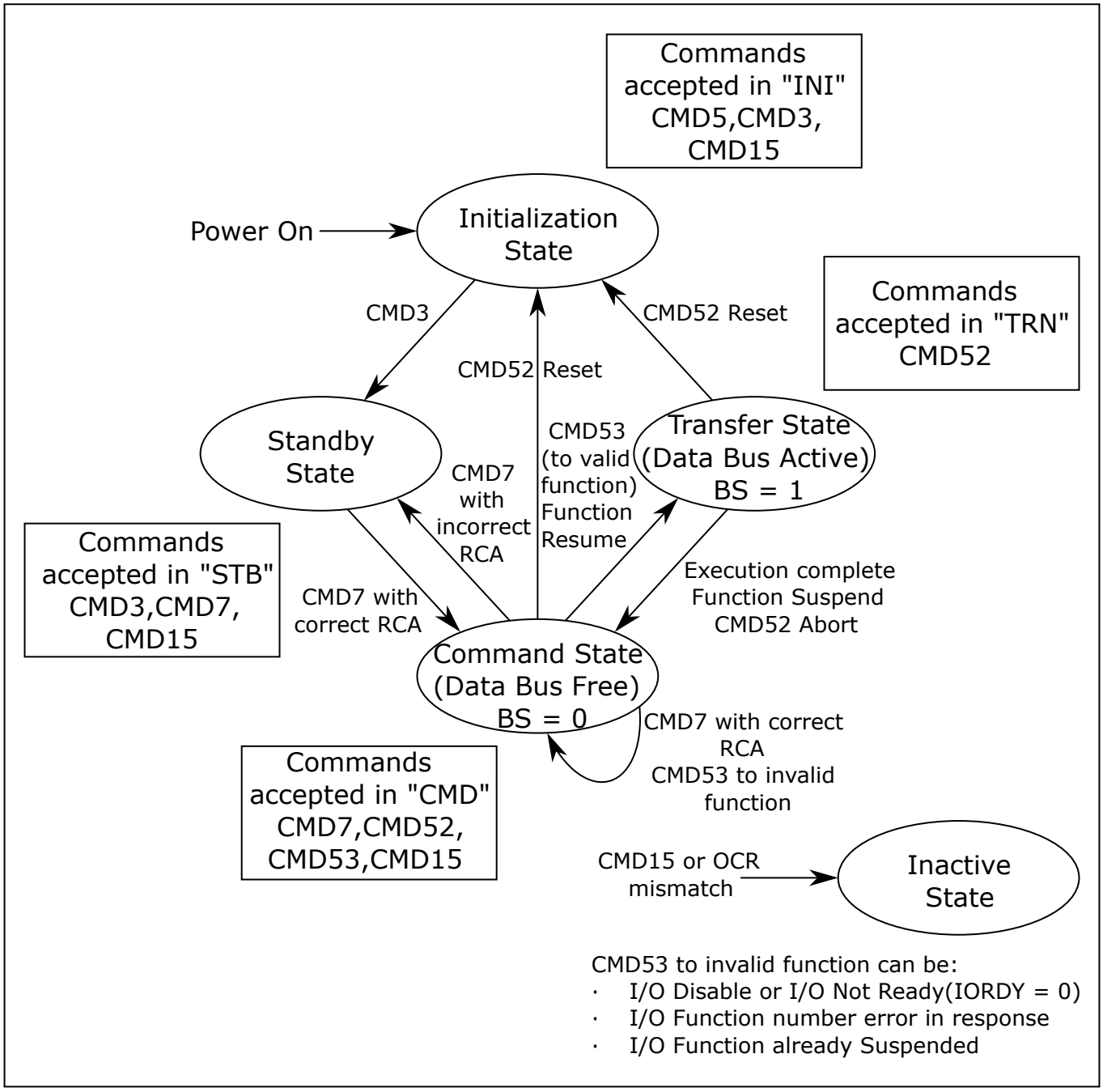


Fig. 23.17: State Diagram for Bus State Machine

23.3.6 Embedded I/O Code Storage Area (CSA)

In order to support the concept of “Plug-and-Play” for SDIO cards, each function contained in a card may need to contain a block of memory for the storage of drivers and/or applications. In addition, since the same SDIO card may be used on multiple different host platforms, several different versions of the code may be needed for each function. One option is to store these programs in a standard SD memory section of a combo card. Alternately, a standard access means to load the code is contained in the optional Code Storage Area (CSA). The CSA is a separate 16 MB memory area that is accessed using the CSA address pointer and the CSA window register contained in the FBR registers.

23.3.6.1 CSA Access

In order for the host to access a function’s CSA, it first shall determine if that function supports a CSA. The host reads the FBR register at address 0x00n00 where n is the function number (0x1 to 0x7). If bit 6=1, then the function supports a CSA and the host enables access by writing bit 7=1. The next step is for the host to load the 24-bit address to start reading or writing. This is accomplished by writing the 24 bits (A23-0) to registers 0x00n0C to 0x00n0E where n is the function number (0x1 to 0x7). Once the start address is written, data can be read or written by accessing the register 0x00n0F, the CSA data window register. If more than 1 byte needs to be read or written, an extended I/O command (byte or block) can be performed with an OP code of 0 (fixed address). The address pointer shall be automatically incremented with each access to the window register, so the access will be to sequential addresses within the CSA. Once the operation is completed, the address of the next operation shall be held in the 24-bit address register for the host to read.

23.3.6.2 CSA Data Format

The data stored in the CSA shall be structured using the FAT12/FAT16 format. The use of the CSA for program or data storage for different host types requires that the SDIO card manufacturers load the programs and data in a file format that may be recognized by the host. An example of this would be the use of a specific file name saved within a specific subdirectory that is recognized and executed by a particular host operating system. Such formats are specific and sometimes proprietary to different host implementations and operating systems.

23.3.7 SDIO Interrupts

In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the SD interface. Pin number 8, which is used as DAT[1] when operating in the 4-bit SD mode, is used to signal the card’s interrupt to the host. The use of interrupt is optional for each card or function within a card. The SDIO interrupt is “level sensitive”. That is, the interrupt line shall be held active (Low) until it is either recognized and acted upon by the host or de-asserted due to the end of the interrupt cycle. Once the host has serviced the interrupt, it is cleared via some function unique I/O operation. All hosts shall provide pull-up resistors on all data lines DAT[3:0].

23.3.7.1 SPI and SD 1-Bit Mode Interrupts

In the SPI and 1-bit SD mode, Pin 8 is dedicated to the interrupt function. Thus, in the SPI and SD 1-bit modes, there are no timing constraints on interrupts. A card in the SPI or 1-bit SD mode signals an interrupt to the host at any time by asserting pin 8 low. The host detects this pending interrupt using a level sensitive input. The host is responsible for clearing the interrupt. If the SDIO card is operating in the SPI mode, the interrupt from the card may not be asserted if the card is not selected. (CS=0). The exception to this requirement occurs only if the card is both capable of interrupting when not selected (the SCSI bit in the CCCR = 1), and has that feature turned on (the ECSI bit = 1). In this case, the card may assert the interrupt irrespective of the state of the CS line.

23.3.7.2 SD 4-Bit Mode Interrupt

Since Pin 8 is shared between the IRQ and DAT[1] when used in 4-bit SD mode, an interrupt shall only be sent by the card and recognized by the host during a specific time. The time that a low level on Pin 8 shall be recognized as an interrupt is defined as the interrupt cycle. An SDIO host shall only sample the level on Pin 8 (DAT[1]/IRQ) into the interrupt detector during the Interrupt Period. At all other times, the host interrupt controller shall ignore the level on Pin 8. Note that the interrupt cycle is applicable for both memory and I/O operations. The definition of the interrupt cycle is different for operations with single block and multiple block data transfer.

23.3.7.3 Interrupt Clear Timing

Since the SDIO card uses level sensitive interrupts, the host shall clear pending interrupts with an I/O read or write to some function unique area. In some host implementations, the sending of a CMD52 to the card is handled by host adapter hardware while the host CPU can execute other operations. This condition may allow an interrupt that has already been handled to re-interrupt the host if the timing of the interrupt clear is not controlled. To prevent this condition, any SDIO card that implements interrupts shall follow some required timing with respect to removing the interrupt from the DAT[1] line after the write to the function unique area that clears the interrupt. The clearing of the interrupt can be caused by an I/O write in a function unique method, or by a function unique I/O read. An example of clearing an interrupt using an I/O read would be a function where the reading of a data register may automatically clear the data ready interrupt.

23.3.8 SDIO Suspend/Resume Operation

The procedure used to perform the Suspend/Resume operation on the SD bus is:

- The host determines which function is currently using the DAT[3:0] line(s).
- The host requests the lower priority or slower transaction to suspend.
- The host checks for the transaction suspension to complete.
- The host begins the higher priority transaction.

- The host waits for the completion of the higher priority transaction.
- The host restores the suspended transaction.

If the current transaction can accept suspend and the card receives a Suspend command during Read Wait, it shall accept the Suspend request.

23.3.9 SDIO Read Wait Operation

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The read Wait operation allows a host to signal a card that is executing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SDIO card. To determine if a card supports the Read Wait protocol, the host shall test the SRW capability bit in the card capability byte of the CCCR. The timing for Read Wait is based on the interrupt cycle. If a card does not support the Read Wait protocol, the only means a host has to stall (not abort) data in the middle of a read multiple command is to control the SDCLK. Read Wait support is mandatory for the card to support Suspend/Resume.

24 LowPower

24.1 Overview

Low power consumption is an important indicator of IoT applications. The chip’s CPU supports the working mode, idle power saving mode, and sleep mode. You can select a proper mode according to the current application scenario to reduce the chip’s power consumption and prolong the battery life.

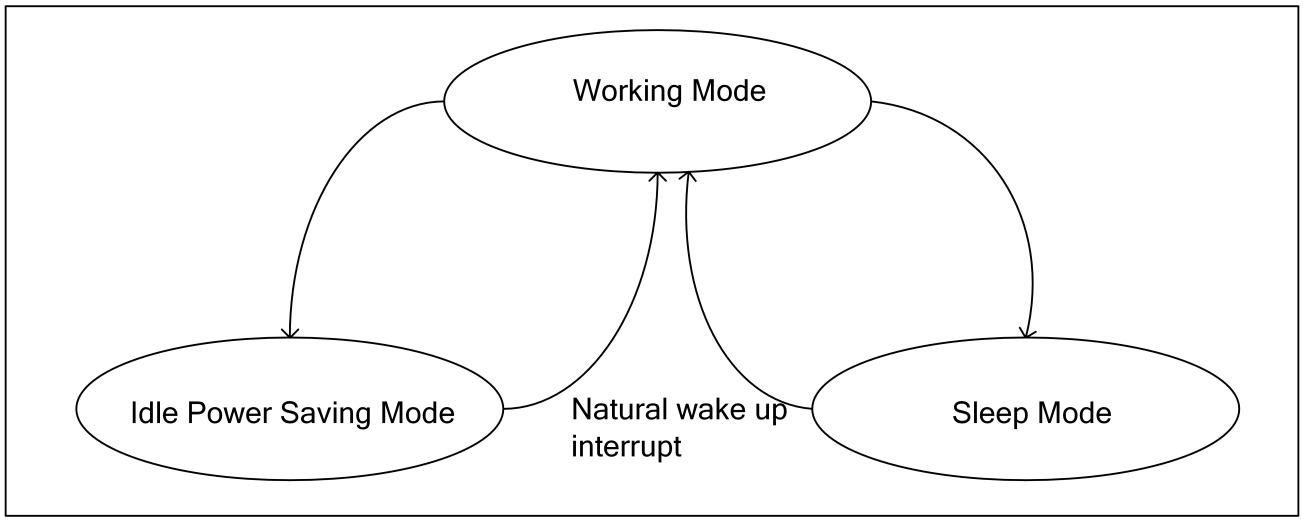


Fig. 24.1: Low power modes

24.2 Features

- Clock control: clock control of peripherals in GLB, small-scale power saving, and fast response speed
- Sleep control (PDS): 5 levels (PDS 1/2/3/7/15), large-scale power saving, moderate response speed
- Deep sleep control (HBN): 4 levels (HBN 0/1/2/3), global power saving, and long response time

24.3 Functional Description

24.3.1 Power Domain

There are 8 power domains in the xxx chip, with main functions as follows:

- PD_AON
  - HBN state machine, which can control the power/isolation unit/reset/clock of PD_AON_HBNRTC/PD_AON_-HBNCORE/PD_CORE
  - AON_PIN (GPIO16/17/18/19) pin wakeup function
  - Reserve 4 readable and writable registers to save data in PDS/HBN0/HBN1/HBN2 modes
  - BOR (Brown Out Reset) function
  - LDO11_AON/RT/SOC output voltage selection unit
  - Root, F32K, and Uart clock source selection
  - HBN_OUT0_PIR (Acomp0/Acomp1/bor/pir) wakeup mask, enable register, and interrupt status register
- PD_AON_HBNRTC
  - Select the control unit of RTC Counter clock source
  - RTC can be used for wakeup or LED flashing
  - RC32K and Xtal32X control functions
- PD_AON_HBNCORE
  - Partial power control register
  - Reserve HBN_RAM, which can be used to store programs and data, so that data will not disappear in PDS/HBN0 modes
  - In HBN mode, it has the function of controlling AON_PIN and keeping other IO
  - PIR digital control: PIR is a Passive Infra-Red (PIR) sensor, a peripheral in HBN area, which can be used as HBN wakeup source
  - Reserve LDO11SOC, LDO15RF, DCDC0, DCDC1 and DCDC2 control registers
  - Reserve XTAL, TSEN, Acomp0/1 and GPADC control registers
  - Reserve 4 readable and writable registers to save data in PDS/HBN0 modes
- PD_CORE
  - The PDS state machine controls the power, isolation unit, reset, clock, and memory of PD_CORE_MISC/PD_-USB/PD_CPU/PD_WB

  - Reserve PDS_RAM, so that data will not disappear in PDS mode
  - WIFI/BLE timer control
  - 160KB WRAM Retention/Sleep
  - Control the PDS interrupt and wakeup function
  - Reserve the control of GPIO0~15/20~36
  - Reserve the control of RC32M
  - PDS_Timer timer
- PD_CORE_MISC
  - PD_CORE_MISC_DIG and PD_CORE_MISC_ANA are collectively referred to PD_CORE_MISC
  - Peripheral (including I2C/SDIO/UART/FLASH/SPI/ADC/DAC/GPIO/PWM)
  - Chip GLB register
- PD_USB
  - USB controller
- PD_CPU
  - NP_CPU and cache unit
  - ROM, TCM
- PD_WB
  - WIFI PHY/MAC
  - BLE PHY/MAC
  - RF Controller

BL616/BL618 has a total of 10 power modes and 8 power domains. The states of the power domains corresponding to each power mode are shown in the table below:

Table 24.1: Power mode

NO.	Scenario	Power Domain							
		PD_AON	PD_AON_-HBNRTC	PD_-AON_HB-NCORE	PD_-CORE	PD_CORE_-MISC	PD_USB	PD_CPU	PD_WB
1	Normal	ON	ON	ON	ON	ON	ON	ON	ON
2	PDS1	ON	ON	ON	ON	ON	ON	ON	OFF
3	PDS2	ON	ON	ON	ON	ON	ON	OFF	ON

Table 24.1: Power mode(continued)

NO.	Scenario	Power Domain							
		PD_AON	PD_AON_-HBNRTC	PD_-AON_HB-NCORE	PD_-CORE	PD_CORE_-MISC	PD_USB	PD_CPU	PD_WB
4	PDS3	ON	ON	ON	ON	ON	ON	OFF	OFF
5	PDS7	ON	ON	ON	ON	ON	OFF	OFF	OFF
6	PDS15	ON	ON	ON	ON	OFF	OFF	OFF	OFF
7	HBN0	ON	ON	ON	OFF	OFF	OFF	OFF	OFF
8	HBN1	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF
9	HBN2	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF
10	HBN3	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

The power consumption of each of the 10 power modes is as follows.

Table 24.2: Power consumption of the power modes

NO.	Scenario	current/uA
1	Normal	30000
2	PDS1	930
3	PDS2	1400
4	PDS3	550
5	PDS7	540
6	PDS15	70
7	HBN0	3.1
8	HBN1	2.7
9	HBN2	2.1
10	HBN3	

24.3.2 Wake-up Source

The chip supports multiple wakeup sources, which can wake up the chip from different power modes.

The wake-up sources for different power modes are shown in the following table:

Table 24.3: Wakeup source

Power mode	Wakeup source
PDS1	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB
PDS2	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB/WIFI
PDS3	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM/USB
PDS7	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/GPIO/IRRX/DM
PDS15	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1/PDS_Timer/PDS_GPIO
HBN0	AON_PIN/BOR/RTC/Pir/Acomp0/Acomp1
HBN1	AON_PIN/RTC
HBN2	AON_PIN
HBN3	Reapply power to VDDIO2

24.3.3 Power Modes

Operating mode

The chip provides independent clock control between CPU and peripherals. The GLB and clock sections detail the clock control of each module. The software can perform clock control over CPUs or peripherals that are not in use according to the current application scenario. The clock control provides real-time response, so there is no need to worry about the response time in this working mode.

Power-down sleep mode

The power-down mode consumes less power than the working mode. In the PDS mode, the clocks other than RTC will be controlled and switched to the internal low-speed clocks, and the external crystal oscillator and PLL will be turned off to save more power, so there will be a time delay when entering and exiting this low-power mode. After the power-down sleep mode is enabled, the data in the OCRAM area can automatically switch to the “retention” state and remain, and automatically exit this state after wake-up.

1. Enter Idle Power Saving Mode

The software can make this module enter the power-down mode through PDS configuration, waiting for processing. After entering the wait for interrupt (WFI) mode, PDS will trigger the clock control module to perform the gate clock operation, and notify the analog circuit to turn off the PLL and external crystal oscillator

2. Exit Idle Power Saving Mode

There are two ways to exit the idle power saving mode. One is that a specific interrupt or event stops the idle state. The other is that the time in PDS_TIM set by the software is met. Both will trigger PDS to exit the power-down mode. Considering that it takes about 1 ms to turn on the crystal oscillator, PDS allows software to turn on the crystal oscillator in advance, to wake up PDS faster. When PDS is ready to wake up, it will notify CPU to exit the WFI mode through

interrupt.

HBN

In the sleep mode, most of the chip logic is powered off (Vcore) while the AON power is kept ON, and the internal circuit will not wake up until an external event is received. This mode can achieve ultimate power saving, but it takes the longest response time compared with the first two modes, so it suits the case where the chip does not need to work for a long time, to prolong the battery life. As most circuits will be powered off in this mode, the corresponding register values and memory data will disappear. Therefore, there is a 4 KB HBN_RAM reserved in HBN that will not be powered off in sleep state. The data or state that the software needs to save can be copied to this memory before the chip enters the sleep mode. When recovering from the sleep mode, the chip can access data directly from RAM, which can usually be used as a record of state or a quick data recovery.

24.3.4 IO wakeup

PDS1~15 and HBN0~2 all support IO wakeup, but the specific implementation methods are different.

Table 24.4: IO wakeup

Power mode	IO wake-up implementation method
PDS1~7	Wake up with GLB_IO
PDS15	Use PDS_IO wakeup (high level/falling edge/rising edge of low level before sleep) and AON_IO wakeup (low level/high level/falling edge/rising edge/rising or falling edge)
HBN0~2	Wake up with AON_IO

GLB_IO wakes up PDS1~7 mode

All IOs support waking up PDS1~7 mode, and the PDS1~7 mode starts IO wakeup code as follows:

Listing 24.1: glb io wakeup pds1~7

```
BL_Err_Type glbio_wakeup(uint32_t gpio_id, uint32_t trigger_type)
{
   gpio_set_mode(gpio_id, trigger_type);
   gpio_irq_enable(gpio_id, ENABLE);
   gpio_attach_irq(gpio_id, pm_gpio_callback);

   pm_pds_wakeup_src_en(PDS_WAKEUP_BY_GLB_GPIO_IRQ_EN_POS);
}
```

PDS_IO wakes up PDS15 mode

PDS_IO is all IOs except AON_IO (GPIO16~19), which support to wake up PDS15 mode.

1. Wake-up modes supported by PDS_IO

- Synchronous falling edge
- Synchronous high level
- asynchronous falling edge
- asynchronous high level

2. Grouping of PDS_IO wake-up modes

The wake-up mode of PDS_IO can be divided into four groups: GPIO0~7, GPIO8~15, GPIO20~27, and GPIO28~34. Each group of GPIOs share the same wake-up mode, for example, GPIO0 is set to high-level wake-up, GPIO7 can only be configured as a high-level wake-up, while GPIO20 can be configured as a falling-edge wake-up.

Attention: If PDS_IO wakeup is enabled in PDS15 state, the internal PU/PD of that GPIO must be enabled, otherwise the current will increase to 1000+uA.

3. Grouping of PDS_IO PUPD configuration

The PUPDIE configuration of PDS_IO can be divided into two groups: GPIO0~15 and GPIO20~34. Each group of GPIOs share the same PUPDIE configuration, for example, GPIO0 is set as PU, GPIO8 can only be configured as PU, and GPIO20 can be configured as PD.

Attention: Since GPIO0~15 share a group of PUPDIE configurations, the trigger configuration of GPIO0~7 and GPIO8~15 may be limited. For example, if GPIO0~15 is set as PU, then the wake-up mode of GPIO0~7 and GPIO8~15 cannot be selected as high-level wake-up. Similarly, since GPIO20~34 share a group of PUPDIE configurations, the trigger configuration of GPIO20~27 and GPIO28~34 may be limited.

Attention: When using GPIO0~15 as the wake-up pin, it should be noted that GPIO2 is the BOOT pin, which is usually connected with a pull-down resistor. pull-up resistors, When GPIO0~15 is set to PD, it may cause leakage due to GPIO4 pull-up.

When GPIO0~15 are used as wake-up pins, if GPIO2 and GPIO4 have pull-down resistors and pull-up resistors at the same time, it is recommended to set GPIO0~15 as PU, and change the pull-down resistors of BOOT pins to 1M ohms to reduce leakage current.

Attention: When using GPIO20~34 as wake-up pins, you need to pay attention that GPIO21 and 22 may be used as UART pins, and the idle state of the serial port is high, so it is recommended to set the internal pull-up; if GPIO21 and 22 are not connected to the serial port, then You can configure pull-up or pull-down.

For example, the PDS15 mode starts the IO3 falling edge wake-up code as follows:

```
PDS_Set_GPIO_Pad_IntClr(PDS_GPIO_INT_SET_1_GPIO0_GPIO7);
PDS_Set_GPIO_Pad_Pn_Pu_Pd_Ie(PDS_GPIO_GROUP_SET_GPIO0_GPIO15, 1, 0, 1);
PDS_Set_GPIO_Pad_IntMode(PDS_GPIO_INT_SET_1_GPIO0_GPIO7,
                         (PDS_GPIO_INT_TRIG_Type)trigger_type);
PDS_Set_GPIO_Pad_IntMask((GLB_GPIO_Type)GLB_GPIO_PIN_3, UNMASK);

pm_pds_wakeup_src_en(PDS_WAKEUP_BY_PDS_GPIO_IRQ_EN_POS);
```

Since the PU/PD inside the chip is set according to groups during low power consumption, in actual use, it is difficult to ensure that all GPIOs in this group are pull-up or pull-down. Therefore, it is recommended to add pull-up or pull-down resistors to GPIO on the PCB to avoid leakage. It is not recommended to use the PU/PD inside the chip to implement pull-up or pull-down.

For the BL616 QFN40 package, GPIO4~9/23~26/31~34 are not sealed, and the internal pins are wired and connected to GND. If the PU inside the chip is used to implement pull-up at this time, these pins will leak current. To avoid this situation, the driver interface of the SDK has prohibited users from setting the pull-up of the chip.

AON_IO wake up PDS15 mode

PDS15 mode supports AON_IO (GPIO16~19) wakeup, AON_IO share the same trigger mode, but PUPDIE configuration is independent. For example, if GPIO16 is set to wake up by falling edge trigger, GPIO17~19 cannot be configured as other trigger methods. If GPIO16 is set as PU, GPIO17~19 can not be configured as PU.

```
static BL_Err_Type aonio_wakeup_pds15(uint32_t pin, uint32_t trigger_type)
{
   uint32_t maskVal = 0xF;
   HBN_AON_PAD_CFG_Type aonPadCfg;

   aonPadCfg.ctrlEn = 1;
   aonPadCfg.ie = 1;
   aonPadCfg.oe = 0;
   aonPadCfg.pullUp = 0;
   aonPadCfg.pullDown = 0;
   HBN_Aon_Pad_Cfg(DISABLE, (pin - 16), &aonPadCfg);
   maskVal = 0xF & ~(1 << (pin - 16));
   HBN_Aon_Pad_WakeUpCfg(DISABLE, trigger_type, maskVal, 0, 7);

   pm_hbn_out0_irq_register();

   pm_pds_wakeup_src_en(PDS_WAKEUP_BY_HBN_IRQ_OUT_EN_POS);

   pm_pds_mode_enter(PM_PDS_LEVEL_15,0);
```

(continues on next page)

(continued from previous page)

```
}
```

AON_IO wake up HBN0~2 mode

HBN0~2 mode supports AON_IO (GPIO16~19) wakeup, and AON_IO share the same trigger mode, but PUPDIE configuration is independent. For example, if GPIO16 is set to wake up by falling edge trigger, GPIO17~19 cannot be configured with other trigger methods. If GPIO16 is set as PU, GPIO17~19 can not be configured as PU.

```
static BL_Err_Type aonio_wakeup_hbn(uint32_t pin, uint32_t trigger_type)
{
   uint32_t maskVal = 0xF;
   HBN_AON_PAD_CFG_Type aonPadCfg;

   aonPadCfg.ctrlEn = 1;
   aonPadCfg.ie = 1;
   aonPadCfg.oe = 0;
   aonPadCfg.pullUp = 0;
   aonPadCfg.pullDown = 0;
   HBN_Aon_Pad_Cfg(DISABLE, (pin - 16), &aonPadCfg);
   maskVal = 0xF & ~(1 << (pin - 16));
   HBN_Aon_Pad_WakeUpCfg(DISABLE, trigger_type, maskVal, 0, 7);

   pm_hbn_mode_enter(PM_HBN_LEVEL_0,0);
}
```

24.3.5 ACOMP wake up

The chip supports ACOMP wake-up. ACOMP has two comparison modules. Each ACOMP module has two input ports, a positive input channel and a negative input channel. Each input port can be selected:

-ADC channel0-7 - Output A/B of the DAC - System 1.25V reference voltage (this voltage does not exist in low power consumption mode) - Voltage division of VDD33 (voltage division coefficient can be adjusted)

The partial pressure coefficient can choose one of the following:

- 0.25
- 0.5
- 0.75
- 1

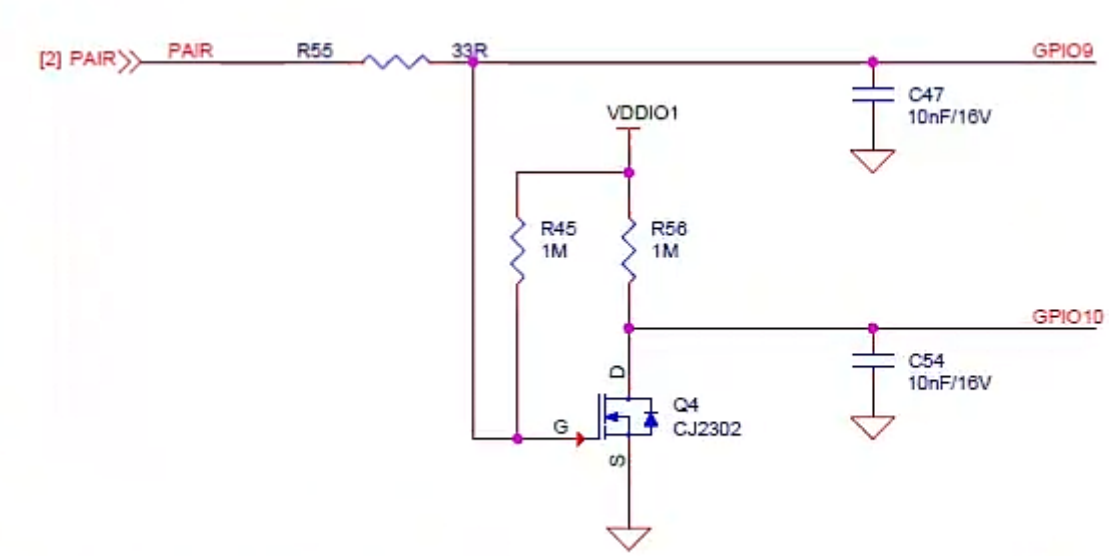
In conventional applications, generally set the positive input channel to one of ADC channel0-7, that is, use one of GPIO2, 3, 10, 12, 13, 14, 19, and 20, and select the voltage divider of VDD33 for the negative pole. The voltage coefficient can be selected according to the actual application scenario. When the positive input voltage changes

from less than the negative voltage to greater than the negative voltage, a rising edge interrupt wake-up source is generated, and a falling edge is generated when the positive input voltage changes from greater than the negative voltage to less than the negative voltage. Interrupt wakeup source. When one ACOMP is enabled, the current increases by about 7uA, and when two ACOMPs are enabled, the current increases by about 10uA.

24.3.6 IO dual-edge wake-up

AON_IO and ACOMP support double-edge wake-up.

PDS_IO needs to use two pins to realize double-edge wake-up, both of which are set to fall-edge wake-up. With the following circuit, double-edge wake-up can be realized:



However, in the scenario of double-edge wake-up, the input of the pin may be high or low, so the internal pull-up and pull-down cannot be enabled; otherwise, once the pin level is opposite to the internal pull-up and pull-down configuration, it will Leakage occurs, leakage current = 3.3V/resistance. Since the pull-up and pull-down cannot be enabled internally, other pins in the same group do not have internal pull-up and pull-down. The pcd needs to cooperate with external pull-up and pull-down to ensure that the pin is not suspended. If the user can ensure that the external input during the chip sleep period is not If it is suspended, you don’t need to pull it up and down.

For example, if GPIO15 and GPIO20 are connected to the same input source, and GPIO15 and GPIO20 are set to wake up on the falling edge, then GPIO0~15 and GPIO20~34 cannot enable internal pull-up and pull-down, and GPIO0~14 and GPIO21~34 pins are external A pull-up resistor is required.

Error: GPIO15 and GPIO20 are connected to the same input source, GPIO15 is set to wake up at falling edge, and GPIO20 is set to wake up at high level; when the level changes from low to high, the chip wakes up from PDS mode, but the value of PDS_GPIO_STAT register is 0x00014000, which means that at the same time High level wake-up and falling edge wake-up, but the oscilloscope did not catch the falling edge.

24.3.7 IO wakeup trigger condition

The requirements for level timing of io wakeup are as follows:

	time / us
pds io	100
aon io	100
acomp	50

For example, GPIO0 is used as the pds_io wake-up pin, and the trigger mode is falling edge, then the timing of the level must meet the requirement that the low level remains greater than or equal to 100us after the falling edge appears.

GPIO18 is used as aon_io wake-up pin, and the trigger mode is high level, so the timing of the level must be kept high for a time greater than or equal to 100us.

GPIO13 wakes up as acomp, and the trigger mode is rising edge wakeup, then the timing of the level must meet the requirement that the high level remains greater than or equal to 50us after the rising edge appears.

24.4 POR & BOD

BL616 incorporates built-in circuits for Under Voltage Lockout (UVLO), Power On Reset (POR), and Brown Out Detector (BOD).

24.4.1 UVLO and POR

When the supply voltage VDD is below 1.8V, the UVLO circuit cannot function properly, resulting in an uncertain chip state and undefined current. When the supply voltage VDD is between 1.8V and 2.0V, the UVLO circuit operates normally, maintaining a high logic level. The chip disconnects all internal power supplies, resulting in very low overall current consumption. As the supply voltage VDD gradually rises to 2.0V, the UVLO output goes low, and the POR signal goes high. After a delay of 2ms, the POR signal transitions from high to low, releasing the system reset.

Note: UVLO and POR circuits operate by default upon power-up and do not require software configuration.

24.4.2 BOD

BOD offers 8 selectable levels: 2.4V, 2.35V, 2.3V, 2.25V, 2.2V, 2.15V, 2.1V, and 2.05V. BOD has a hysteresis of 50mV. Taking the example of the 2.4V threshold, when the supply voltage VDD is below 2.4V, the BOD signal remains high. When VDD exceeds 2.45V, the BOD signal goes low.

Note: When the supply voltage VDD is below 2.0V, the UVLO circuit shuts down the BOD circuit.

Note: BOD circuit is disabled by default upon power-up and requires software enablement to function.

BOD can be used in two modes: interrupt and reset. In the interrupt mode, an interrupt is triggered when BOD transitions to a high level, but no interrupt is triggered when BOD returns to a low level. In the reset mode, BOD and POR are associated. When both BOD and POR signals are high, a system reset occurs, including the BOD circuit. After the reset, BOD is disabled by default. BOD and POR signals transition from high to low, resulting in a chip reset. BOD remains inactive until it is enabled again, and BOD and POR are associated, triggering another system reset.

BOD reset resets the registers, but RAM data is retained, including HBNRAM, WRAM, and SRAM.

25 SEC ENG

25.1 Overview

SEC ENG has a variety of built-in computing modules, including AES, SHA, CRC, GMAC, PKA, and TRNG.

25.2 Features

- AES
  - Supports 128-bit, 192-bit and 256-bit key lengths
  - Support encryption and decryption of multiple link modes (ECB/CBC/CTR/XTS)
  - Exclusive AES LINK function
- SHA
  - Support SHA1/SHA224/SHA256/SHA384/SHA512
  - Exclusive SHA LINK function
- Support MD5, CRC-16, CRC-32

25.3 Functional Description

25.3.1 AES Accelerator

25.3.1.1 key

The key length required for encryption mode and decryption mode can be selected by configuring se_aes_0_mode and se_aes_0_dec_en in the register se_aes_0_ctrl.

amode	adec en	Operation
0	0	AES-128 encryption
1	0	AES-256 encryption
2	0	AES-192 encryption
0	1	AES-128 decryption
1	1	AES-256 decryption
2	1	AES-192 decryption

Fig. 25.1: AES operation mode diagram

Select whether to enable the hardware key through aes_0_hw_key_en in the register se_aes_0_ctrl. If you use a software key, you also need to configure the registers se_aes_0_key_0~se_aes_0_key_7 to store the key, and each register stores a 4-byte key.

25.3.1.2 link mode

Different link modes can be selected through se_aes_0_block_mode in the register se_aes_0_ctrl. Currently, ECB, CBC, CTR, and XTS modes are supported.

25.3.1.3 Plaintext, ciphertext

- Plaintext or ciphertext needs to be a multiple of 16
- The register se_aes_0_msa stores the plaintext address entered during encryption or the ciphertext address entered during decryption
- The register se_aes_0_mda stores the ciphertext address output during encryption or the plaintext address output during decryption
- se_aes_0_msg_len in the register se_aes_0_ctrl is used to set the length of ciphertext or plaintext (in units of 16 bytes)

25.3.1.4 initialization vector

The registers se_aes_0_iv_0~se_aes_0_iv_3 store the initialization vector IV. You can choose whether to use a new iv by configuring aes_0_iv_sel in se_aes_0_ctrl. You need to clear 0 when configuring iv for the first time, and you need to set 1 if you continue to use this iv or automatically update iv.

25.3.1.5 Encryption and decryption configuration process

- Enable AES with se_aes_0_en in the configuration register se_aes_0_ctrl
- Configuration register se_aes_0_endian, including se_aes_0_dout_endian, se_aes_0_din_endian, se_aes_0_key_endian, se_aes_0_iv_endian, se_aes_0_twk_endian, if the value is 0, it means little-endian, if the value is 1, it means big-endian
- se_aes_0_block_mode in the configuration register se_aes_0_ctrl selects the link mode
- se_aes_0_mode in the configuration register se_aes_0_ctrl to select the key length
- To use software key, configure registers se_aes_0_key_0~se_aes_0_key_7 to store the key. To use a hardware key, set aes_0_hw_key_en in register se_aes_0_ctrl
- Configure registers se_aes_0_iv_0~se_aes_0_iv_3 to set IV, the filling order for MSB is se_aes_0_iv_0~se_aes_0_iv_3, and the filling order for LSB is se_aes_0_iv_3~se_aes_0_iv_0
- se_aes_0_dec_en in the configuration register se_aes_0_ctrl selects the encryption or decryption mode
- The configuration register se_aes_0_msa sets the source address of the data to be processed
- The configuration register se_aes_0_mda sets the destination address where the processing result is stored
- se_aes_0_msg_len in the configuration register se_aes_0_ctrl sets the length of the data to be processed, in units of 16 bytes
- se_aes_0_trig_1t in the configuration register se_aes_0_ctrl triggers AES to run

The result is output to the destination address specified by se_aes_0_mda.

25.3.2 SHA Accelerator

25.3.2.1 SHA mode

The se_sha_0_mode in the register se_sha_0_ctrl: 0:SHA-256 1:SHA-224 2:SHA-1 3:SHA-1 4:SHA-512 5:SHA-384 6:SHA-512/224 7:SHA-512/256

The se_sha_0_mode_ext in the register se_sha_0_ctrl: hash mode extention; 0:SHA 1:MD5 2:CRC-16 3:CRC-32

25.3.2.2 Plaintext and ciphertext

- The register se_sha_0_msa stores the plaintext address.
- The register se_sha_0_hash_l_0~se_sha_0_hash_l_7 stores the ciphertext.

25.3.2.3 Operation flow

- Configure se_sha_0_mode in the register se_sha_0_ctrl to set the specific mode of SHA
- Enable SHA by configuring se_sha_0_en in the register se_sha_0_ctrl
- Configure se_sha_0_hash_sel in the register se_sha_0_ctrl; 0 means starting a new HASH calculation, and 1 means using the last result for HASH calculation
- Configure the register se_sha_0_msa to set the source address of the data to be processed
- Configure se_sha_0_msg_len in the register se_sha_0_ctrl to set the length of the data to be processed (512 bits for SHA1, SHA224 and SHA256, while 1024 bits for SHA512, SHA384, SHA512/224, and SHA512/256)
- Configure se_sha_0_trig_1t in the register se_sha_0_ctrl to trigger SHA
- The output result is stored in se_sha_0_hash_l_0~se_sha_0_hash_l_7, MSB:se_sha_0_hash_l_0~se_sha_0_-hash_l_7, LSB:se_sha_0_hash_l_7~se_sha_0_hash_l_0

25.3.3 Random Number Generator (RNG)

The random numbers generated by the built-in true RNG can be used as the basis for encryption and other operations.

- True random numbers: They are generated through physical phenomena, such as coin tossup, dicing, wheel rotation, noise from using electronic components, and nuclear fission. Such RNGs are called physical RNGs, and their weaknesses are high technical requirements.
- Pseudo-random numbers: Truly random numbers (or random events) are randomly generated in a generation process according to the distribution probability shown in the experimental process, and the result is unpredictable and invisible. The random function in the computer is simulated according to a specified algorithm, and its result is deterministic and visible. We may consider that the probability of this foreseeable result is 100%. Hence the

“random number” generated by computer random function is not truly random, but pseudo-random.

25.3.3.1 Usage process

- Enable TRNG by configuring se_trng_0_en in the register se_trng_0_ctrl_0
- Configure se_trng_0_trig_1t in the register se_trng_0_ctrl_0 to trigger TRNG
- The output result is stored in se_trng_0_dout_0~se_trng_0_dout_7

26 Revision history

Table 26.1: Document revision history

Date	Revision	Changes
2021/9/22	0.9	Initial version
2022/8/22	0.91	Add register description
2022/11/28	0.92	Add the memory map address corresponding to Non-cache and cache
2022/12/22	0.93	Modify Reset table and add ACOMP description
2023/3/30	0.94	Modify the I2C sequence diagram, fix typos
2023/4/28	0.95	Update Clock Tree Diagram
2023/5/23	0.96	Add CHIP_EN reset function in Reset table
2024/3/5	0.97	Modify EMAC features
2025/1/15	0.98	Add BOOT process