BL808

Reference Manual

Version: 1.0

www.bouffalolab.com

Contents

List of Figures

List of Tables

1 System and Memory

1.1 System Architecture

BL808 series chips mainly include wireless and multimedia subsystems.

With one RISC-V 32-bit high-performance CPU and Wi-Fi/BT/Zigbee, the wireless subsystem supports wireless connection and data transmission in multiple ways, creating diversified experience.

With one RISC-V 64-bit ultra-high performance CPU and DVP/CSI/ H264/NPU, the multimedia subsystem can be widely used in AI fields like video surveillance and smart speakers.

BL808 system architecture:

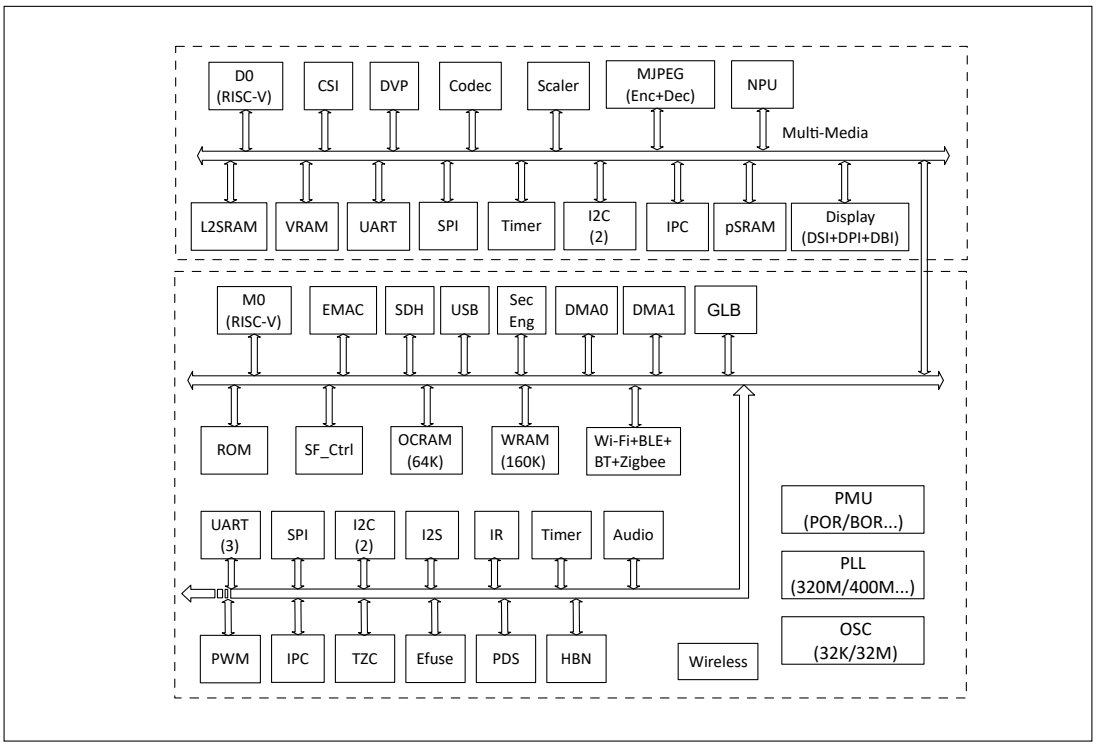


Fig. 1.1: Block Diagram of System

Composition of wireless subsystem:

- CPU
  - One RISC-V high-performance CPU (M0) and one RISC-V low-power CPU (LP)
  - Wireless connection and data transmission in multiple ways, diversified user experience
- Bus interface: AXI and AHB
  - CPU accesses the memory through AXI
  - CPU accesses peripherals through AHB
  - Low power and high performance
- Peripherals
  - Has 30 peripheral modules including UART/SPI/I2C/PWM/SDH/EMAC/USB, allowing connection and storage to and control of peripherals
  - Integrates the Audio module, and supports the input of audio data, analog microphone, and digital microphone, making audio application development easy for users
  - Suits low-power application scenarios, for example, disabling other CPUs and peripherals in low-power scenarios of smart speakers, and using LP for wake-up test of VAD

Composition of multimedia subsystem:

- CPU
  - One RISC-V ultra-high performance CPU (D0)
  - For application fields like video surveillance, smart doorbell, smart cat eye, and smart speaker
  - NPU module for various AI scenarios that accelerates neural network operators
- Peripherals
  - Conventional peripherals like UART/SPI/I2C, facilitating development of system functions
  - CMOS Sensor connected with CSI and DVP interfaces to acquire images and videos; built-in Codec modules for video coding; OSD/Scaler/2DDMA modules for importing video data stream into the Display module

1.2 CPU

BL808 chip contains three RISC-V CPUs, namely M0, D0, and LP. M0 mainly processes network data stream and peripheral access. LP mainly implements some functions in the low-power mode.

1.2.1 M0

1.2.1.1 Features

M0, a 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
- 32K instruction cache, two-way set associative structure
- 16K 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

1.2.1.2 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

1.2.1.3 Implemented Standards

M0 is compatible with the following RISC-V standards:

- 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

1.2.2 D0

1.2.2.1 Features

D0 is a 64-bit RISC-V CPU with 5-stage pipeline. This ultra-high performance embedded microprocessor suits fields like security monitoring, intelligent audio processing, and intelligent video processing, and has the following features:

- 64-bit RISC processor
- Supports RISC-V RV64IMAFCV instruction architecture
- Five-stage single-issue sequentially executed pipeline
- Level-1 instruction and data cache of Harvard architecture, with a size of 32 KB and a cache line of 64B
- Sv39 memory management unit, realizing the conversion of virtual and real addresses and memory management
- jTLB that supports 128 entries

- Supports AXI 4.0 128-bit master interface
- Supports core local interrupt (CLINT) and platform-level interrupt controller (PLIC)
- With 80 external interrupt sources, 3 bits for configuring interrupt priority
- Supports BHT (8K) and BTB
- Compatible with RISC-V PMP, 8 configurable areas
- Supports hardware performance monitor (HPM) units

Features of the vector computing unit:

- Compatible with RISC-V V vector extension standard (revision 0.7.1)
- Supports vector execution unit (128 bits)
- Supports INT8/INT16/INT32/FP16/FP32 vector operations
- Supports segment load and store instructions

1.2.2.2 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
- Supports unaligned memory access, which can be enabled/disabled by software, making programming and debugging easy for software programmers

1.2.2.3 Implemented Standards

D0 is compatible with the following RISC-V standards:

- 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 “V” Vector Extension, Version 0.7.1-20190610-Workshop-Release.
- RISC-V External Debug Support Version 0.13.2

1.2.3 LP

1.2.3.1 Features

LP, a 32-bit RISC-V CPU, uses a 2-stage pipeline (fetch, decode, execute, and write back), characterized by ultra-low power, ultra-low cost, high code density, and the following features:

- 32-bit RISC processor
- Supports RISC-V RV32EMC instruction set
- Supports RISC-V 32-bit/16-bit mixed instruction set
- Sixteen 32-bit integer GPRs
- Two-stage sequentially executed pipeline
- Supports RISC-V machine mode and user mode
- Compatible with RISC-V CLIC interrupt standard and interrupt nesting, with 32 external interrupt sources
- Supports AHBLite bus protocol, instruction bus, and system bus
- No internal cache
- No PMP
- Supports 2-wire debugging interface

1.2.3.2 Implemented Standards

LP is compatible with the following RISC-V standards:

- 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

1.3 Boot

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

Table 1.1: Boot mode

Boot Pin	Level	Description
GPIO39	1	Boot from UART (GPIO20/21)/USB for downloading Flash or downloading image to RAM for execution
	0	Boot the application image from Flash

1.4 Address Mapping

Table 1.2: Address mapping

Module	Destination	Start Address	Size	Description
FLASH	FlashA	0x58000000	64MB	Application address space
PSRAM	pSRAM	0x54000000	64MB	pSRAM memory address space
RAM	OCRAM(MCU)	0x22020000	64KB	On Chip RAM address space mainly for M0 application data
	WRAM(MCU)	0x22030000	160KB	Wireless RAM address space mainly for M0 wireless network data
	XRAM(EMI)	0x40000000	16KB	Shared RAM mainly for data communication among M0/LP/D0
	DRAM(MM)	0x3EF80000	512KB	Multimedia-side RAM address space used by D0 application data and modules like H264/NPU
	VRAM(MM)	0x3F000000	32KB	Multimedia-side RAM address space used by D0 application data and modules like H264/NPU
MMPERI	TIMER1	0x30009000	4KB	TIMER1 control register
	SPI1	0x30008000	4KB	SPI1 control register
	MM_GLB	0x30007000	4KB	Multimedia-side global register
	DMA2D	0x30006000	4KB	DMA2D control register
	I2C3	0x30004000	4KB	I2C3 control register
	I2C2	0x30003000	4KB	I2C2 control register
	UART3	0x30002000	4KB	UART3 control register
	DMA2	0x30001000	4KB	DMA2 control register
MCUPERI	DMA1	0x20071000	4KB	DMA1 control register
	EMAC	0x20070000	4KB	EMAC control register
	AUDIO	0x20055000	4KB	Audio control register
	USB	0x20072000	4KB	USB control register
	HBN	0x2000F000	4KB	HBN register
	PDS	0x2000E000	4KB	PDS register
	DMA0	0x2000C000	4KB	DMA0 control register
	I2S	0x2000AB00	256B	I2S control register
	ISO11898	0x2000AA00	256B	ISO11898 control register
	UART2	0x2000AA00	256B	UART2 control register
	I2C1	0x2000A900	256B	I2C1 control register
	IR	0x2000A600	256B	IR control register
	TIMER0	0x2000A500	256B	TIMER0 control register
	PWM	0x2000A400	256B	PWM control register
	I2C0	0x2000A300	256B	I2C0 control register
	SPI0	0x2000A200	256B	SPI0 control register

Table 1.2: Address mapping

Module	Destination	Start Address	Size	Description
MCUPERI	UART1	0x2000A100	256B	UART1 control register
	UART0	0x2000A000	256B	UART0 control register
	eFuse	0x20056000	4KB	eFuse memory control register
	TZ	0x20005000	4KB	TrustZone control register
	SEC_ENG	0x20004000	4KB	Security engine control register
	GPIP	0x20002000	1KB	General DAC / ADC / ACOMP interface control register
	GLB	0x20000000	4KB	Global control register
ROM	ROM	0x90000000	128KB	Bootrom area address space

1.5 Interrupt Source

CPU_M0 and CPU_LP contain 23 interrupt sources. The interrupt sources and interrupt numbers are shown in the following table:

Table 1.3: Interrupt assignment

Interrupt source		Interrupt Number	Description
DMA	DMA0_ALL	IRQ_NUM_BASE+15	DMA0 ALL Interrupt
	DMA1_ALL	IRQ_NUM_BASE+16	DMA1 ALL Interrupt
IR	IRTX	IRQ_NUM_BASE+19	IR TX Interrupt
	IRRX	IRQ_NUM_BASE+20	IR RX Interrupt
USB	USB	IRQ_NUM_BASE+21	USB Interrupt
EMAC	EMAC	IRQ_NUM_BASE+24	EMAC Interrupt
ADC	GPADC_DMA	IRQ_NUM_BASE+25	GPADC_DMA Interrupt
SPI	SPI0	IRQ_NUM_BASE+27	SPI Interrupt
UART	UART0	IRQ_NUM_BASE+28	UART0 Interrupt
	UART1	IRQ_NUM_BASE+29	UART1 Interrupt
	UART2	IRQ_NUM_BASE+30	UART2 Interrupt
GPIO	GPIO_DMA	IRQ_NUM_BASE+31	GPIO DMA Interrupt
I2C	I2C0	IRQ_NUM_BASE+32	I2C0 Interrupt
	I2C1	IRQ_NUM_BASE+39	I2C1 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

Table 1.3: Interrupt assignment

Interrupt source		Interrupt Number	Description
I2S	I2S	IRQ_NUM_BASE+40	I2S Interrupt
GPIO	GPIO_INT0	IRQ_NUM_BASE+44	GPIO Interrupt
PDS	PDS_WAKEUP	IRQ_NUM_BASE+50	PDS Wakeup Interrupt
HBN	HBN_OUT0	IRQ_NUM_BASE+51	Hibernate out 0 Interrupt
	HBN_OUT1	IRQ_NUM_BASE+52	Hibernate out 1 Interrupt

CPU_D0 contains 21 interrupt sources. The interrupt sources and interrupt numbers are shown in the following table:

Table 1.4: Interrupt assignment

Interrupt source		Interrupt Number	Description
UART	UART3	IRQ_NUM_BASE+4	UART3 Interrupt
I2C	I2C2	IRQ_NUM_BASE+5	I2C2 Interrupt
	I2C3	IRQ_NUM_BASE+6	I2C3 Interrupt
SPI	SPI1	IRQ_NUM_BASE+7	SPI1 Interrupt
DMA2	DMA2_INT0	IRQ_NUM_BASE+24	DMA INT0 Interrupt
	DMA2_INT1	IRQ_NUM_BASE+25	DMA INT1 Interrupt
	DMA2_INT2	IRQ_NUM_BASE+26	DMA INT2 Interrupt
	DMA2_INT3	IRQ_NUM_BASE+27	DMA INT3 Interrupt
	DMA2_INT4	IRQ_NUM_BASE+28	DMA INT4 Interrupt
	DMA2_INT5	IRQ_NUM_BASE+29	DMA INT5 Interrupt
	DMA2_INT6	IRQ_NUM_BASE+30	DMA INT6 Interrupt
	DMA2_INT7	IRQ_NUM_BASE+31	DMA INT7 Interrupt
EMAC	EMAC2	IRQ_NUM_BASE+36	EMAC2 Interrupt
DMA2D	DMA2D_INT0	IRQ_NUM_BASE+45	DMA2D INT0 Interrupt
	DMA2D_INT1	IRQ_NUM_BASE+46	DMA2D INT1 Interrupt
PWM	PWM	IRQ_NUM_BASE+48	PWM1 Interrupt
TIMER1	TIMER0_CH0	IRQ_NUM_BASE+61	Timer1 Channel 0 Interrupt
	TIMER0_CH1	IRQ_NUM_BASE+62	Timer1 Channel 1 Interrupt
	TIMER0_WDT	IRQ_NUM_BASE+63	Timer1 Watch Dog Interrupt
AUDIO	AUDIO	IRQ_NUM_BASE+64	Audio Interrupt
PDS	PDS	IRQ_NUM_BASE+66	PDS Interrupt

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

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

Table 2.1: Software reset function table 1

BL808	RST_PIN /Watch Dog / PDS Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS	PDS /CPU	PDS
CPU(M0)	✓			✓	
CPU(LP)	✓				
bus	✓		✓		
glb	✓	swrst_s1[0]			
mix	✓	swrst_s1[1]			✓
gpip	✓	swrst_s1[2]	✓		
sec_eng	✓	swrst_s1[4]	✓		
TZ	✓		✓		
efuse	✓				
dma	✓	swrst_s1[12]	✓		
psram	✓	swrst_s1_ext[2]	✓		
usb	✓	swrst_s1_ext[3]	✓		
emac	✓	swrst_s1_ext[3]	✓		
audio	✓	swrst_s1_ext[5]	✓		
dma2	✓	swrst_s1_ext[8]	✓		

Table 2.1: Software reset function table 1

BL808	RST_PIN /Watch Dog / PDS Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])/ PDS	PDS /CPU	PDS
pds		swrst_s1[14]			
uart0	✓	swrst_s1a[0]	✓		
uart1	✓	swrst_s1a[1]	✓		
spi	✓	swrst_s1a[2]	✓		
i2c	✓	swrst_s1a[3]	✓		
pwm	✓	swrst_s1a[4]	✓		
timer	✓	swrst_s1a[5]	✓		
irr	✓	swrst_s1a[6]	✓		
uart2/can	✓	swrst_s1a[10]	✓		
i2s	✓	swrst_s1a[11]	✓		
pdm	✓	swrst_s1a[12]	✓		
wifi	✓	swrst_s2[0]			✓
ble	✓	swrst_s3[0][2]			✓

Table 2.2: Software reset function table 2

BL808_MM	Watch Dog / Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])	MMCPU0
CPU(D0)	✓			✓
bus	✓		✓	
mm_misc	✓	swrst_mm_misc	✓	
mm_dma	✓	swrst_dma	✓	
mm_2ddma	✓	swrst_dma2d	✓	
mm_uart	✓	swrst_uart0	✓	
mm_i2c	✓	swrst_i2c0	✓	
mm_ipc	✓	swrst_i2c1	✓	
mm_timer	✓	swrst_timer	✓	
uhs_ctrl	✓	swrst_pUHS	✓	
disp_tsrc	✓	swrst_dp_tsrc	✓	
nr3d_ctrl	✓	swrst_nr3d_ctrl	✓	
dvp2busA	✓	swrst_dvp2busA	✓	
dvp2busB	✓	swrst_dvp2busB	✓	

Table 2.2: Software reset function table 2

BL808_MM	Watch Dog / Software Power On (swrst_cfg2[0])	SW.Reset (swrst_cfg1)	System Reset (swrst_cfg2[2])	MMCPU0
dvp2busC	✓	swrst_dvp2busC	✓	
dvp2busD	✓	swrst_dvp2busD	✓	
dvp2busE	✓	swrst_dvp2busE	✓	
dvp2busF	✓	swrst_dvp2busF	✓	
dvp2busG	✓	swrst_dvp2busG	✓	
dvp2busH	✓	swrst_dvp2busH	✓	
jdec	✓	swrst_mjpeg_dec	✓	
blai	✓	swrst_cnn	✓	

2.3 Clock Source

Types:

- XTAL: external crystal oscillator clock, with optional frequencies of 24, 32, 38.4, and 40 MHz depending on system requirements
- XTAL32K: external crystal oscillator clock, with frequency of 32 kHz
- RC32K: RC oscillator clock with frequency of 32 kHz 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, ADC, and DAC.

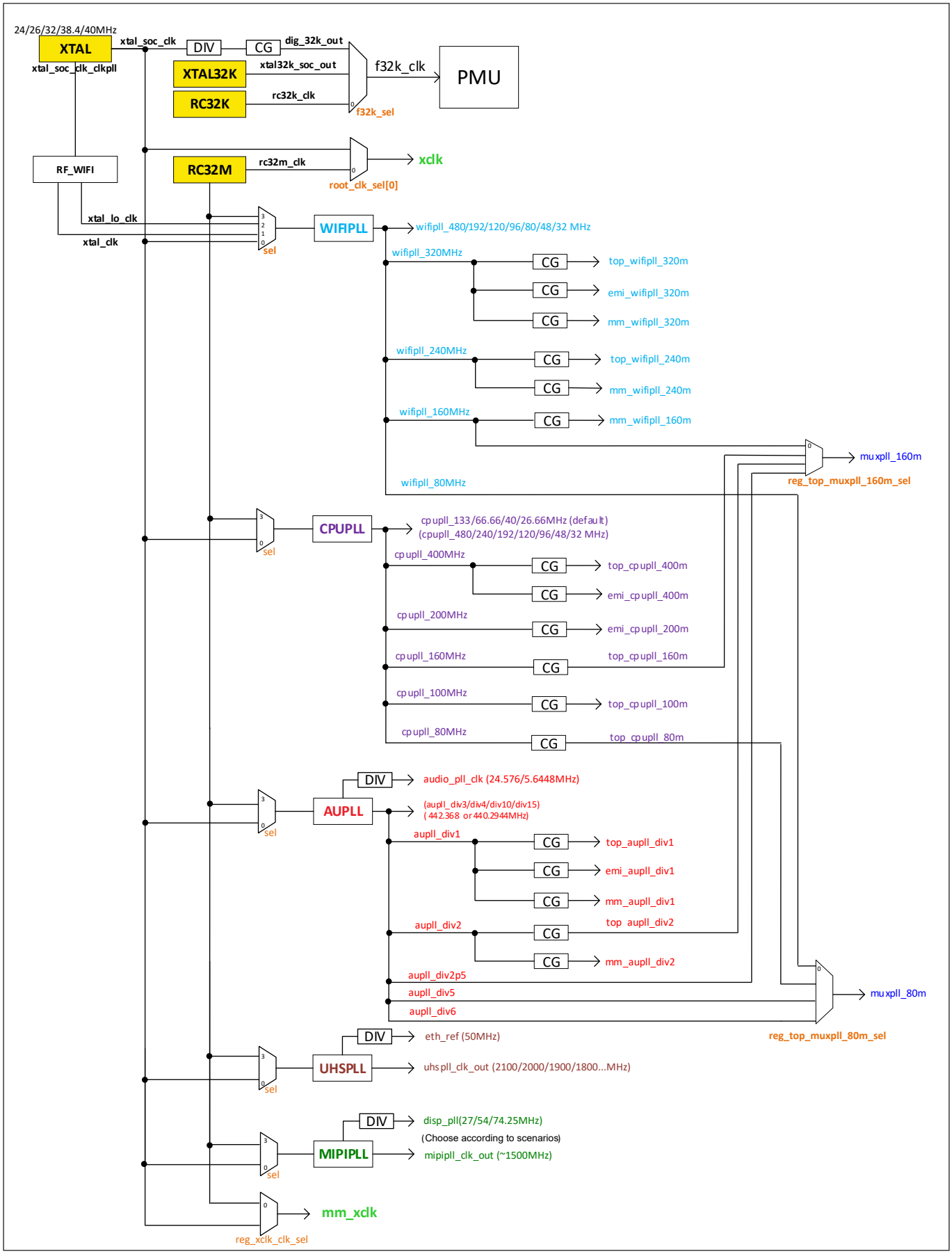


Fig. 2.1: System Clock Architecture

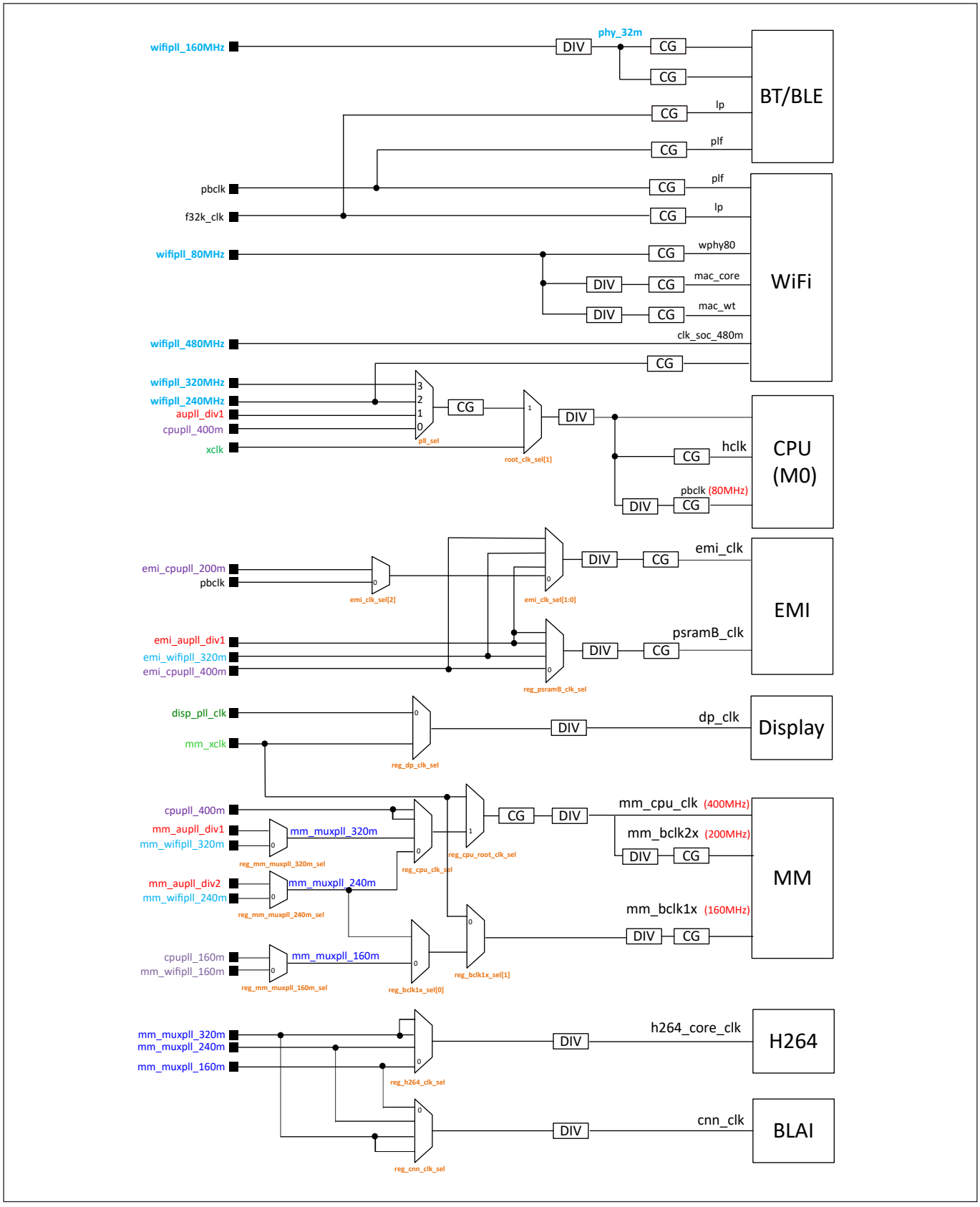


Fig. 2.2: Module Clock Architecture

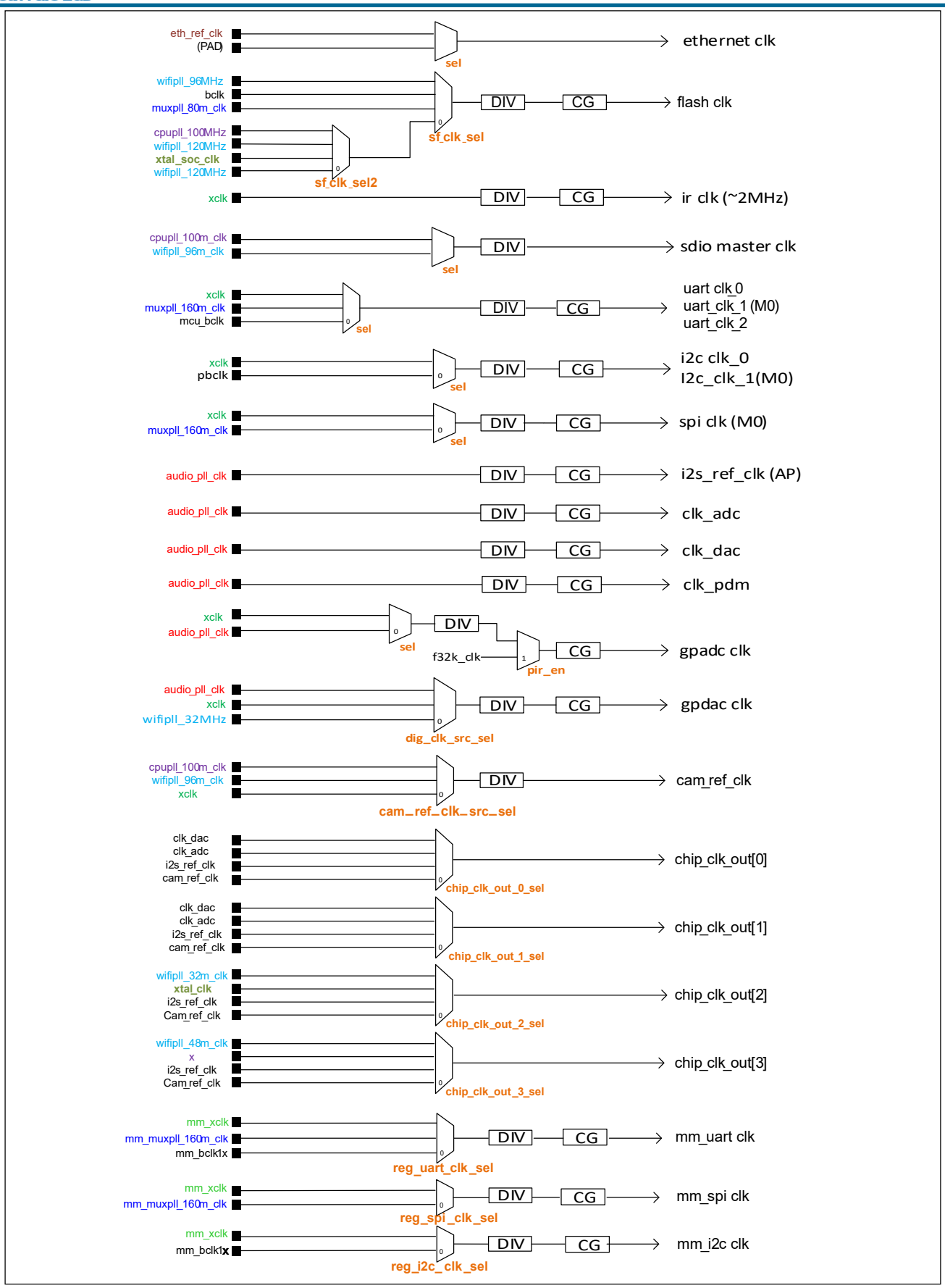


Fig. 2.3: Peripheral Clock Architecture

3 GLB

3.1 Overview

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.

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.

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 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

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 40 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 halfwords
- Data can be transferred through DMA from RAM to I/O pins for output

4.3 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>

4.4 GPIO Output Settings

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

4.4.1 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>

4.4.2 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.

4.4.3 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:

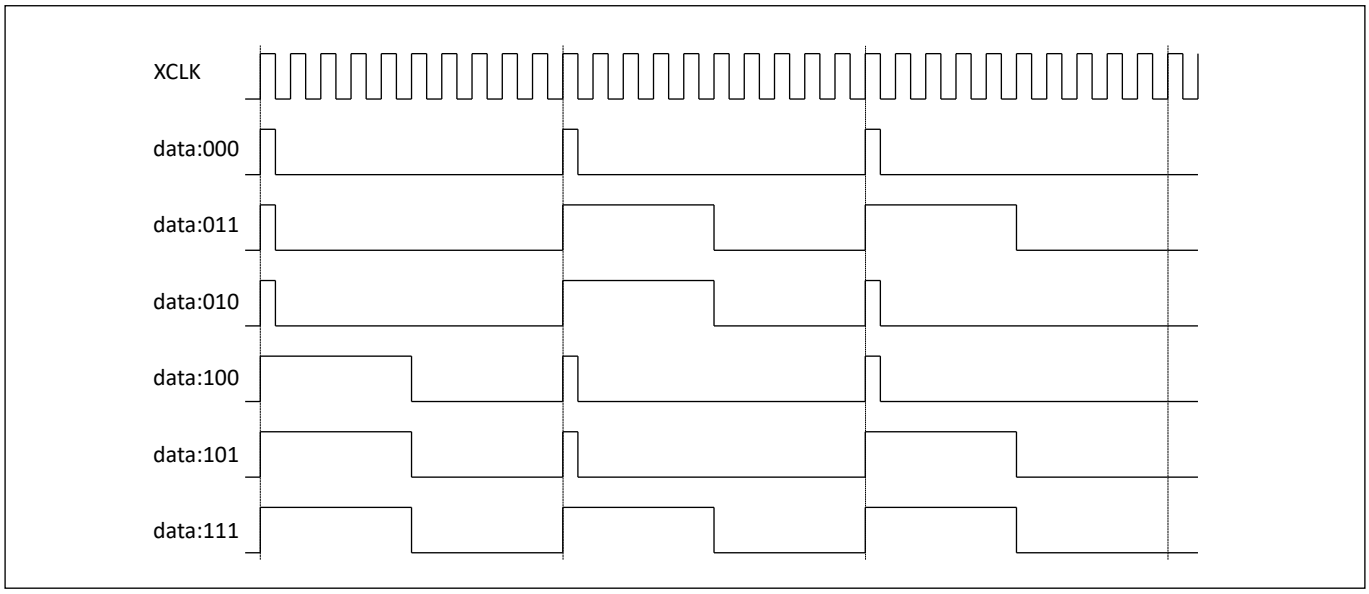


Fig. 4.1: General GPIO Output Waveform

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:

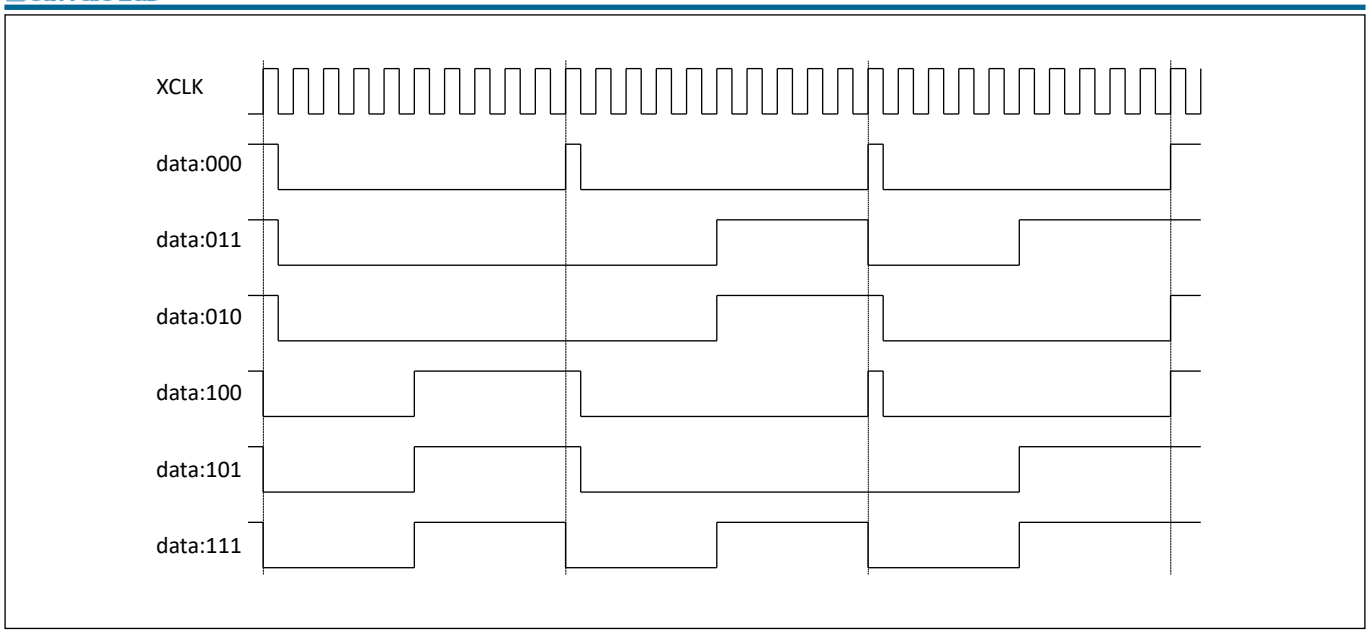


Fig. 4.2: Output Waveform of Inverted Level of Logical ‘1’ in Default High Level

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:

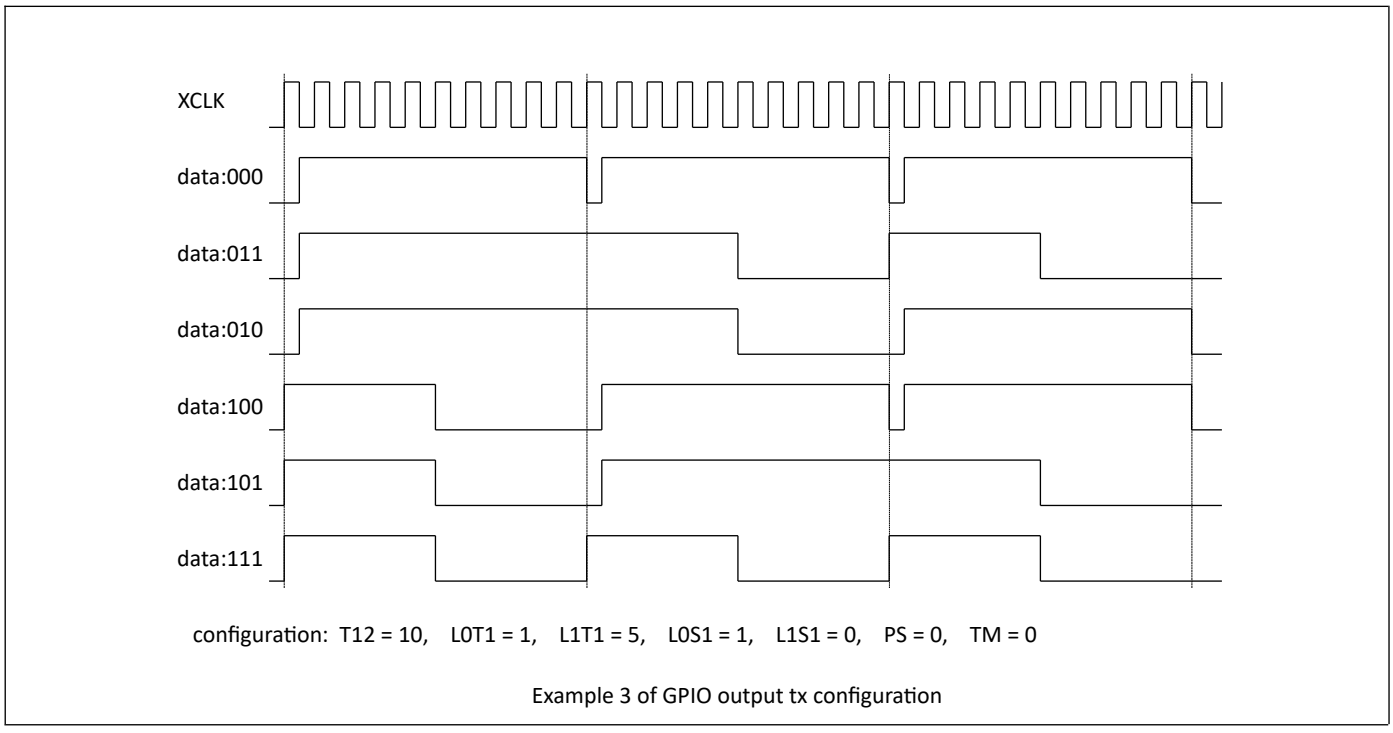


Fig. 4.3: Output Waveform of Inverted Level of Logical ‘0’ in Default Low Level

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:

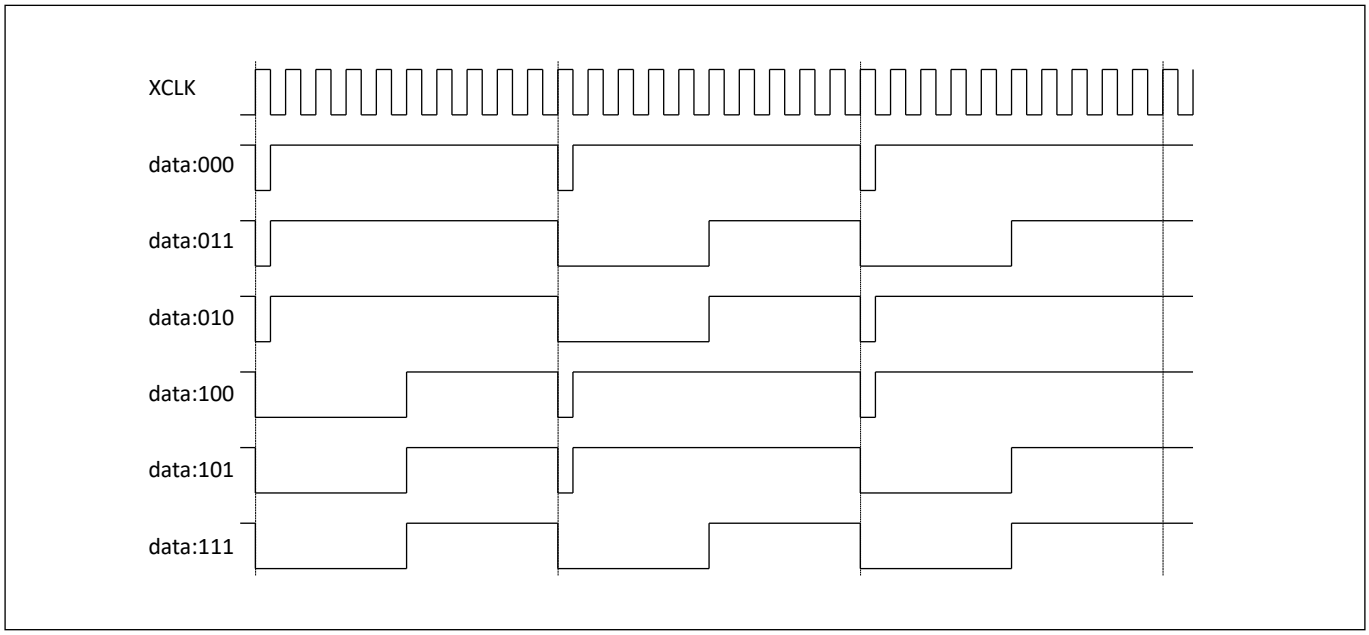


Fig. 4.4: Output Waveforms of Inverted Levels of Logical ‘0’ and Logical ‘1’ in Default High Level

4.4.4 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.

4.5 I/O FIFO

The depth of I/O FIFO is 128 halfwords. 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.

4.6 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>.

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
  - Optional 12-bit/14-bit/16-bit conversion result for output
  - Shortest ADC conversion time of 0.5 us (12-bit conversion result)
  - Optional reference voltage of 1.8V/3.3V
  - Transfer of conversion result to memory through DMA
  - Supports single channel conversion and multi-channel scanning modes
  - Single-ended and differential input modes
  - Jitter compensation
  - User-defined offset value of conversion result
  - Up to 1M for scanning mode clock and 2M for non-scanning mode clock
- 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.

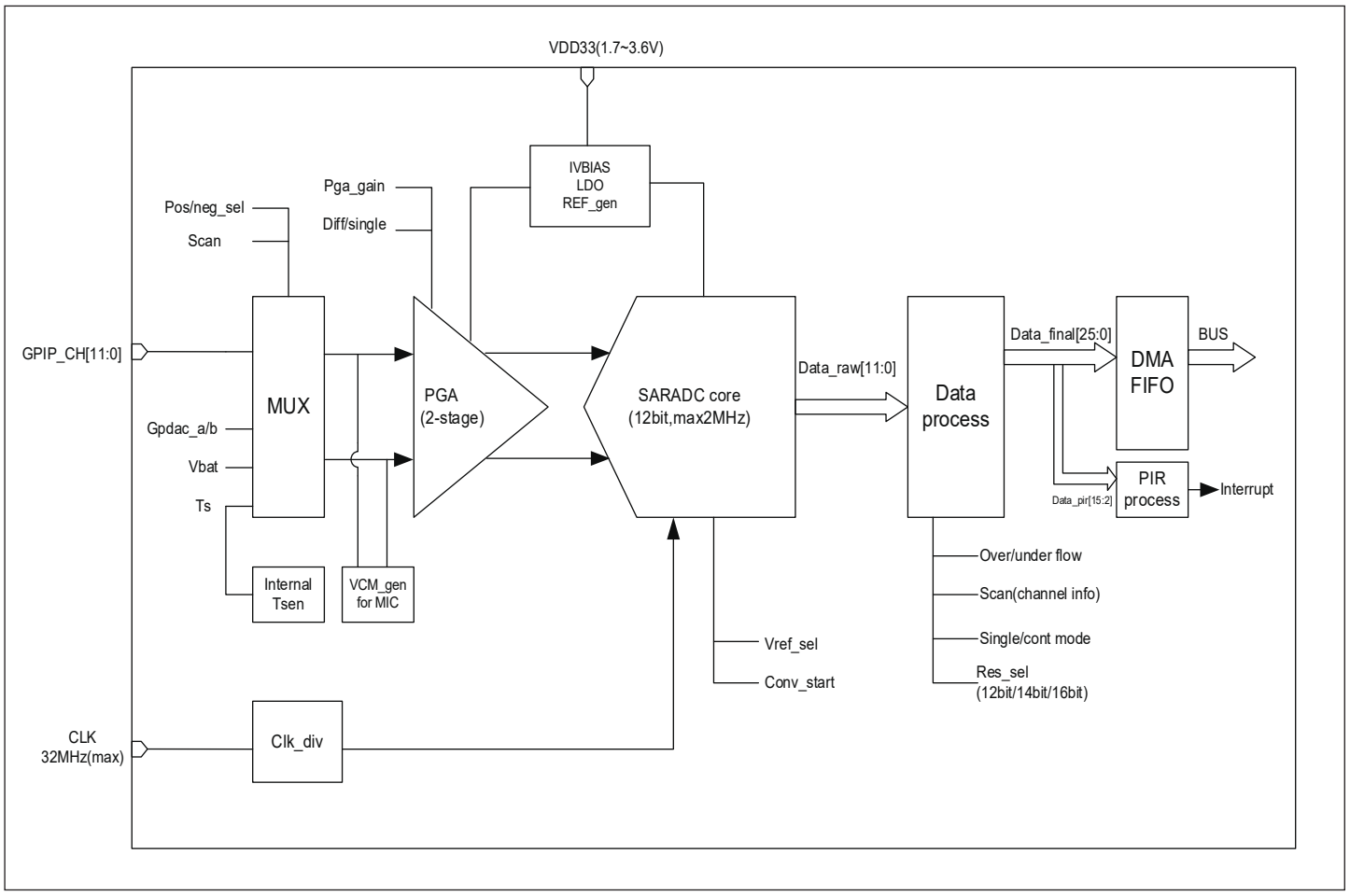


Fig. 5.1: Block Diagram of ADC

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 (12 bit) from analog signal to digital signal by successive comparison.

The data processing module further processes the conversion result, including adding channel information.

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.

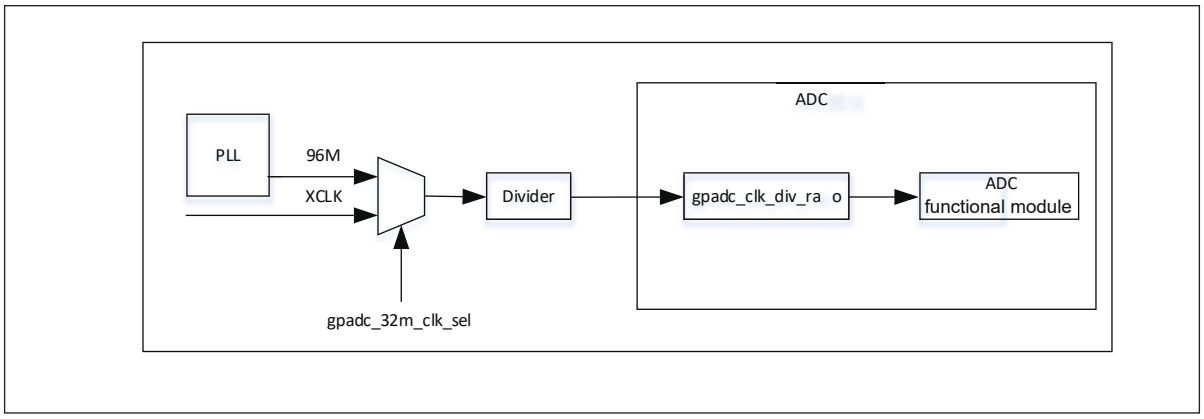


Fig. 5.2: ADC Clocks

The clock source of ADC can be the 96M from PLL, XTAL or internal RC32M. The GLB module controls the setting of clock source selection and provides clock frequency division. By default, the clock source of ADC is 96M and the clock frequency division is 2. The clock reaching the ADC module is 32M.

Inside the ADC module, clock frequency division (div=16 by default) is provided, so the default clock is 2M. Users can adjust the clock source and division factor of ADC according to the actual sampling needs.

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

ADC supports single channel conversion and scanning conversion modes.

- Single Channel Conversion Mode
  - Select the positive input channel by setting <gpadc_pos_sel> and the negative input channel by setting <gpadc_-neg_sel> of the register gpadc_reg_cmd
  - Set the <gpadc_cont_conv_en> control bit in the register gpadc_reg_config1 to 0, indicating single channel conversion,then set the gpadc_conv_start control bit to start conversion

In the scanning conversion mode, the gpadc_cont_conv_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.

The conversion result of ADC is usually pushed into FIFO. Users need to set the FIFO data receiving threshold interrupt according to the actual number of conversion channels, and this threshold interrupt is used as the ADC Conversion Complete Interrupt.

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 optional setting values are as follows:

- 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

The ADC module will generate interrupts upon positive or negative sampling over-range, and such interrupts can be masked by gpadc_pos_satur_mask and gpadc_neg_satur_mask. When interrupts are generated, the interrupt status can be queried through the registers gpadc_pos_satur and gpadc_neg_satur. Interrupts can be cleared through gpadc_pos_satur_clr and gpadc_neg_satur_clr. This function can be used to check whether 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 1.8 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 1.8 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

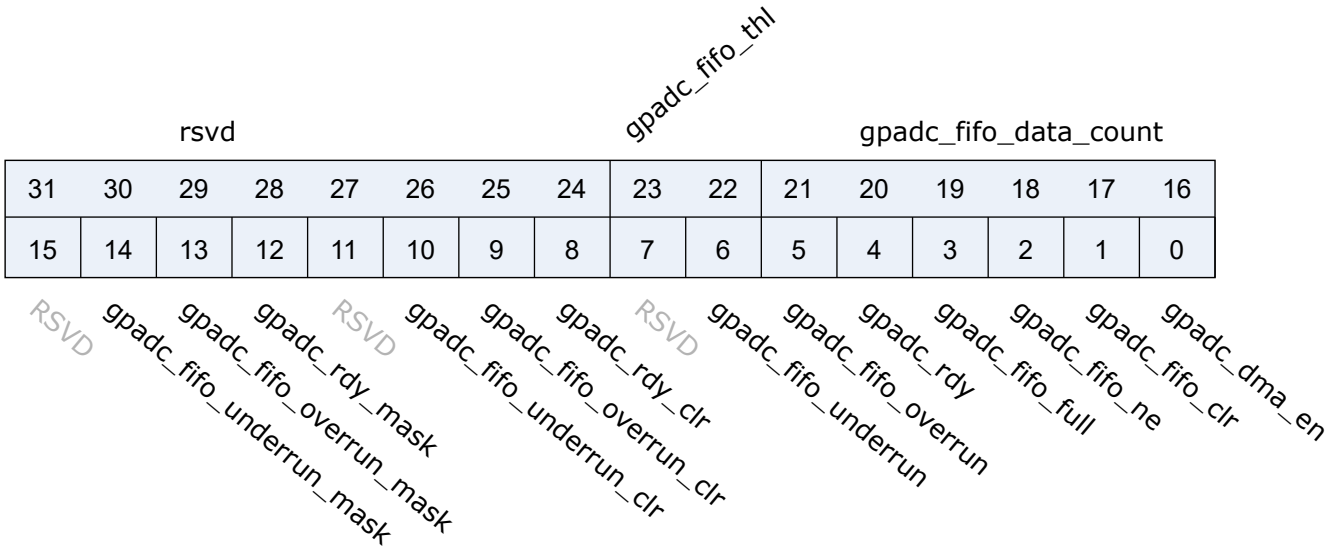
external diode. The negative input channel is set to GND. After finishing the above settings, set gpadc_tsvbe_low=0 to start measurement, and get the measurement result V0. Then set gpadc_tsvbe_low=1 to start measurement, and get the measurement result V1, Δ(ADC_out)=V1V0. The temperature T is calculated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

5.4 Register description

Name	Description
gpadc_config	
gpadc_dma_rdata	
gpadc_pir_train	

5.4.1 gpadc_config

Address：0x20002000



Bits	Name	Type	Reset	Description
31:24	rsvd	rsvd	8'd0	
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

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

5.4.2 gpadc_dma_rdata

Address：0x20002004

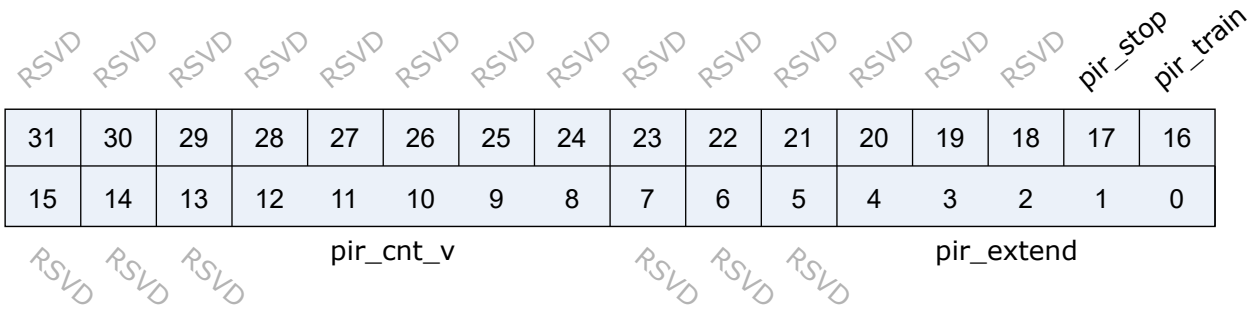
rsvd						gpadc_dma_rdata									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpadc_dma_rdata

Bits	Name	Type	Reset	Description
31:26	rsvd	rsvd	6'd0	
25:0	gpadc_dma_rdata	r	26'd0	GPADC finial conversion result stored in the FIFO

5.4.3 gpadc_pir_train

Address：0x20002020



Bits	Name	Type	Reset	Description
31:18	RSVD			
17	pir_stop	r	0	PIR Training End
16	pir_train	r/w	0	PIR Training Mode
15:13	RSVD			
12:8	pir_cnt_v	r	0	GPADC Record Extension Counter Value
7:5	RSVD			
4:0	pir_extend	r/w	5’d15	GPADC Record Extension after PIR interrupt

6 DAC

6.1 Overview

The chip integrates a 10bits digital-to-analog converter (DAC) with FIFO depth of 1, and supports two DAC modulation outputs. DAC can be used for playing audio and modulating transmitter voltage.

6.2 Features

- Modulation accuracy of 10bits
- Optional input clocks of 32k, 16k, 8k or 512k
- Allows DMA to transfer memory to DAC modulation registers
- Supports dual-channel playback and DMA transfer mode
- Output pins of DAC are fixed as GPIO4 for ChannelA and GPIO11 for ChannelB

6.3 Functional Description

The block diagram of DAC is shown as follows.

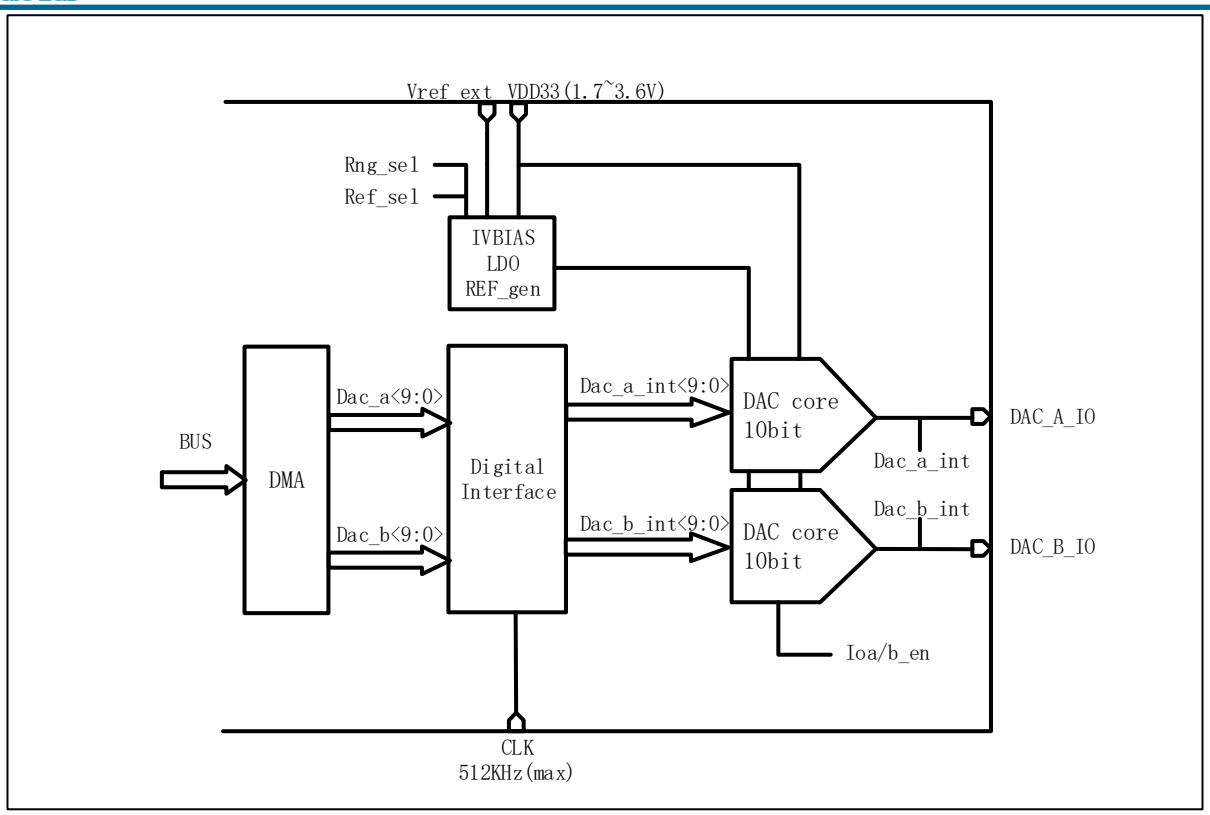


Fig. 6.1: Block Diagram of DAC

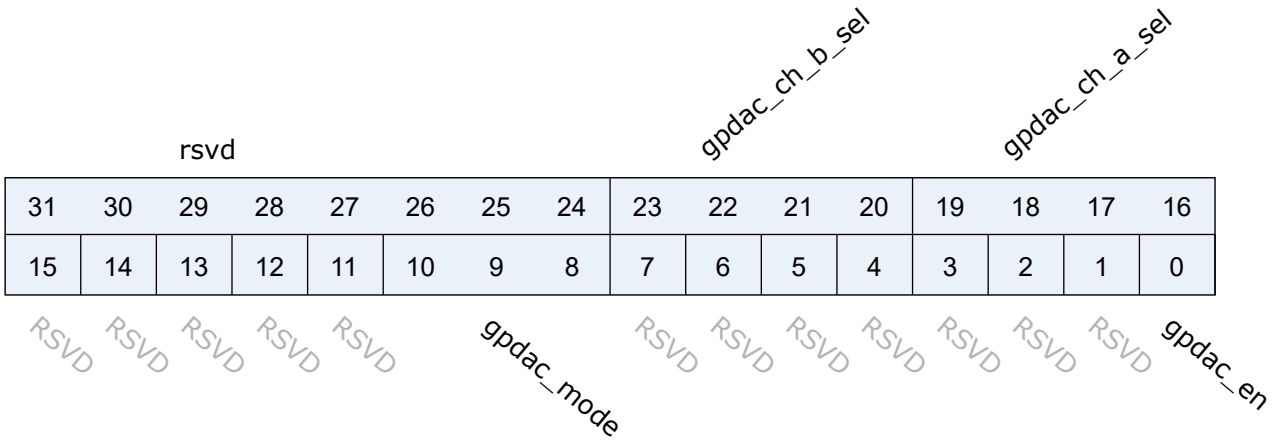
- Supports up to two modulated outputs
- Supports dual-channel playback and DMA data transfer
- Supports the DMA data interface with a length of 32bit, in which the high 16 bits will be modulated on the pin of ChannelA and the low 16 bits will be modulated on the pin of ChannelB

6.4 Register description

Name	Description
gpdac_config	
gpdac_dma_config	
gpdac_dma_wdata	
gpdac_tx_fifo_status	

6.4.1 gpdac_config

Address：0x20002040



Bits	Name	Type	Reset	Description
31:24	rsvd	rsvd	8'h0d	
23:20	gpdac_ch_b_sel	r/w	0	Channel B Source Select 0: Reg 1: DMA 2: DMA + Filter 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 2: DMA + Filter 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

6.4.2 gpdac_dma_config

Address：0x20002044

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	gpdac_dma_format				RSVD	RSVD	gpdac_dma_inv_msb	gpdac_dma_tx_en

Bits	Name	Type	Reset	Description
31:8	RSVD			
7:4	gpdac_dma_format	r/w	0	DMA TX format (Data 10-bit) 0: ([9:0]) A0, A1, A2… 1: ([25:16][9:0]) B0,A0, B1,A1, B2,A2… 2: ([25:16][9:0]) A1,A0, A3,A2, A5,A4… 4: ([15:6]) A0, A1, A2… 5: ([31:22][15:6]) B0,A0, B1,A1, B2,A2… 6: ([31:22][15:6]) 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

6.4.3 gpdac_dma_wdata

Address：0x20002048

gpdac_dma_wdata

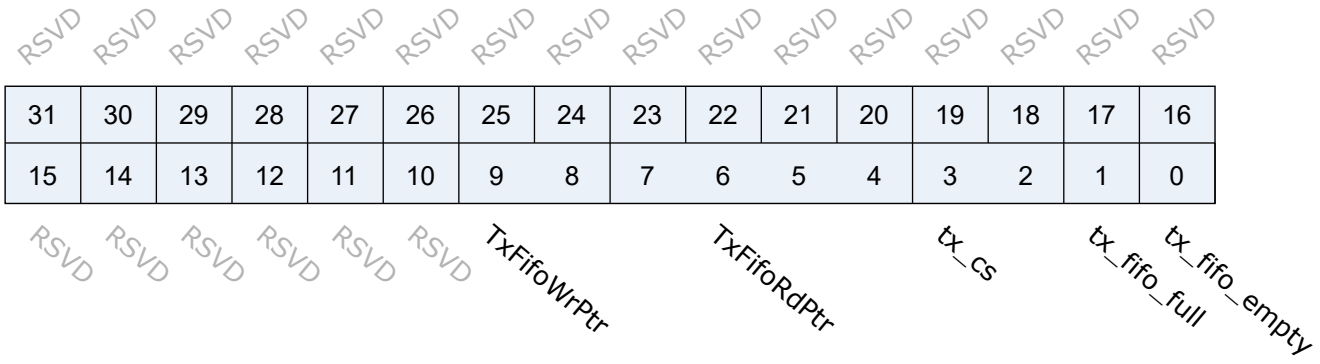
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

gpdac_dma_wdata

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

6.4.4 gpdac_tx_fifo_status

Address：0x2000204c



Bits	Name	Type	Reset	Description
31:10	RSVD			
9:8	TxFifoWrPtr	r	0	
7:4	TxFifoRdPtr	r	4'd8	
3:2	tx_cs	r	0	
1	tx_fifo_full	r	0	
0	tx_fifo_empty	r	0	

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 3 independent DMA controllers, of which DMA0 and DMA2 have 8 independent dedicated channels, and DMA1 has 4 such 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

- Three DMA controllers: DMA0, DMA1, and DMA2
- DMA0 and DMA2 have 8 independent dedicated channels, and DMA1 has 4 such 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
- Peripherals supported by DMA0 and DMA1: UART, I2C, SPI, ADC, IR, GPIO, Audio, I2S, and PDM
- Peripherals supported by DMA2: UART, I2C, SPI, DBI, and DSI
- Eight 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
  - Destination peripheral process control, source peripherals, destination peripherals
  - Destination peripheral process control, source memory, destination peripherals
  - Source peripheral process control, source peripherals, destination memory
  - Source peripheral 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.

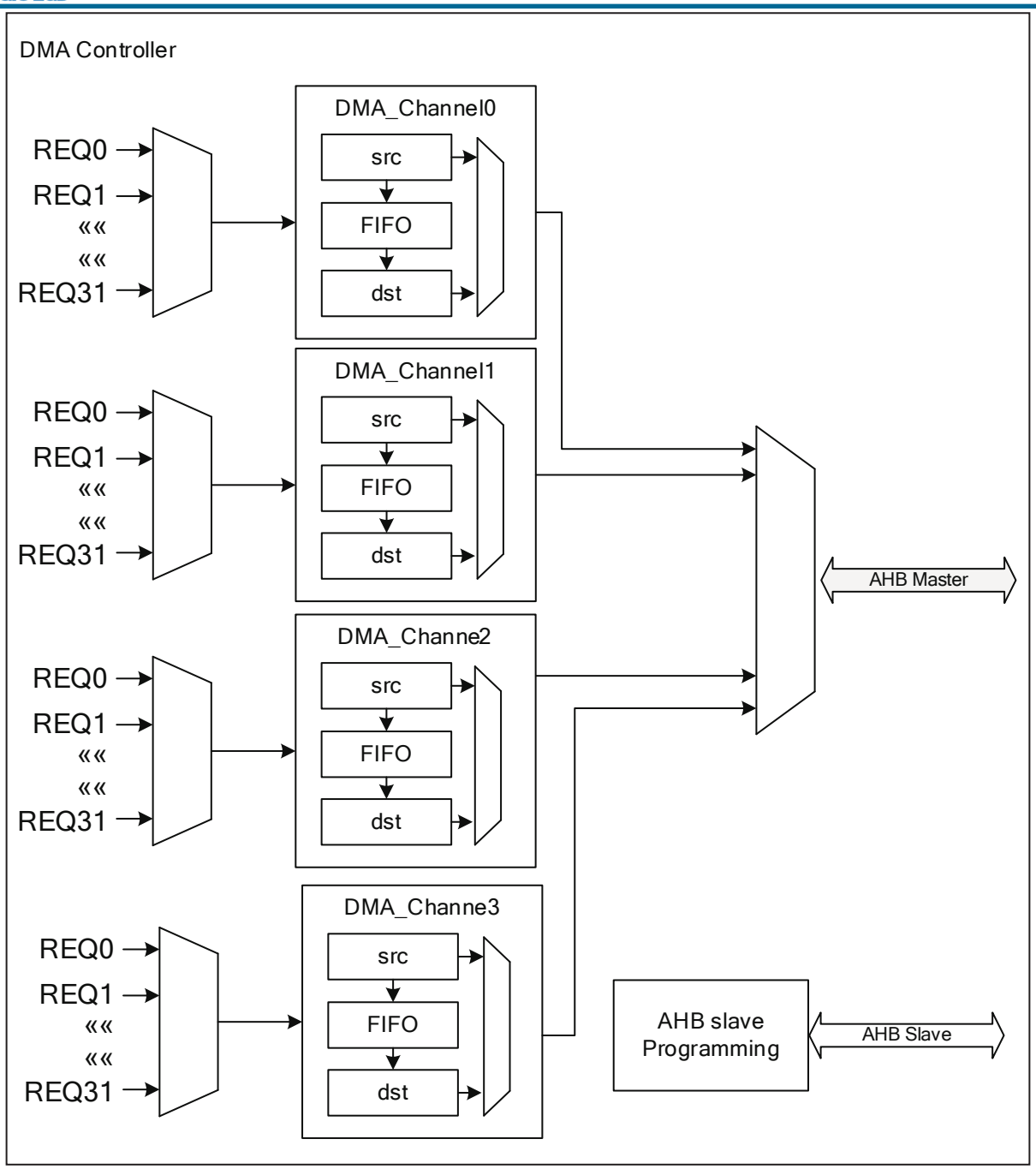


Fig. 7.1: Block Diagram of DMA

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

DMA0 and DMA2 support 8 channels, and DMA1 supports 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, four bytes, and eight bytes (for DMA2 only) 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 DMA0 and DMA1 must not exceed 16 bytes, and that of DMA2 must not exceed 128 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	DMA1	DMA2
0	UART0_RX	UART0_RX	UART3_RX
1	UART0_TX	UART0_TX	UART3_TX
2	UART1_RX	UART1_RX	SPI1_RX
3	UART1_TX	UART1_TX	SPI1_TX
4	UART2_RX	UART2_RX	-
5	UART2_TX	UART2_TX	-
6	I2C0_RX	I2C0_RX	I2C2_RX
7	I2C0_TX	I2C0_TX	I2C2_TX
8	IR_TX	IR_TX	I2C3_RX
9	GPIO_TX	GPIO_TX	I2C3_TX
10	SPI0_RX	SPI0_RX	DSI_RX
11	SPI0_TX	SPI0_TX	DSI_TX
12	AUDIO_RX	AUDIO_RX	-
13	AUDIO_TX	AUDIO_TX	-
14	I2C1_RX	I2C1_RX	-
15	I2C1_TX	I2C1_TX	-
16	I2S_RX	I2S_RX	-
17	I2S_TX	I2S_TX	-
18	PDM_RX	PDM_RX	-
22	GPADC_RX	GPADC_RX	DBI_TX
23	GPADC_TX	GPADC_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

ADC0/1 use DMA to transfer data

Configuration process:

1. Set the value of the SrcPeripheral bit in the register DMA_C0Config to 22/23. That is, set Source peripheral to GPADC0/GPADC1.

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.

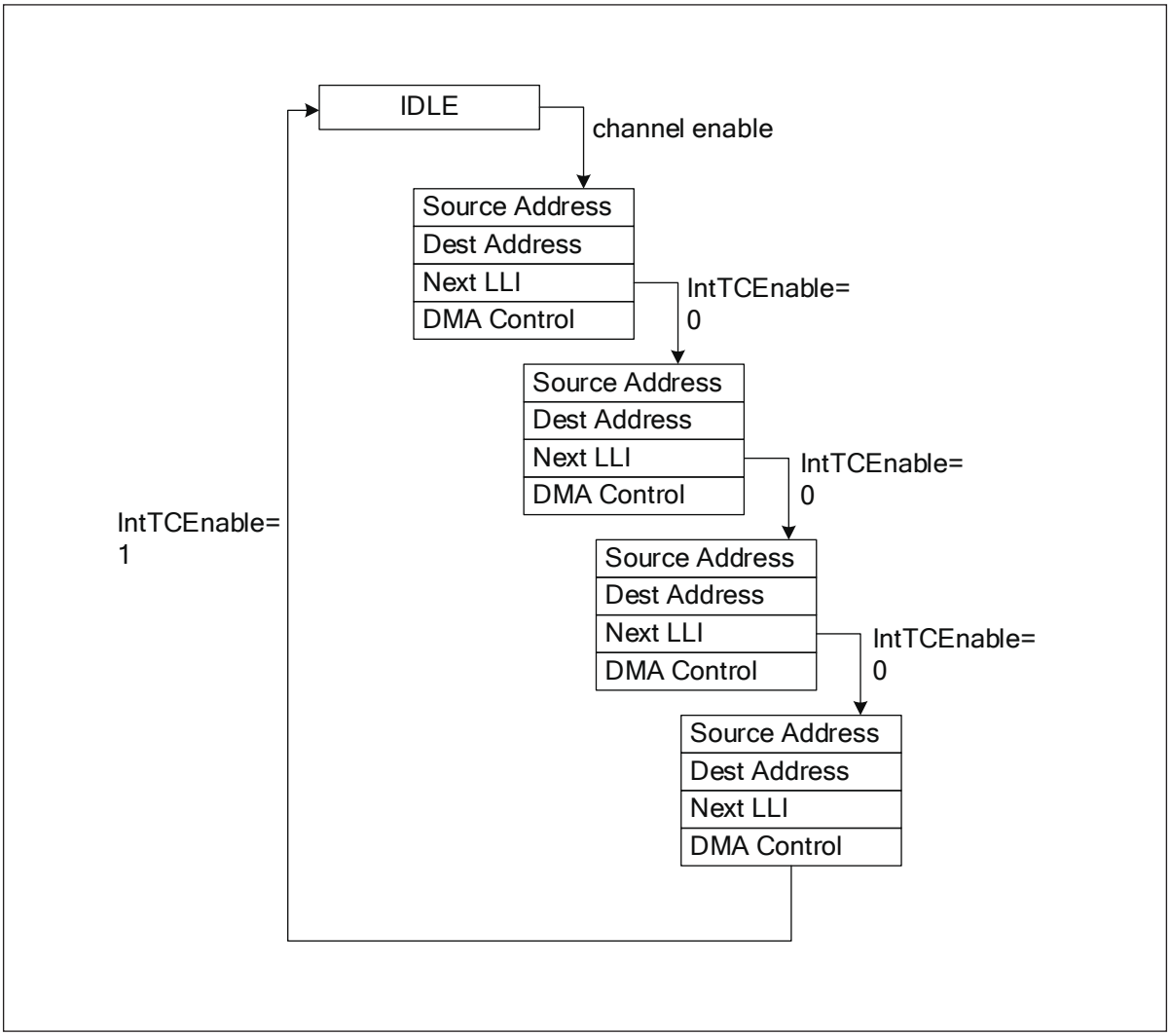


Fig. 7.3: LLI Framework

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: 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

7.4.2 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: 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

7.4.3 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	
DMA_IntTCStatus	
DMA_IntTCClear	
DMA_IntErrorStatus	
DMA_IntErrClr	
DMA_RawIntTCStatus	
DMA_RawIntErrorStatus	
DMA_EnbldChns	
DMA_SoftBReq	
DMA_SoftSReq	
DMA_SoftLBReq	
DMA_SoftLSReq	
DMA_Config	

Name	Description
DMA_Sync	
DMA_C0SrcAddr	
DMA_C0DstAddr	
DMA_C0LLI	
DMA_C0Control	
DMA_C0Config	
DMA_C0RSVD	
DMA_C1SrcAddr	
DMA_C1DstAddr	
DMA_C1LLI	
DMA_C1Control	
DMA_C1Config	
DMA_C1RSVD	
DMA_C2SrcAddr	
DMA_C2DstAddr	
DMA_C2LLI	
DMA_C2Control	
DMA_C2Config	
DMA_C2RSVD	
DMA_C3SrcAddr	
DMA_C3DstAddr	
DMA_C3LLI	
DMA_C3Control	
DMA_C3Config	
DMA_C3RSVD	
DMA_C4SrcAddr	
DMA_C4DstAddr	
DMA_C4LLI	
DMA_C4Control	

Name	Description
DMA_C4Config	
DMA_C4RSVD	
DMA_C5SrcAddr	
DMA_C5DstAddr	
DMA_C5LLI	
DMA_C5Control	
DMA_C5Config	
DMA_C5RSVD	
DMA_C6SrcAddr	
DMA_C6DstAddr	
DMA_C6LLI	
DMA_C6Control	
DMA_C6Config	
DMA_C6RSVD	
DMA_C7SrcAddr	
DMA_C7DstAddr	
DMA_C7LLI	
DMA_C7Control	
DMA_C7Config	
DMA_C7RSVD	

7.5.1 DMA_IntStatus

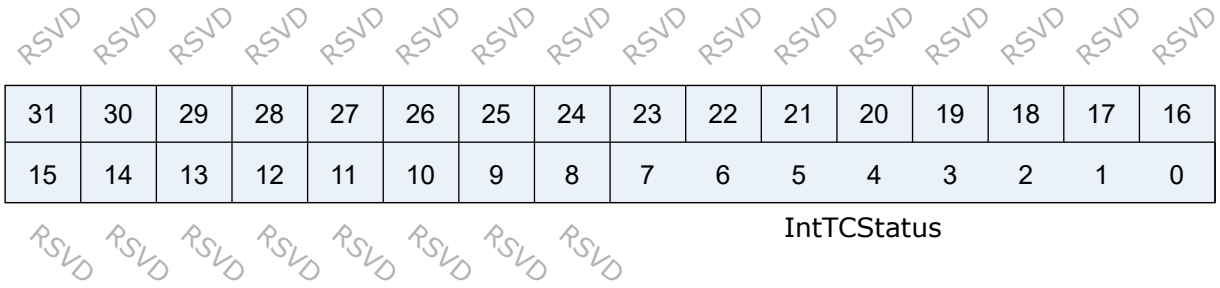
Address：0x2000c000

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	IntStatus							

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

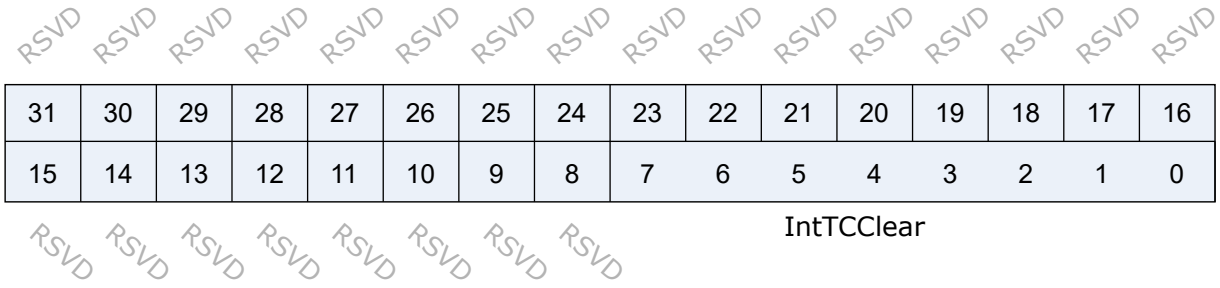
Address：0x2000c004



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

7.5.3 DMA_IntTCClear

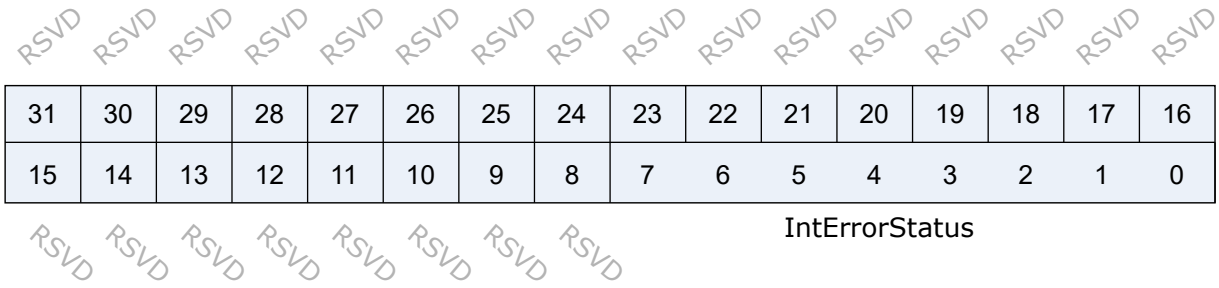
Address：0x2000c008



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

7.5.4 DMA_IntErrorStatus

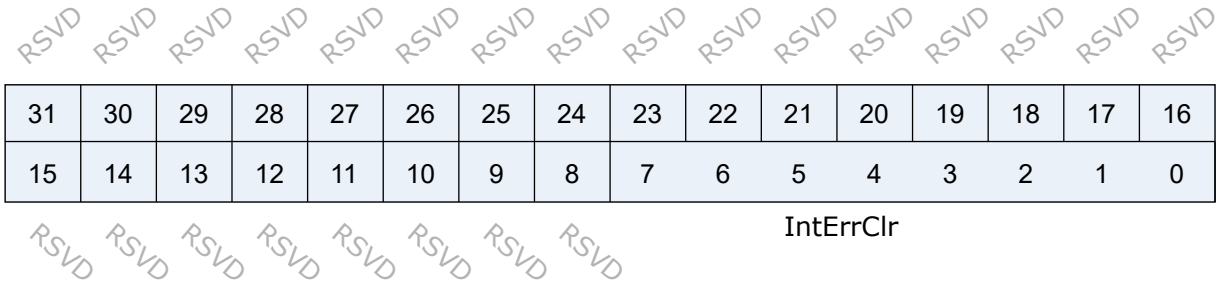
Address：0x2000c00c



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

7.5.5 DMA_IntErrClr

Address：0x2000c010



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

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 EnabledChannels

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

7.5.9 DMA_SoftBReq

Address：0x2000c020

SoftBReq

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

SoftBReq

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

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 M E

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

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

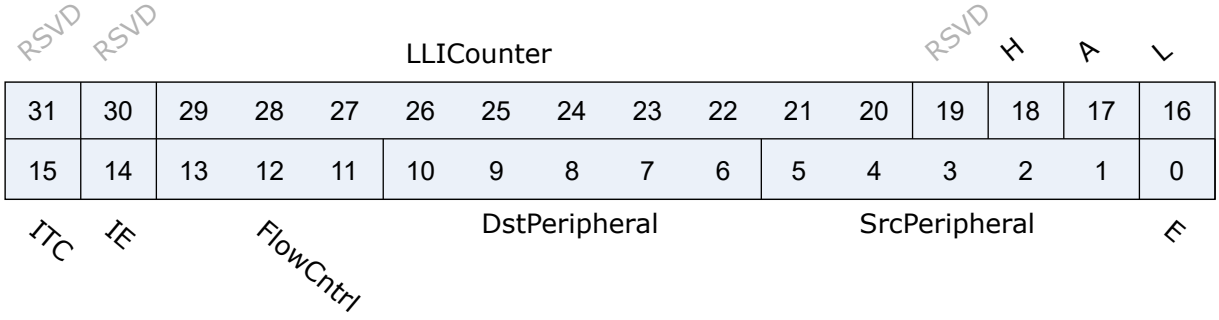
DBSize dst_min_mode SBSize

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

Bits	Name	Type	Reset	Description
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.

7.5.19 DMA_C0Config

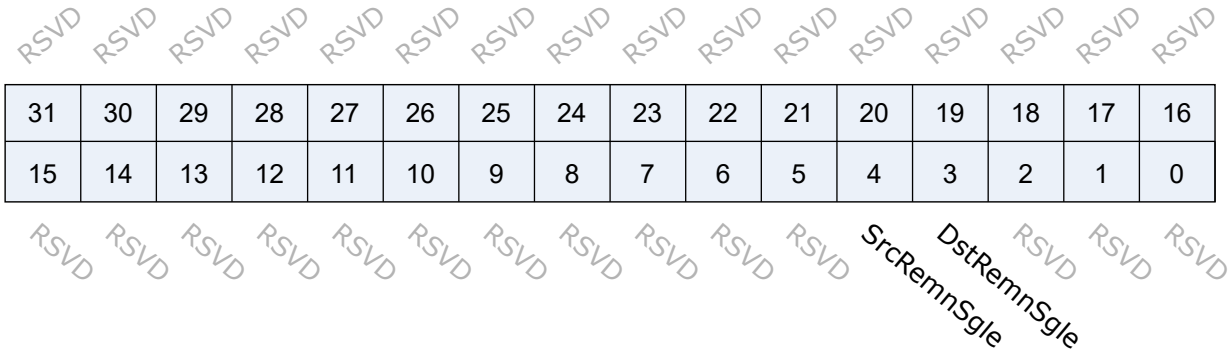
Address：0x2000c110



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.
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.

7.5.20 DMA_C0RSVD

Address：0x2000c11c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.21 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	

7.5.22 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	

7.5.23 DMA_C1LLI

Address：0x2000c208

LLI

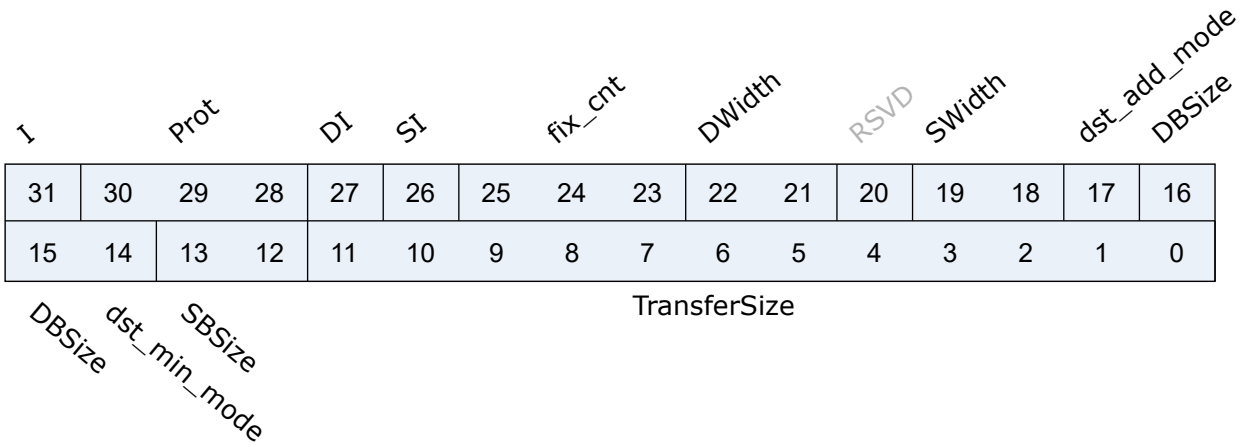
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.24 DMA_C1Control

Address：0x2000c20c

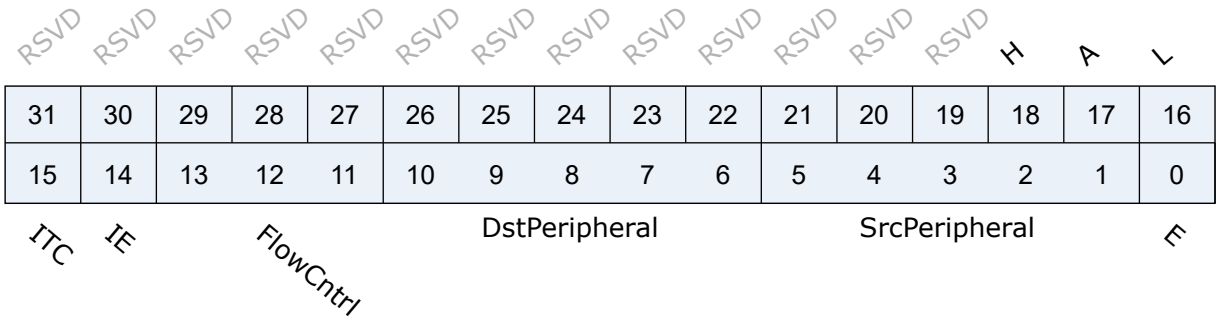


Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each 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

Bits	Name	Type	Reset	Description
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.

7.5.25 DMA_C1Config

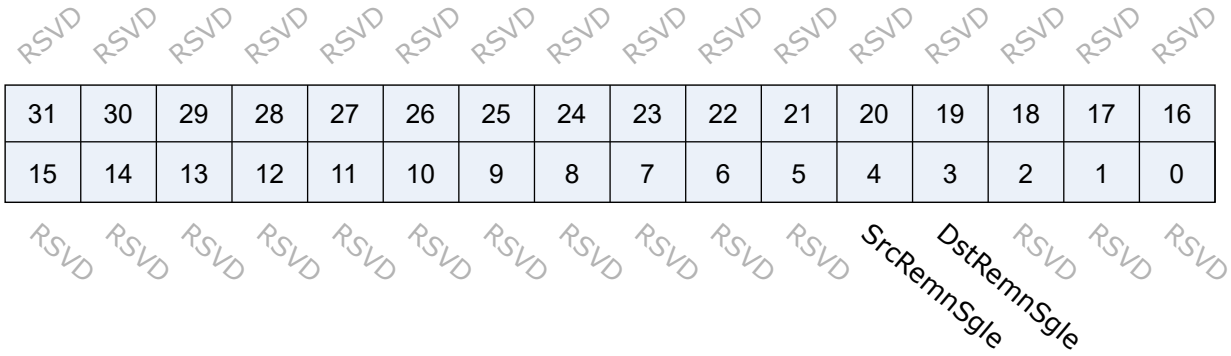
Address：0x2000c210



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.26 DMA_C1RSVD

Address：0x2000c21c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.27 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	

7.5.28 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	

7.5.29 DMA_C2LLI

Address：0x2000c308

LLI

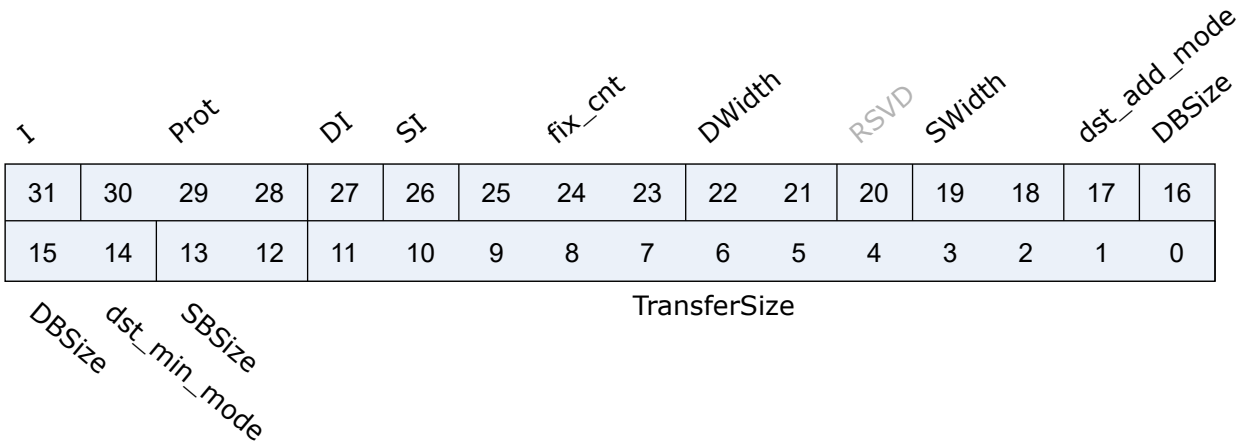
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.30 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			
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.

7.5.31 DMA_C2Config

Address：0x2000c310

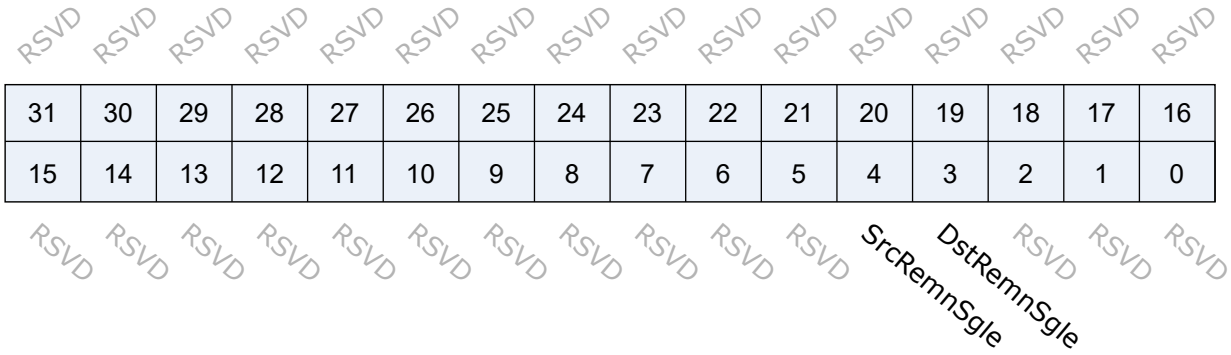
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	H	A	L
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITC	IE	FlowCntrl			DstPeripheral					SrcPeripheral					E

Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	

Bits	Name	Type	Reset	Description
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.32 DMA_C2RSVD

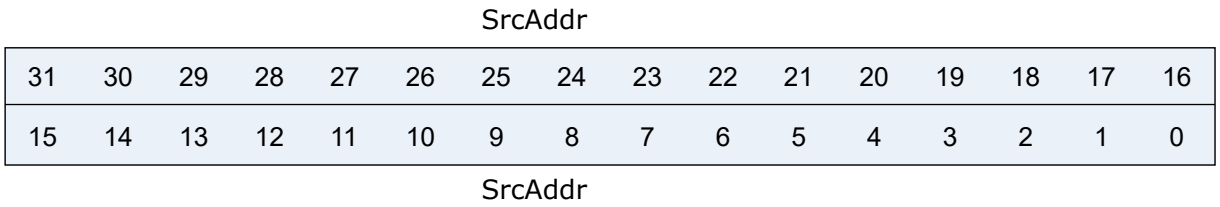
Address：0x2000c31c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.33 DMA_C3SrcAddr

Address：0x2000c400



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

7.5.34 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	

7.5.35 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	

7.5.36 DMA_C3Control

Address：0x2000c40c

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

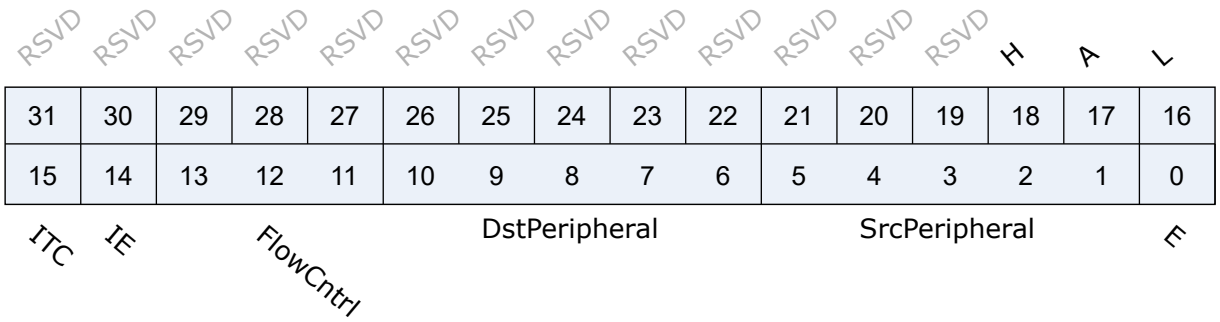
TransferSize

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.

7.5.37 DMA_C3Config

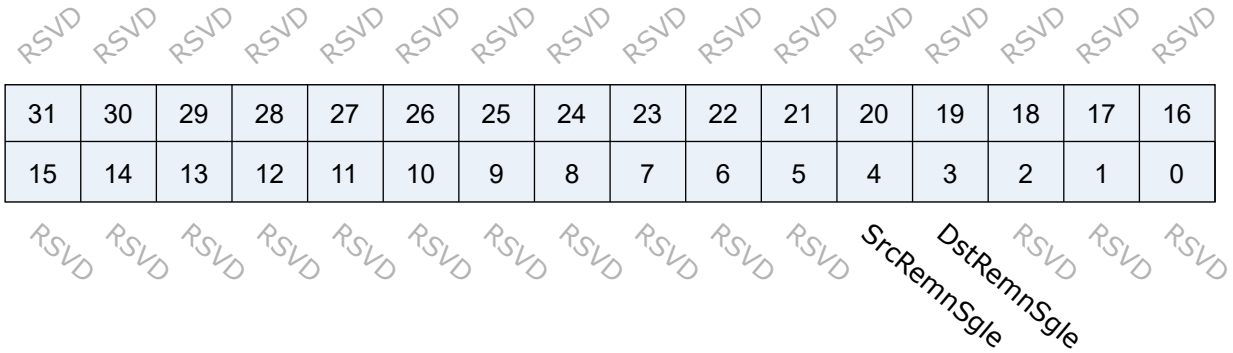
Address：0x2000c310



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.38 DMA_C3RSVD

Address：0x2000c31c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.39 DMA_C4SrcAddr

Address：0x2000c500

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	

7.5.40 DMA_C4DstAddr

Address：0x2000c504

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	

7.5.41 DMA_C4LLI

Address：0x2000c508

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	

7.5.42 DMA_C4Control

Address：0x2000c50c

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

Bits	Name	Type	Reset	Description
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.

7.5.43 DMA_C4Config

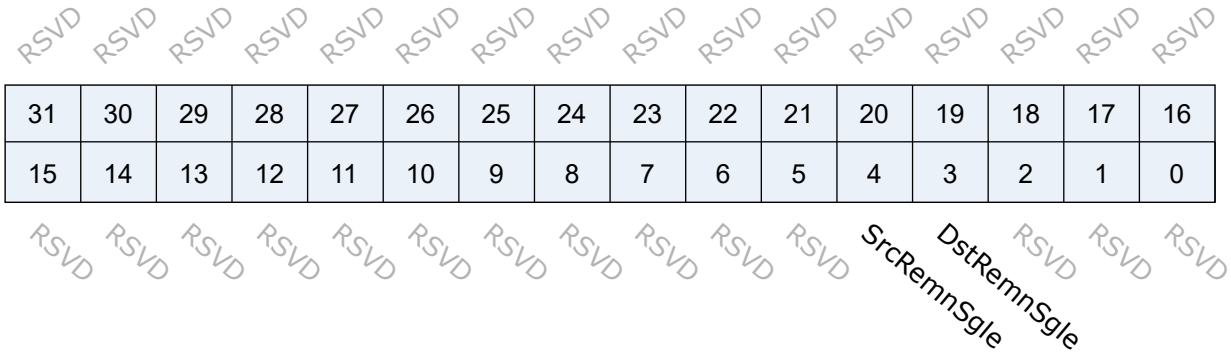
Address：0x2000c510

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	H	A	L
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
ITC	IE	FlowCntrl			DstPeripheral					SrcPeripheral					E

Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.44 DMA_C4RSVD

Address：0x2000c51c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.45 DMA_C5SrcAddr

Address：0x2000c600

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	

7.5.46 DMA_C5DstAddr

Address：0x2000c604

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	

7.5.47 DMA_C5LLI

Address：0x2000c608

LLI

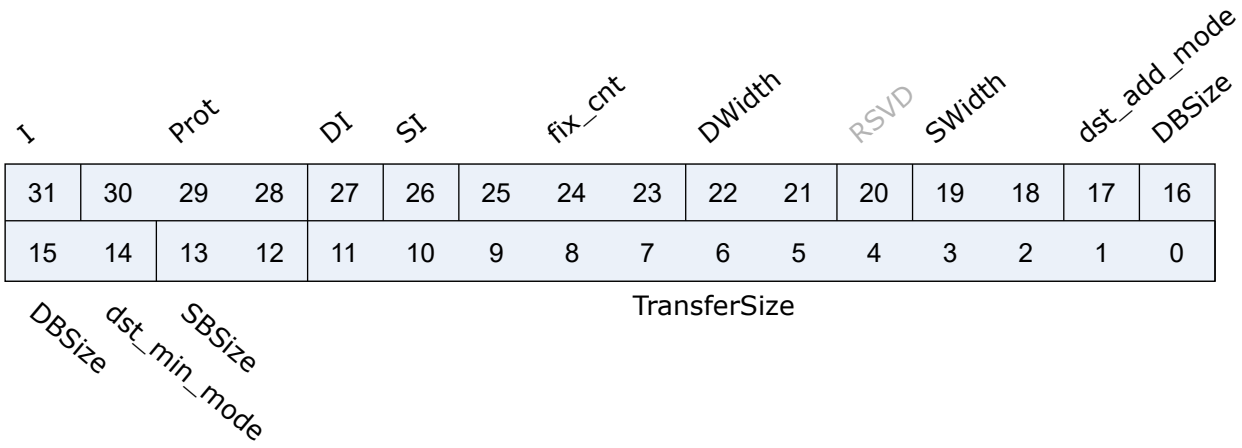
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.48 DMA_C5Control

Address：0x2000c60c



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.

7.5.49 DMA_C5Config

Address：0x2000c610

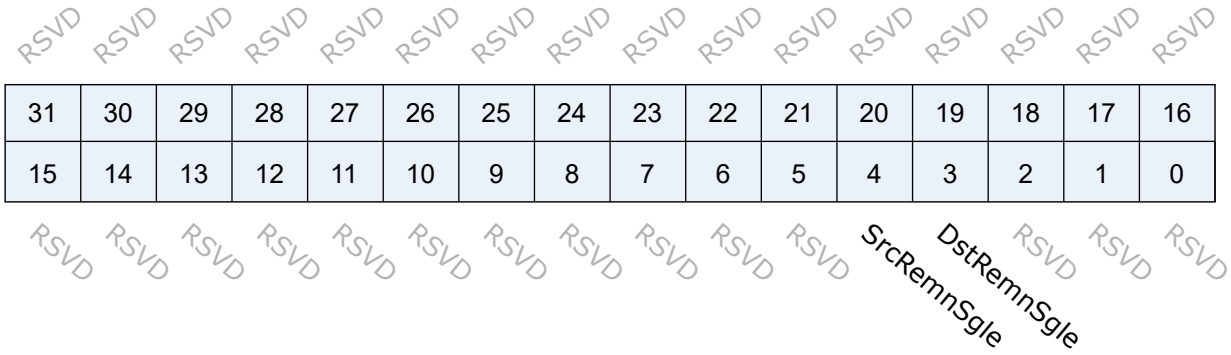
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	H	A	L
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITC	IE	FlowCntrl			DstPeripheral					SrcPeripheral					E

Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	

Bits	Name	Type	Reset	Description
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.50 DMA_C5RSVD

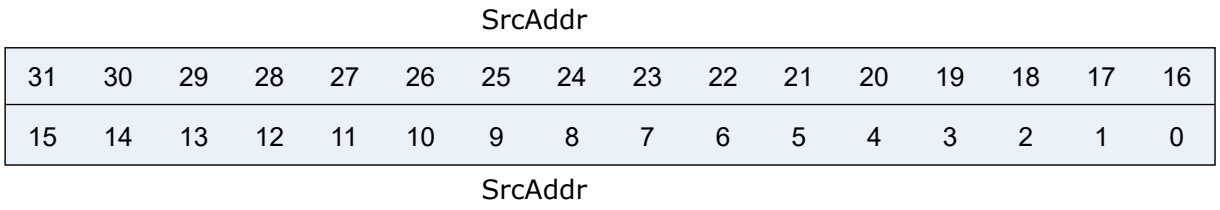
Address：0x2000c61c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.51 DMA_C6SrcAddr

Address：0x2000c700



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

7.5.52 DMA_C6DstAddr

Address：0x2000c704

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	

7.5.53 DMA_C6LLI

Address：0x2000c708

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	

7.5.54 DMA_C6Control

Address：0x2000c70c

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

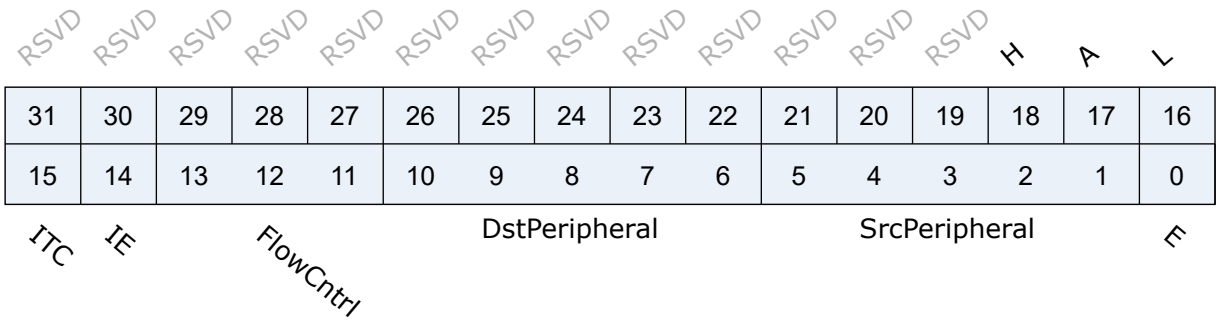
TransferSize

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.

7.5.55 DMA_C6Config

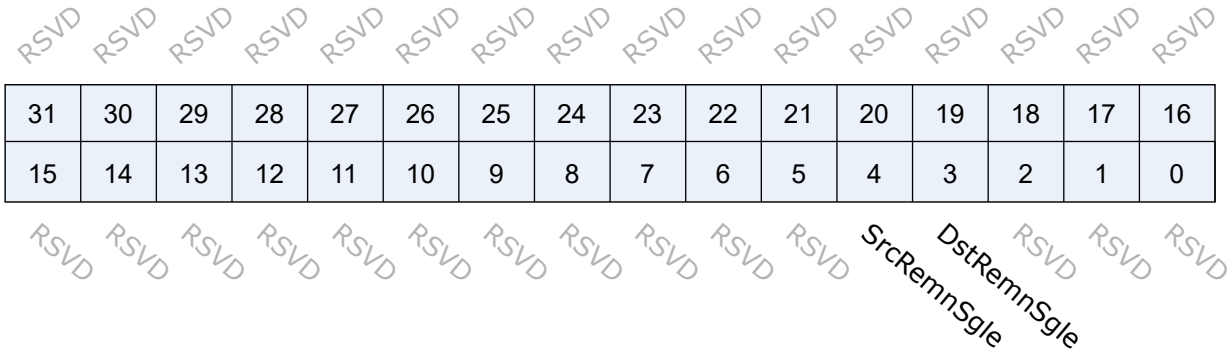
Address：0x2000c710



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.56 DMA_C6RSVD

Address：0x2000c71c



Bits	Name	Type	Reset	Description
31:5	RSVD			
4	SrcRemnSgle	r/w	0	Source remain single issue mode
3	DstRemnSgle	r/w	0	Destination remain single issue mode
2:0	RSVD			

7.5.57 DMA_C7SrcAddr

Address：0x2000c800

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	

7.5.58 DMA_C7DstAddr

Address：0x2000c804

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	

7.5.59 DMA_C7LLI

Address：0x2000c808

LLI

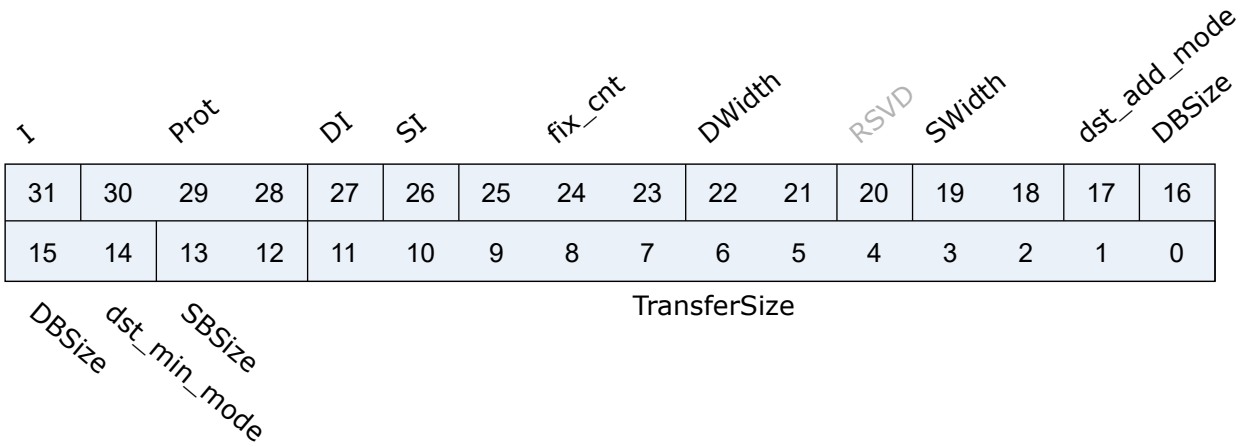
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

LLI

Bits	Name	Type	Reset	Description
31:0	LLI	r/w	0	

7.5.60 DMA_C7Control

Address：0x2000c80c

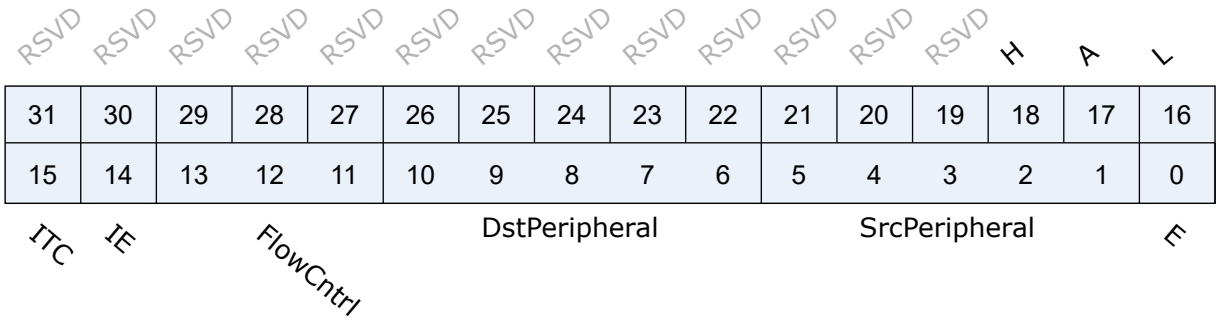


Bits	Name	Type	Reset	Description
31	I	r/w	0	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
30:28	Prot	r/w	0	No use for currently
27	DI	r/w	1	Destination increment. When set, the Destination address is incremented after each 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

Bits	Name	Type	Reset	Description
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.

7.5.61 DMA_C7Config

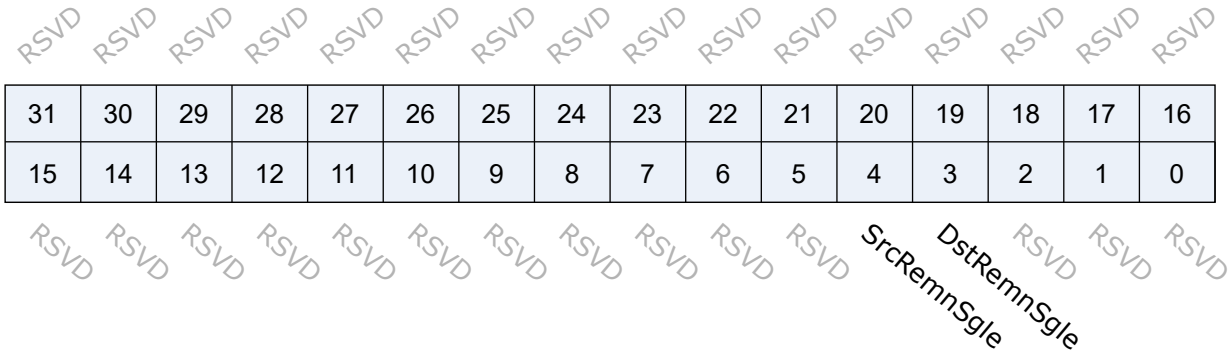
Address：0x2000c810



Bits	Name	Type	Reset	Description
31:19	RSVD			
18	H	r/w	0	
17	A	r	0	
16	L	r/w	0	
15	ITC	r/w	0	
14	IE	r/w	0	
13:11	FlowCntrl	r/w	0	
10:6	DstPeripheral	r/w	0	
5:1	SrcPeripheral	r/w	0	
0	E	r/w	0	

7.5.62 DMA_C7RSVD

Address：0x2000c81c



Bits	Name	Type	Reset	Description
31:5	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 DMA2D

8.1 Overview

The 2 Dimensional Direct Memory Access (DMA2D or 2D DMA) is a technique extended from 1D DMA that allows a 2D DMA controller to transfer data in the set strides (see the DMA section for details about 1D DMA). The memory address of data can be non-contiguous, incremented or decremented at a certain interval, or unchanged. The 2D DMA controller has 2 independent dedicated lanes, managing data transmission between peripherals and memory to enhance bus efficiency. It supports the transfer from memory to memory, memory to peripherals, and peripheral to memory, and provides the LLI linked list function.

8.2 Features

- Two dedicated independent lanes
- Configurable as 1D DMA or 2D DMA
- Independent control source and destination access width (single byte, double bytes, and four bytes)
- Each lane acts as a read-write cache independently
- Each lane can be triggered by independent peripheral hardware or software
- Supported peripherals: UART, I2C, SPI, DBI, and DSI
- Eight 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
  - Destination peripheral process control, source peripherals, destination peripherals

  - Destination peripheral process control, source memory, destination peripherals
  - Source peripheral process control, source peripherals, destination memory
  - Source peripheral process control, source peripherals, destination peripherals
- Supports the LLI linked list function to improve DMA efficiency, and configures whether to trigger interrupt for each node separately
- Exclusive features of 2D DMA:
  - Supports translation of 8/16/32-bit images in any direction
  - Supports rotation of 8/16/32-bit images at 90°/180°/270°
  - Supports horizontal/vertical folding of 8/16/32-bit images
  - Supports filling by 8/16/32-bit images
  - Supports 8/16/24/32-bit Color Key

8.3 Functional Description

8.3.1 Operating Principle

On the basis of 1D DMA, 2D DMA adds the SRC XINCR, SRC XCOUNT, SRC YINCR, and SRC YCOUNT; and adds the DEST XINCR, DEST XCOUNT, and DEST YINCR. DEST YCOUNT will not be configured, as it will be derived from other values. Here the stride value can be negative. The 2D DMA transfer can be viewed as a nested loop, where the outer loop is specified by YCOUNT and each time the loop address increments by YINCR bytes (decrements when YINCR is negative); the inner loop is specified by XCOUNT and each time the loop address increments XINCR bytes (decrements when XINCR is negative). The working principle is illustrated as follows:

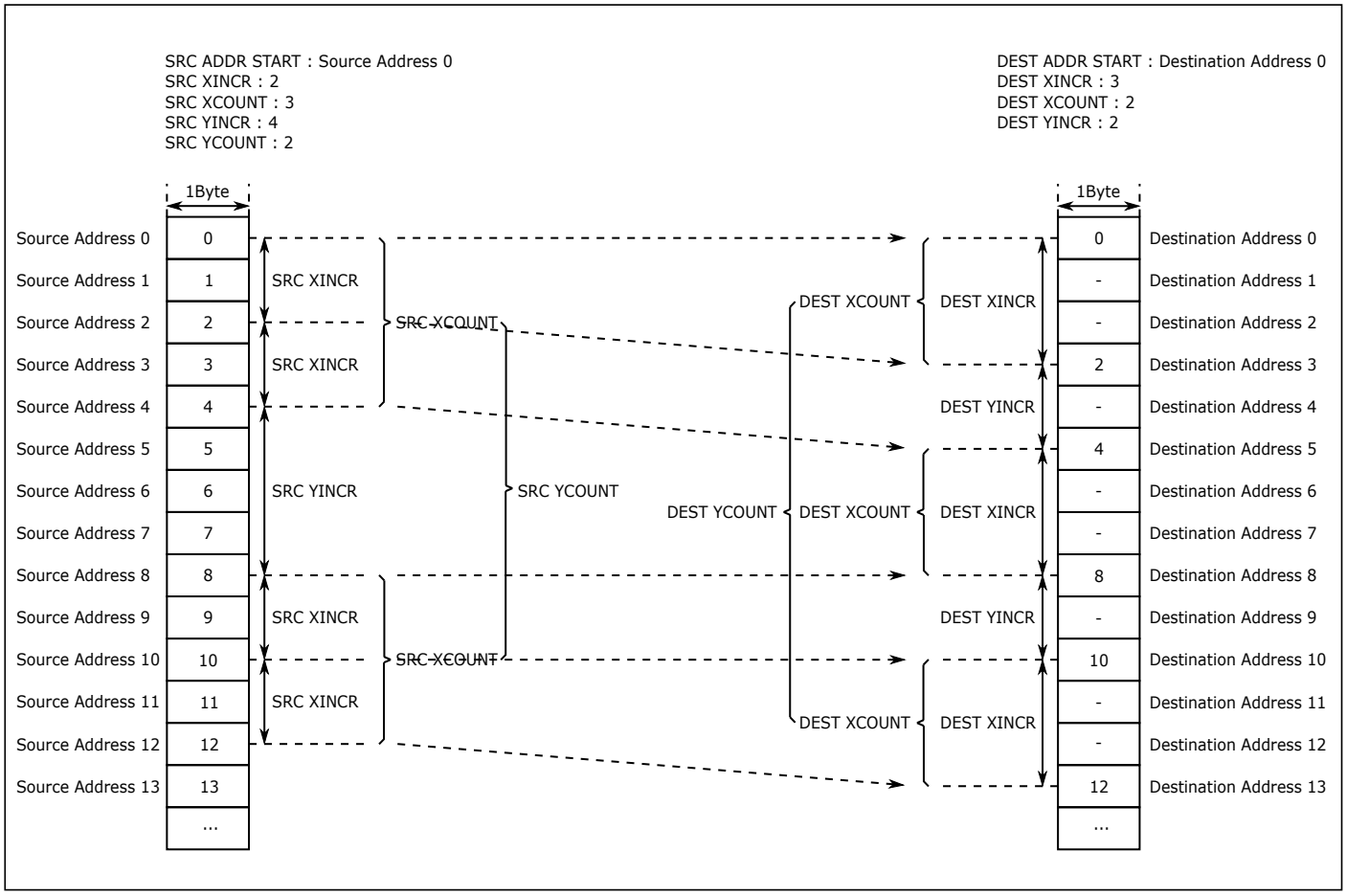


Fig. 8.1: Operating Principle

8.3.2 Image Translation

If the memory data is considered as a rectangular image, the image can be translated by setting the stride and loop values, as shown below:

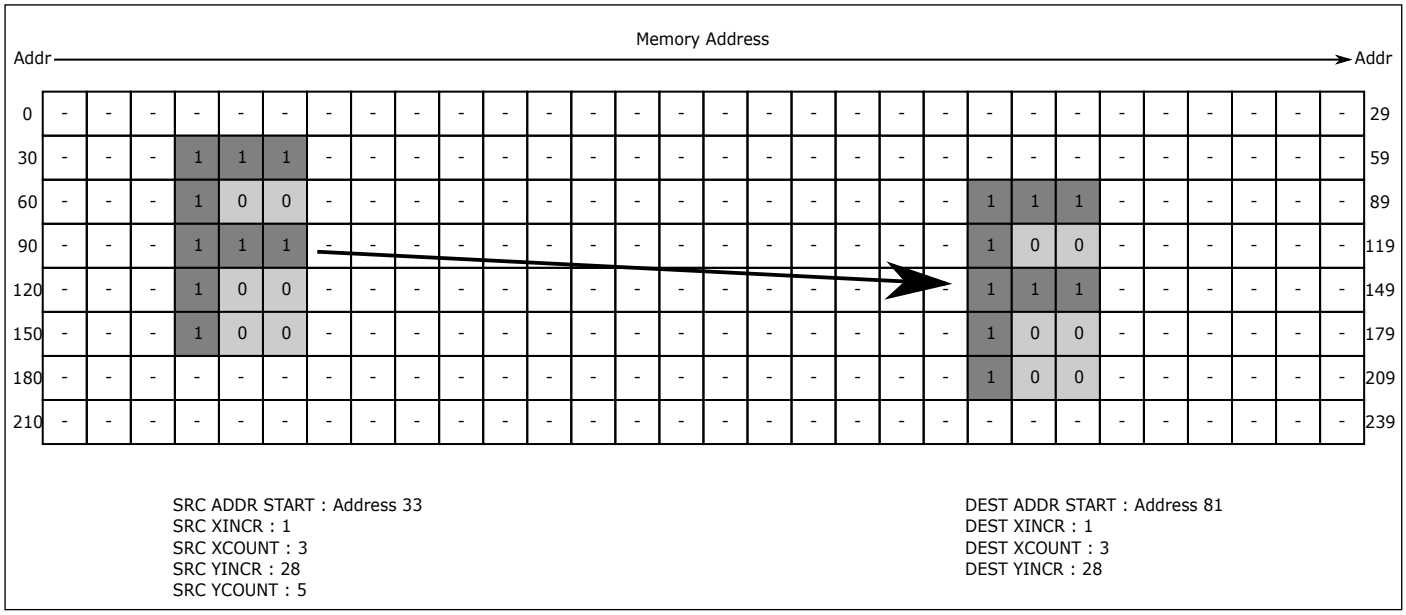


Fig. 8.2: Image Translation

As shown in the figure, it is assumed that a pixel width (P) is 8 bit, and the transfer width of 2D DMA is set to 8 bit. In an M × N image (30 × 8 in the figure), a rectangular area of W × H (3 × 5 in the figure) is translated from the coordinate (x1, y1) [(3, 1) in the figure] to the coordinate (x2, y2) [(21, 2) in the figure]. The start address of image data starts from addr0 (0 in the figure). Relevant parameters of 2D DMA are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = P = 1
- SRC XCOUNT = W = 3
- SRC YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28
- SRC YCOUNT = H = 5
- DEST ADDR START = addr0+(M*y2+x2)*P = 0+(30*2+21)*1 = 81
- DEST XINCR = P = 1
- DEST XCOUNT = W = 3
- DEST YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28

8.3.3 Image Rotation

If the memory data is considered as a rectangular image, the image can be rotated at 90°/180°/270° by setting the stride and loop values, as shown below:

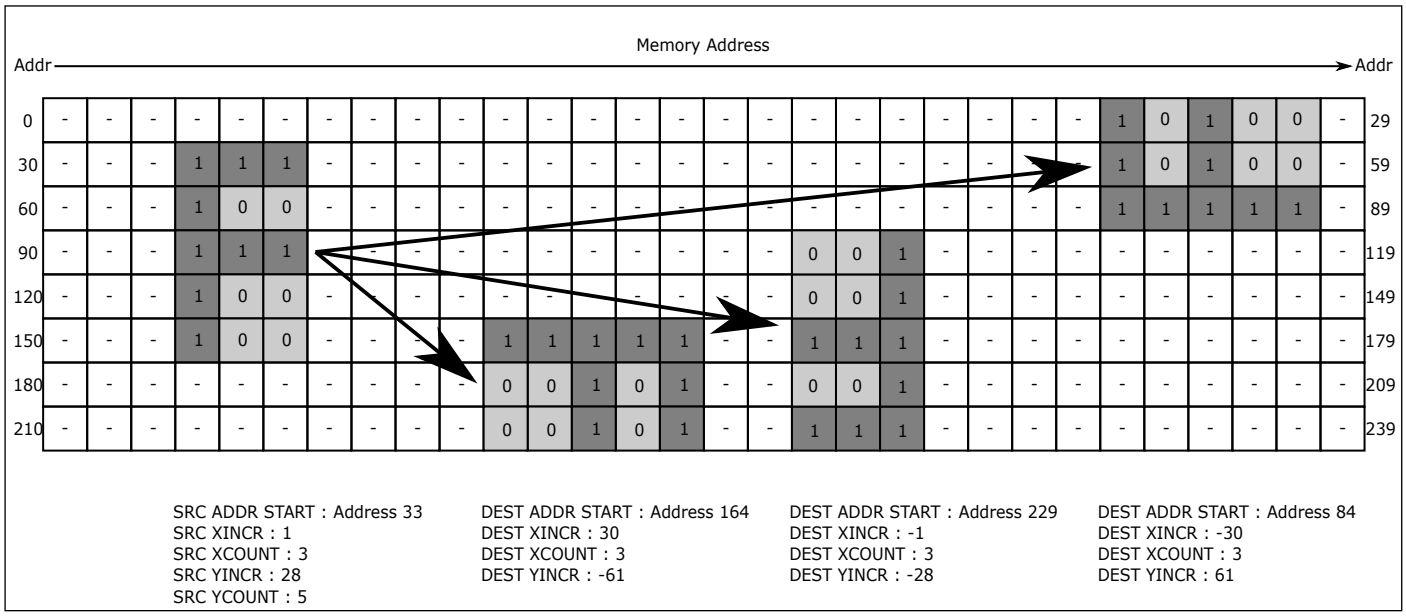


Fig. 8.3: Image Rotation

As shown in the figure, it is assumed that a pixel width (P) is 8 bit, and the transfer width of 2D DMA is set to 8 bit. In an M × N image (30 × 8 in the figure), a rectangular area of W × H (3 × 5 in the figure) is rotated clockwise at 90° from the coordinate (x1, y1) [(3, 1) in the figure] to the coordinate (x2, y2) [(14,5) in the figure]. The start address of image data starts from addr0 (0 in the figure). Relevant parameters of 2D DMA are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = P = 1
- SRC XCOUNT = W = 3
- SRC YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28
- SRC YCOUNT = H = 5
- DEST ADDR START = addr0+(M*y2+x2)*P = 0+(30*5+14)*1 = 164
- DEST XINCR = M*P = 30*1 = 30
- DEST XCOUNT = W = 3
- DEST YINCR = -(M*(W-1)+1)*P = -(30*(3-1)+1)*1 = -61

For the clockwise rotation of the rectangular area at 180° from the coordinate (x1, y1) [(3, 1) in the figure] to the coordinate (x3, y3) [(19, 7) in the figure], relevant parameters are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = P = 1
- SRC XCOUNT = W = 3
- SRC YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28
- SRC YCOUNT = H = 5
- DEST ADDR START = addr0+(M*y3+x3)*P = 0+(30*7+19)*1 = 229
- DEST XINCR = -P = -1
- DEST XCOUNT = W = 3
- DEST YINCR = -(M-(W-1))*P = -(30-(3-1))*1 = -28

For the clockwise rotation of the rectangular area at 270° from the coordinate (x1, y1) [(3, 1) in the figure] to the coordinate (x4, y4) [(24, 2) in the figure], relevant parameters are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = P = 1
- SRC XCOUNT = W = 3
- SRC YINCR = (M(W1))*P = (30(31))*1 = 28
- SRC YCOUNT = H = 5
- DEST ADDR START = addr0+(M*y4+x4)*P = 0+(30*2+24)*1 = 84
- DEST XINCR = -M*P = -30*1 = -30
- DEST XCOUNT = W = 3
- DEST YINCR = (M*(W-1)+1)*P = (30*(3-1)+1)*1 = 61

8.3.4 Image Folding

If the memory data is considered as a rectangular image, the image can be folded horizontally or vertically by setting the stride and loop values, as shown below:

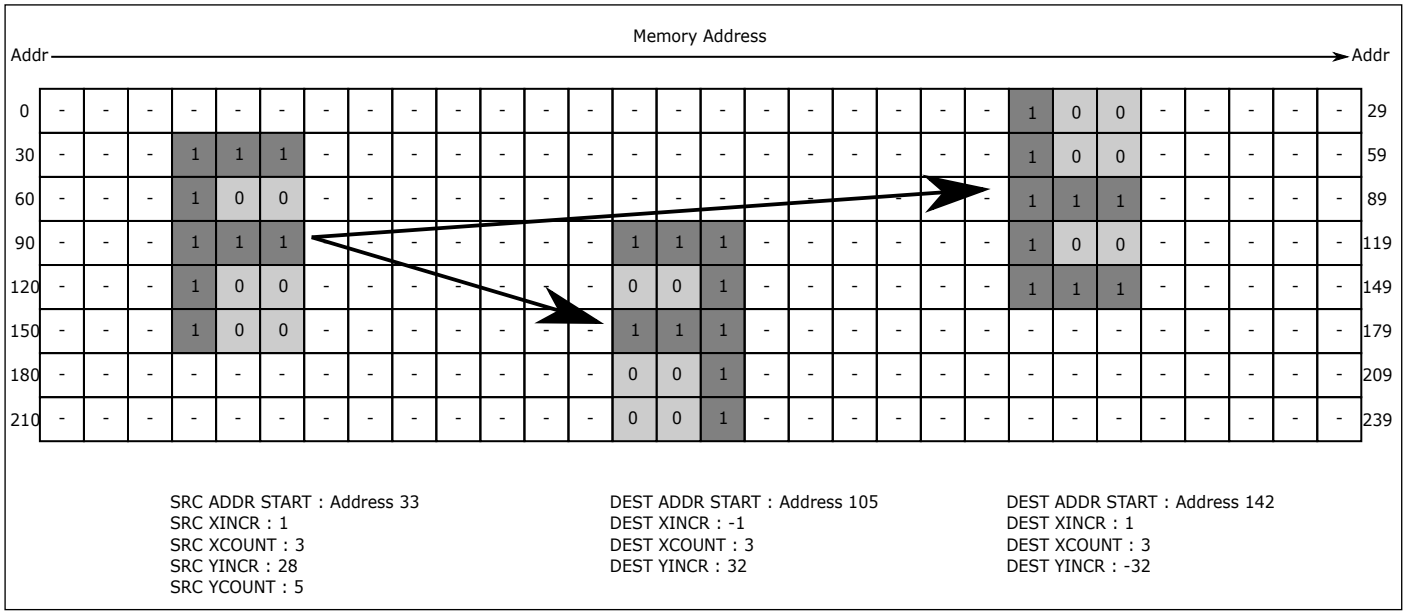


Fig. 8.4: Image Folding

As shown in the figure, it is assumed that a pixel width (P) is 8 bit, and the transfer width of 2D DMA is set to 8 bit. In an M × N image (30 × 8 in the figure), a rectangular area of W × H (3 × 5 in the figure) is folded horizontally from the coordinate (x1, y1) [(3, 1) in the figure] to the coordinate (x2, y2) [(15, 3) in the figure]. The start address of image data starts from addr0 (0 in the figure). Relevant parameters of 2D DMA are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = P = 1
- SRC XCOUNT = W = 3
- SRC YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28
- SRC YCOUNT = H = 5
- DEST ADDR START = addr0+(M*y2+x2)*P = 0+(30*3+15)*1 = 105
- DEST XINCR = -P = -1
- DEST XCOUNT = W = 3
- DEST YINCR = (M+(W-1))*P = (30+(3-1))*1 = 32

For vertical folding of the rectangular area from the coordinate (x1, y1) [(3, 1) in the figure] to the coordinate (x3, y3) [(22, 4) in the figure], relevant parameters are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = P = 1
- SRC XCOUNT = W = 3

- SRC YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28
- SRC YCOUNT = H = 5
- DEST ADDR START = addr0+(M*y3+x3)*P = 0+(30*4+22)*1 = 142
- DEST XINCR = P = 1
- DEST XCOUNT = W = 3
- DEST YINCR = -(M+(W-1))*P = -(30+(3-1))*1 = -32

8.3.5 Image Filling

If the memory data is considered as a rectangular image, the image can be filled by setting the stride and loop values, as shown below:

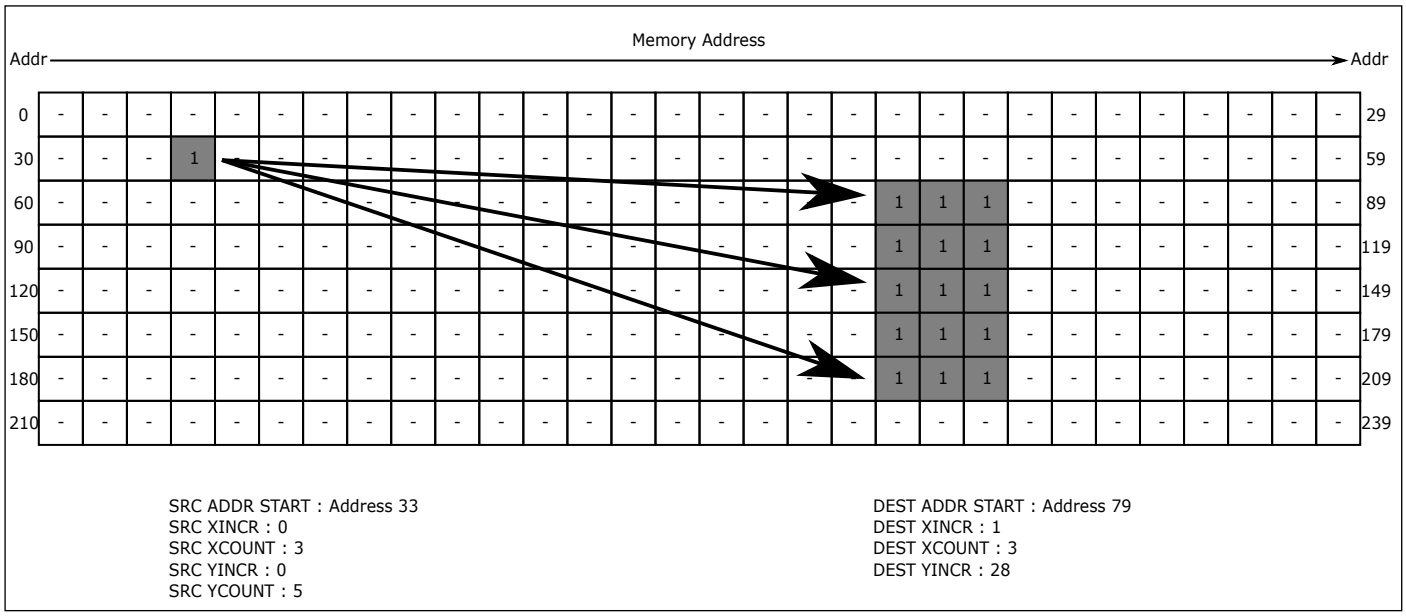


Fig. 8.5: Image Filling

As shown in the figure, it is assumed that a pixel width (P) is 8 bit, and the transfer width of 2D DMA is set to 8 bit. In an M × N image (30 × 8 in the figure), a rectangular area of W × H (3 × 5 in the figure) with starting coordinate (x2, y2) [(19, 2) in the figure] is filled by the value of the coordinate (x1, y1) [(3, 1) in the figure]. The start address of image data starts from addr0 (0 in the figure). Relevant parameters of 2D DMA are calculated as follows:

- SRC ADDR START = addr0+(M*y1+x1)*P = 0+(30*1+3)*1 = 33
- SRC XINCR = 0
- SRC XCOUNT = W = 3
- SRC YINCR = 0
- SRC YCOUNT = H = 5

- DEST ADDR START = addr0+(M*y2+x2)*P = 0+(30*2+19)*1 = 79
- DEST XINCR = P = 1
- DEST XCOUNT = W = 3
- DEST YINCR = (M-(W-1))*P = (30-(3-1))*1 = 28

8.3.6 Color Key

When Color Key is enabled, if 2D DMA encounters a value equal to the preset Color Key value during data transfer, it skips the value and the original value is still stored in the skipped address. The width of Color Key can be set to 8/16/24/32-bit. The working principle is shown below:

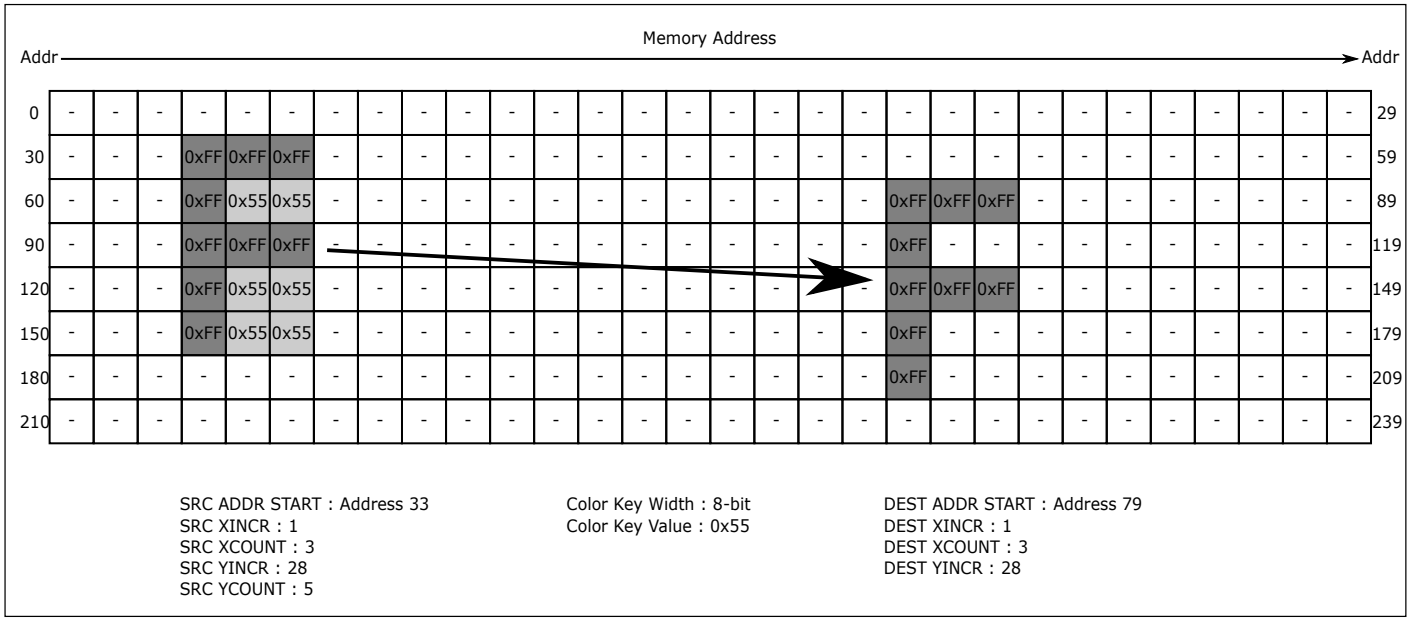


Fig. 8.6: Color Key

As shown in the above figure, the width of Color Key is set to 8 bit and the Key value is set to 0x55. When transferring the F-shaped rectangular area, 2D DMA skips the 0x55 value. So it can be seen from the right side that only the F-shaped rectangular area is transferred and the area that contains the 0x55 value is not transferred.

8.4 Register description

Name	Description
DMA2D_IntStatus	
DMA2D_IntTCStatus	
DMA2D_IntTCClear	

Name	Description
DMA2D_EnbldChns	
DMA2D_Config	
DMA2D_Sync	
DMA2D_SoftBReq	
DMA2D_SoftLBReq	
DMA2D_SoftSReq	
DMA2D_SoftLSReq	
DMA2D_C0SrcAddr	
DMA2D_C0DstAddr	
DMA2D_C0LLI	
DMA2D_C0_BUS	
DMA2D_C0_SRC_CNT	
DMA2D_C0_SRC_XIC	
DMA2D_C0_SRC_YIC	
DMA2D_C0_DST_CNT	
DMA2D_C0_DST_XIC	
DMA2D_C0_DST_YIC	
DMA2D_C0_KEY	
DMA2D_C0_KEY_EN	
DMA2D_C0_CFG	

8.4.1 DMA2D_IntStatus

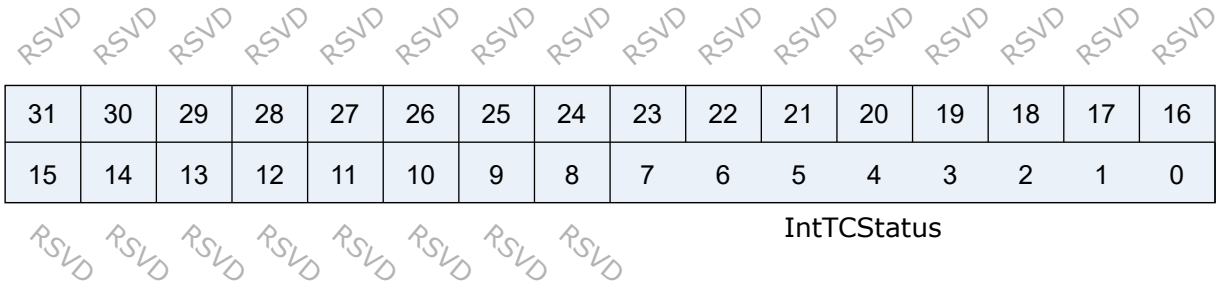
Address：0x30006000

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	IntStatus							

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

8.4.2 DMA2D_IntTCStatus

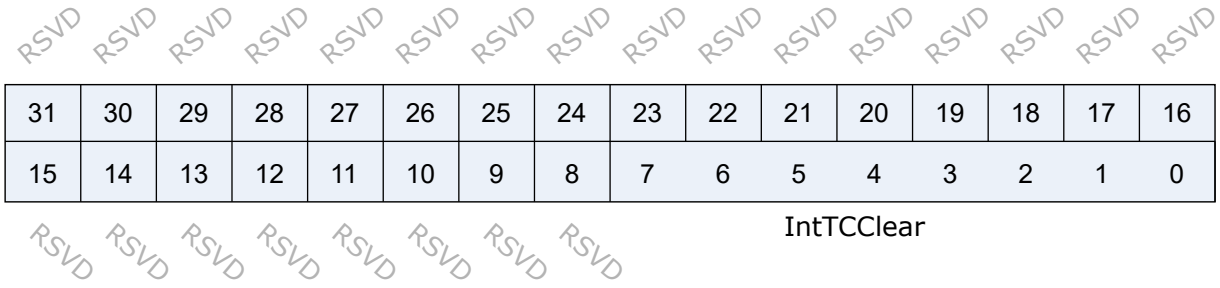
Address：0x30006004



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

8.4.3 DMA2D_IntTCClear

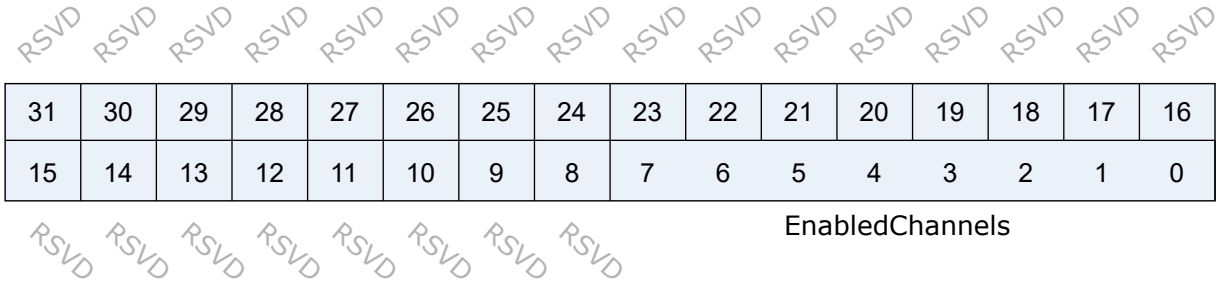
Address：0x30006008



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	IntTCClear	w1p	0	Terminal count request clear

8.4.4 DMA2D_EnbldChns

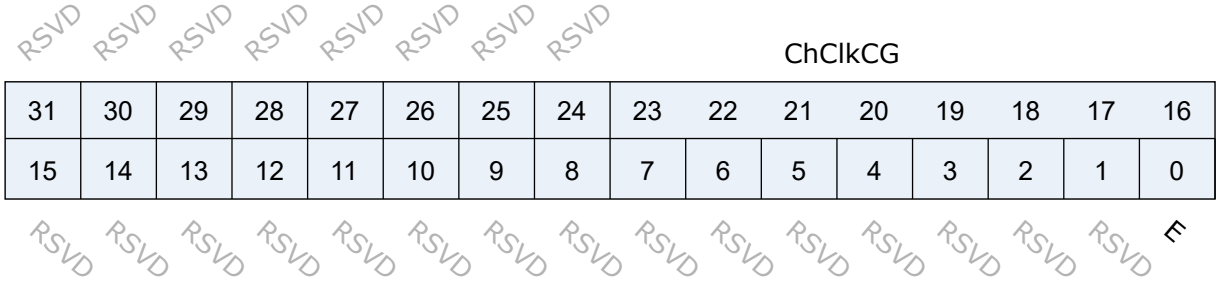
Address：0x3000600c



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

8.4.5 DMA2D_Config

Address：0x30006010



Bits	Name	Type	Reset	Description
31:24	RSVD			
23:16	ChClkCG	r/w	8'hff	Channel Clock cg enable
15:1	RSVD			
0	E	r/w	0	SMDMA Enable.

8.4.6 DMA2D_Sync

Address：0x30006014

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

8.4.7 DMA2D_SoftBReq

Address：0x30006018

SoftBReq

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

SoftBReq

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

8.4.8 DMA2D_SoftLBReq

Address：0x3000601c

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

8.4.9 DMA2D_SoftSReq

Address：0x30006020

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 signle request

8.4.10 DMA2D_SoftLSReq

Address：0x30006024

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 signle request

8.4.11 DMA2D_C0SrcAddr

Address：0x30006100

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

8.4.12 DMA2D_C0DstAddr

Address：0x30006104

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

8.4.13 DMA2D_C0LLI

Address：0x30006108

LLI_ADDR

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

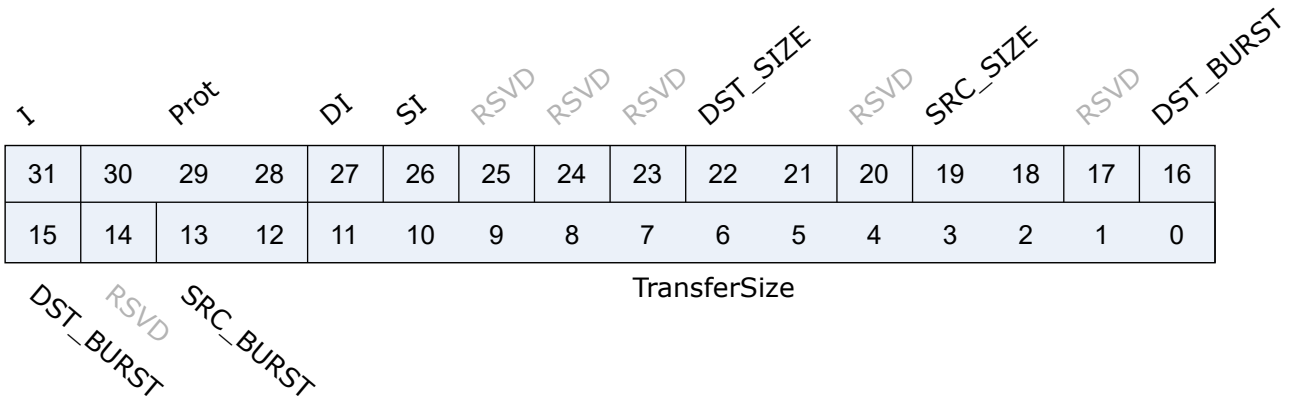
LLI_ADDR

RSVD LLI_EN

Bits	Name	Type	Reset	Description
31:2	LLI_ADDR	r/w	0	Next linked list address
1	RSVD			
0	LLI_EN	r/w	0	Next LLI enable

8.4.14 DMA2D_C0_BUS

Address：0x3000610c



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	Hprot Setting
27	DI	r/w	1	Only work when reg_dma_2d_en = 0 Destination increment. When set, the Destination address is incremented after each transfer.
26	SI	r/w	1	Only work when reg_dma_2d_en = 0 Source increment. When set, the source address is incremented after each transfer.
25:23	RSVD			
22:21	DST_SIZE	r/w	2'd2	Destination transfer width: 8/16/32
20	RSVD			
19:18	SRC_SIZE	r/w	2'd2	Source transfer width: 8/16/32
17	RSVD			
16:15	DST_BURST	r/w	2'd1	Destination burst size: 1/4/8/16
14	RSVD			
13:12	SRC_BURST	r/w	2'd1	
11:0	TransferSize	r/w	0	Only work when reg_dma_2d_en = 0 Transfer size: 0 4095. Number of data transfers left to complete when the SMDMA is the flow controller.

8.4.15 DMA2D_C0_SRC_CNT

Address：0x30006110

SRC_Y_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

SRC_X_CNT

Bits	Name	Type	Reset	Description
31:16	SRC_Y_CNT	r/w	16'd0	Only work when reg_dma_2d_en = 1 Source Y-direction Count
15:0	SRC_X_CNT	r/w	16'd0	Only work when reg_dma_2d_en = 1 Source X-direction Count

8.4.16 DMA2D_C0_SRC_XIC

Address：0x30006114

SRC_X_INCR

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

SRC_X_INCR

Bits	Name	Type	Reset	Description
31:0	SRC_X_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Source Y-direction address increment Must allign "SWidth" setting [16] sign-bit

8.4.17 DMA2D_C0_SRC_YIC

Address：0x30006118

SRC_Y_INCR

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

SRC_Y_INCR

Bits	Name	Type	Reset	Description
31:0	SRC_Y_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Source Y-direction address increment Must allign "SWidth" setting [16] sign-bit

8.4.18 DMA2D_C0_DST_CNT

Address：0x3000611c

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
DST_X_CNT															

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	DST_X_CNT	r/w	16'd0	Only work when reg_dma_2d_en = 1 Source X-direction Count

8.4.19 DMA2D_C0_DST_XIC

Address：0x30006120

DST_X_INCR															
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
DST_X_INCR															

Bits	Name	Type	Reset	Description
31:0	DST_X_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Source Y-direction address increment Must allign "SWidth" setting [16] sign-bit

8.4.20 DMA2D_C0_DST_YIC

Address：0x30006124

DST_Y_INCR

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

DST_Y_INCR

Bits	Name	Type	Reset	Description
31:0	DST_Y_INCR	r/w	32'd0	Only work when reg_dma_2d_en = 1 Desitination Y-direction address increment Must allign "DWidth" setting [16] sign-bit

8.4.21 DMA2D_C0_KEY

Address：0x30006174

KEY3 KEY2

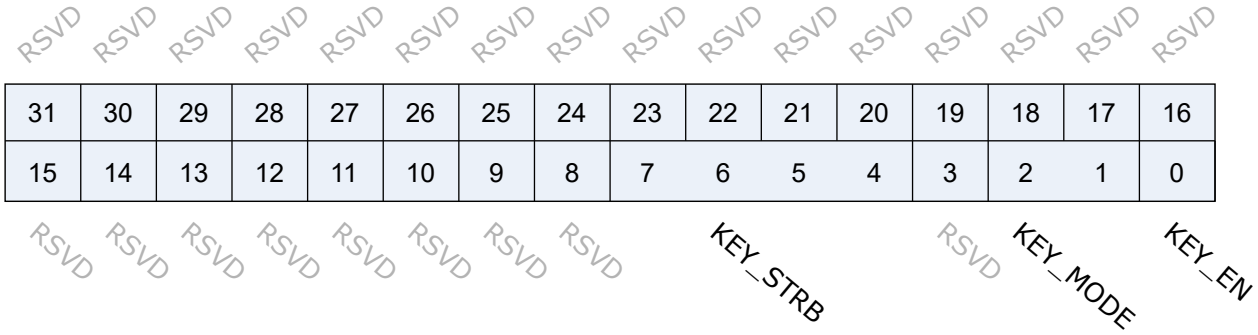
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

KEY1 KEY0

Bits	Name	Type	Reset	Description
31:24	KEY3	r/w	0	Key value
23:16	KEY2	r/w	0	Key value
15:8	KEY1	r/w	0	Key value
7:0	KEY0	r/w	0	Key value

8.4.22 DMA2D_C0_KEY_EN

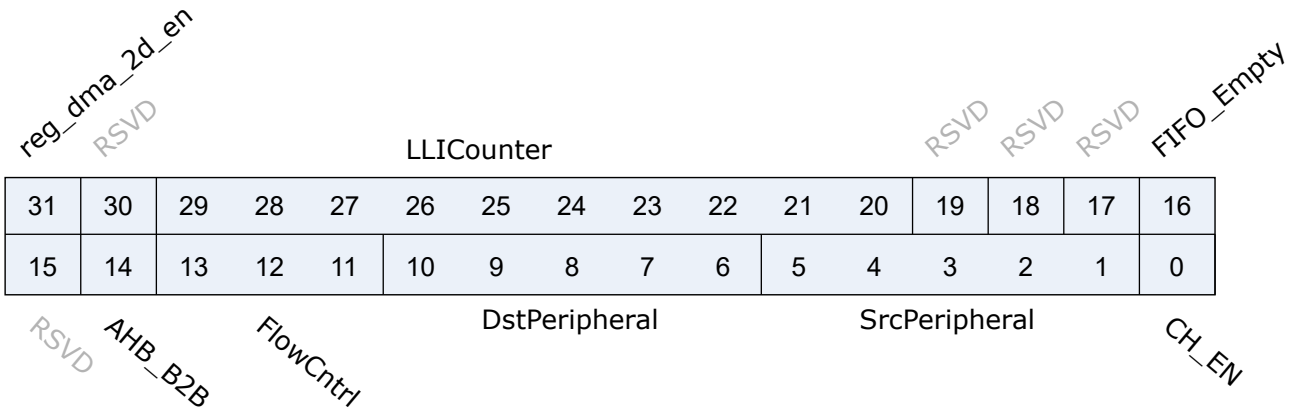
Address：0x30006178



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:4	KEY_STRB	r/w	4'h0	According Key mode set Key STRB Key mode : Key STRB 2'd0 : 4'h1 2'd1 : 4'h3 2'd2 : 4'h7 2'd3 : 4'hf
3	RSVD			
2:1	KEY_MODE	r/w	0	Key Mode 2'd0 : 8-bit mode 2'd1 : 16-bit mode 2'd2 : 24-bit mode 2'd3 : 32-bit mode
0	KEY_EN	r/w	0	Key enable

8.4.23 DMA2D_C0_CFG

Address：0x3000617c



Bits	Name	Type	Reset	Description
31	reg_dma_2d_en	r/w	0	DMA 2d enable
30	RSVD			
29:20	LLICounter	r	0	LLI counter. Increased 1 each LLI run. Cleared 0 when config Control.
19:17	RSVD			
16	FIFO_Empty	r	0	Active: 0 = no data in FIFO of the channel, 1 = FIFO of the channel has data.
15	RSVD			
14	AHB_B2B	r/w	0	AHB back to back enable
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) The following only work when reg_dma_2d_en=0 : 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	CH_EN	r/w	0	Channel enable.

9 LZ4D

9.1 Overview

LZ4 is a lossless data compression algorithm that features extremely fast decompression speed and average compression ratio. LZ4D is a functional module for decompression of data blocks compressed using LZ4. LZ4D contains a status and control register set used to set the start address of the input compressed data, the output address of the decompressed data, and the decompression status and optional interrupt configuration.

9.2 Features

- Supports data blocks compressed using LZ4
- Supports decompression of data blocks compressed at different levels
- Supports decompression of multiple compressed data blocks
- Event flags for decompression completion and errors
- Generates a corresponding interrupt when an event occurs

9.3 Functional Description

The module gets the compressed data blocks from the source address through bus access, decompresses them using the internal algorithm, and writes the decompressed data to the specified destination address. During decompression, the module updates relevant registers and sets flags for events including successful decompression and decompression errors. In turn, under these event statuses, the module can request interrupts from the CPU.

9.4 Clock

LZ4D Module requires one work clock.

9.5 Programming Flow

9.5.1 Data Decompression

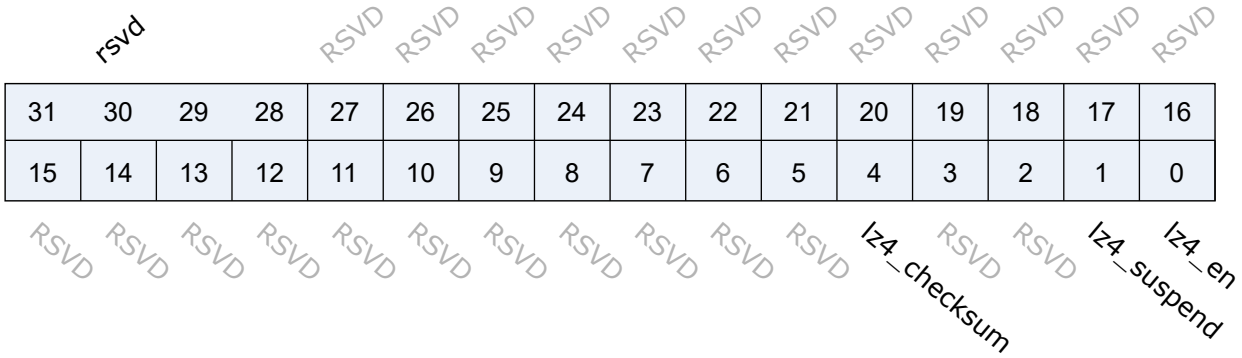
- Turn on the module’s clock
- Disable the module, write the address of the raw data to be decompressed (excluding compressed file header) to the start address register for source data
- Prepare a sufficient contiguous memory to hold the decompressed data, with a recommended length aligned on 4-byte boundary. Write the address of this memory block to the start address register for destination data
- If interrupts are required, install an interrupt handler function and set the corresponding interrupt enable bit
- In the control register, the module will start decompression once enabled
- If a “complete interrupt” is enabled, then wait for the “complete interrupt”
- If no interrupt is enabled, read the status register and then wait for the completion of decompression

9.6 Register description

Name	Description
lz4_config	
lz4_src_fix	
lz4_dst_fix	
lz4_src_start	
lz4_src_end	
lz4_dst_start	
lz4_dst_end	
lz4_int_en	
lz4_int_sta	
lz4_monitor	

9.6.1 lz4_config

Address：0x2000ad00

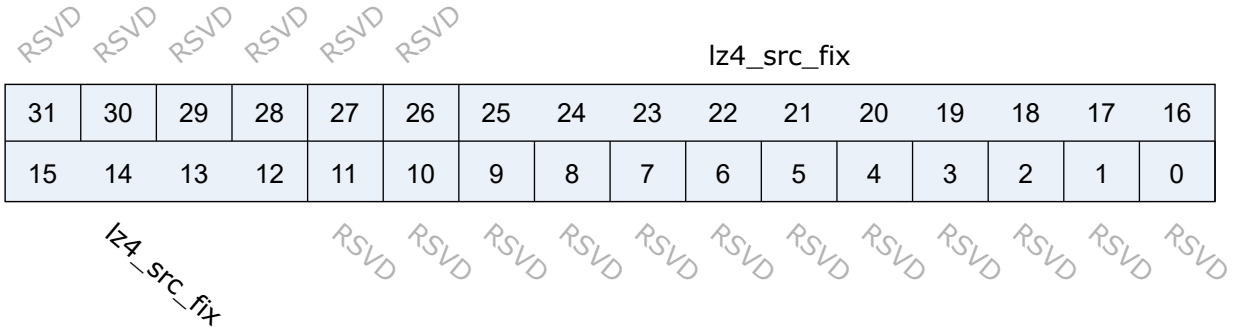


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Reset	Description
31:28	rsvd	rsvd	0	
27:5	RSVD			
4	lz4_checksum	r/w	0	LZ4 block has checksum or not (info from frame header checksum flag)
3:2	RSVD			
1	lz4_suspend	r/w	0	LZ4 blocks decode suspend (FSM idle)
0	lz4_en	r/w	0	LZ4 blocks decode enable (each block has size/sequences/checksum)

9.6.2 lz4_src_fix

Address：0x2000ad04



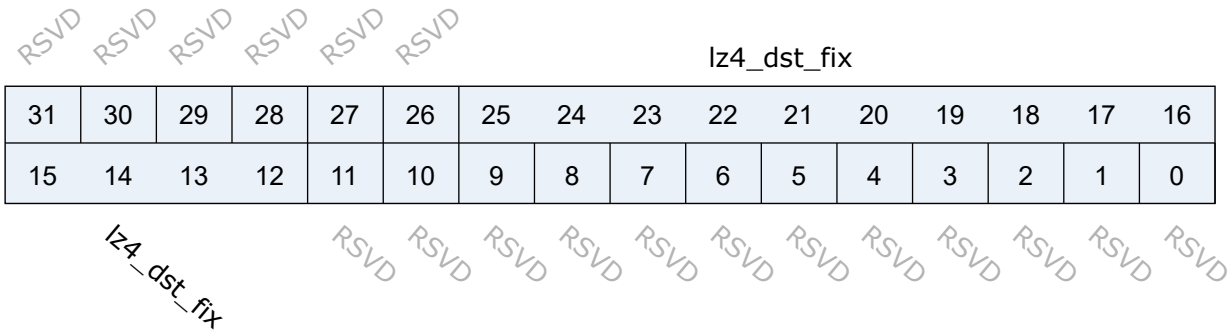
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Reset	Description
31:26	RSVD			
25:12	lz4_src_fix	r/w	0	LZ4 source address fix bits

Bits	Name	Type	Reset	Description
11:0	RSVD			

9.6.3 lz4_dst_fix

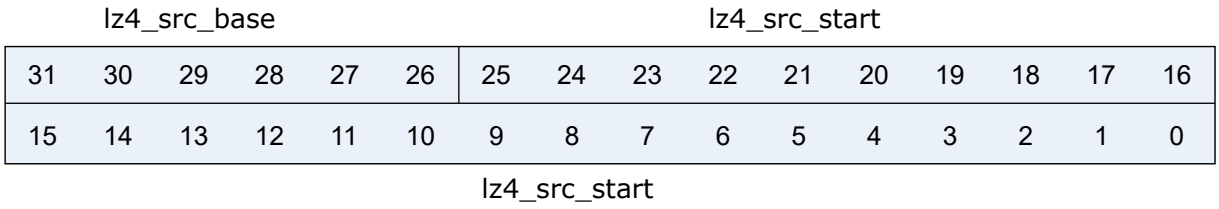
Address：0x2000ad08



Bits	Name	Type	Reset	Description
31:26	RSVD			
25:12	lz4_dst_fix	r/w	0	LZ4 destination address fix bits
11:0	RSVD			

9.6.4 lz4_src_start

Address：0x2000ad10



Bits	Name	Type	Reset	Description
31:26	lz4_src_base	r/w	0	LZ4 source address base (fix)
25:0	lz4_src_start	r/w	0	LZ4 source address start

9.6.5 lz4_src_end

Address：0x2000ad14

RSVD RSVD RSVD RSVD RSVD RSVD lz4_src_end

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

lz4_src_end

Bits	Name	Type	Reset	Description
31:26	RSVD			
25:0	lz4_src_end	r	0	LZ4 source address end (Max. 64MB)

9.6.6 lz4_dst_start

Address：0x2000ad18

lz4_dst_base lz4_dst_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

lz4_dst_start

Bits	Name	Type	Reset	Description
31:26	lz4_dst_base	r/w	0	LZ4 destination address base (fix)
25:0	lz4_dst_start	r/w	0	LZ4 destination address start

9.6.7 lz4_dst_end

Address：0x2000ad1c

RSVD RSVD RSVD RSVD RSVD RSVD lz4_dst_end

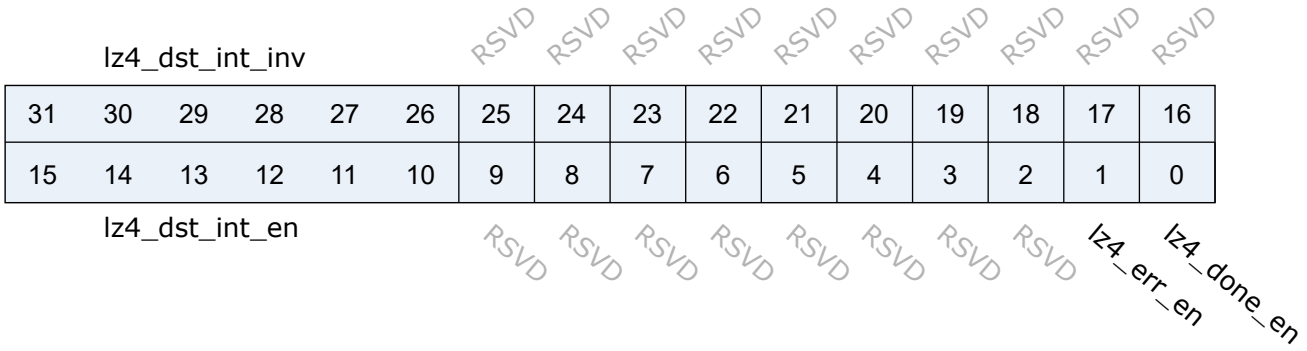
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

lz4_dst_end

Bits	Name	Type	Reset	Description
31:26	RSVD			
25:0	lz4_dst_end	r	0	LZ4 destination address end (Max. 64MB)

9.6.8 lz4_int_en

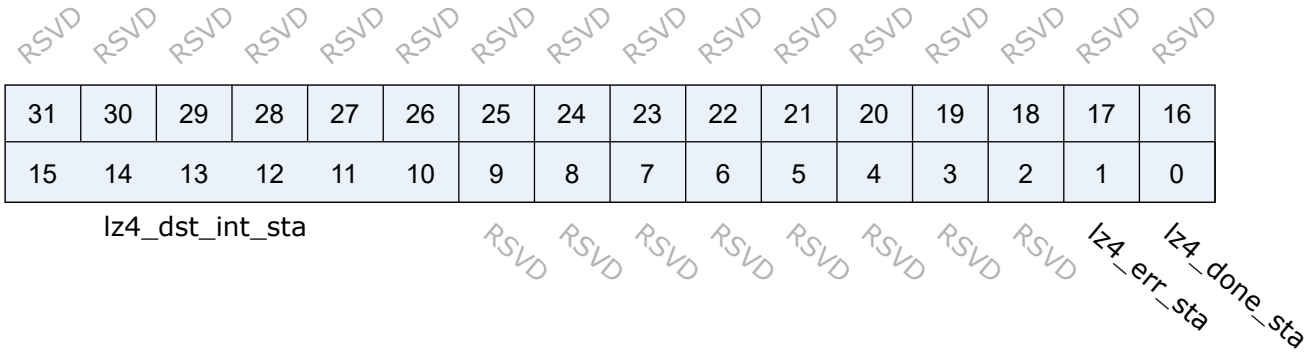
Address：0x2000ad20



Bits	Name	Type	Reset	Description
31:26	lz4_dst_int_inv	r/w	0	0: high int 1: low int
25:16	RSVD			
15:10	lz4_dst_int_en	r/w	0	
9:2	RSVD			
1	lz4_err_en	r/w	1	
0	lz4_done_en	r/w	1	

9.6.9 lz4_int_sta

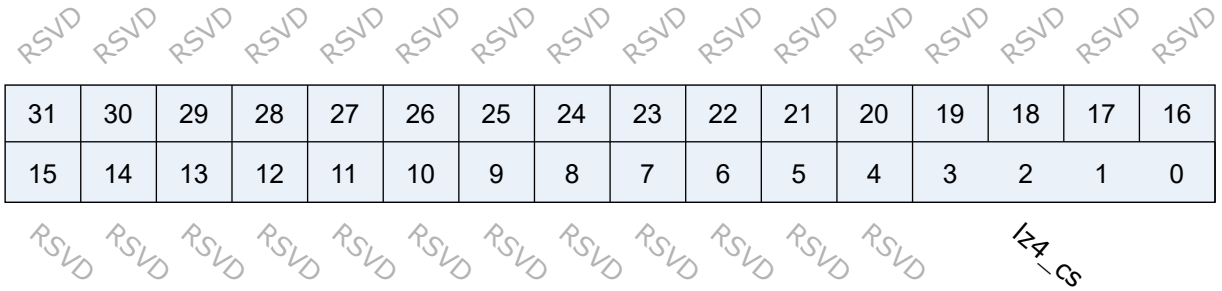
Address：0x2000ad24



Bits	Name	Type	Reset	Description
31:16	RSVD			
15:10	lz4_dst_int_sta	r	0	LZ4 destination address bit change
9:2	RSVD			
1	lz4_err_sta	r	0	
0	lz4_done_sta	r	0	LZ4 done status

9.6.10 lz4_monitor

Address：0x2000ad28



Bits	Name	Type	Reset	Description
31:4	RSVD			
3:0	lz4_cs	r	0	

10 DBI

10.1 Overview

The Display Bus Interface (DBI), also known as MCU interface, is an interface protocol defined by MIPI Alliance for communicating with display screens. The data and control lines of the DBI protocol are multiplexed, and the driven display module must have GRAM.

DBI is subdivided into MIPI DBI Type A, MIPI DBI Type B, and MIPI DBI Type C modes. The hardware interfaces and data sampling in different modes are different. For example, MIPI DBI Type A uses falling edge sampling (based on Motorola 6800 bus), while MIPI DBI Type B uses the rising edge sampling (based on Intel 8080 bus).

10.2 Features

- Meets MIPI® Alliance standard
- Supports Type B and Type C modes
- Type B is an 8-bit data interface
- Type C supports 3Line and 4Line
- Supports 8 input pixel formats:
  - RGBA8888
  - BGRA8888
  - ARGB8888
  - ABGR8888
  - RGB888
  - BGR888
  - RGB565

  - BGR565

- Supported output pixel formats: RGB565, RGB666 and RGB888
- Multiple interrupt control
- Supports DMA transfer mode

10.3 Functional Description

The block diagram of DBI module is shown as follows.

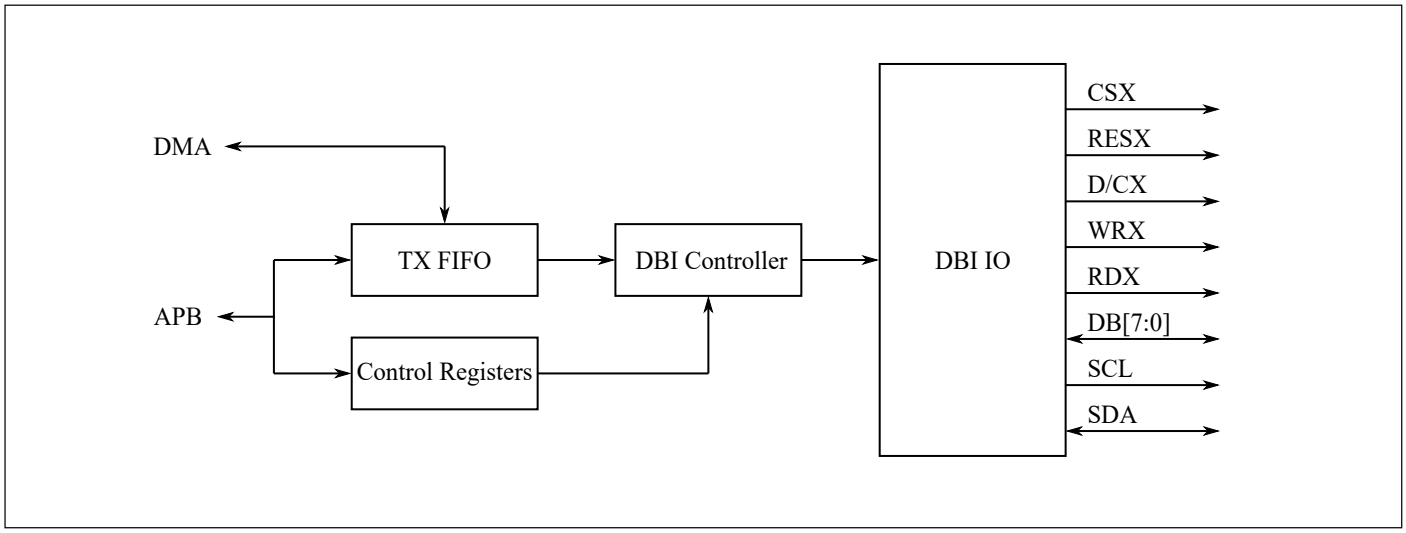


Fig. 10.1: Block Diagram of DBI

10.3.1 DBI Type B

10.3.1.1 Write Timing

The DBI Type B mode has 8-bit parallel data lines, and the timing of MCU writing data to the display module is shown as follows:

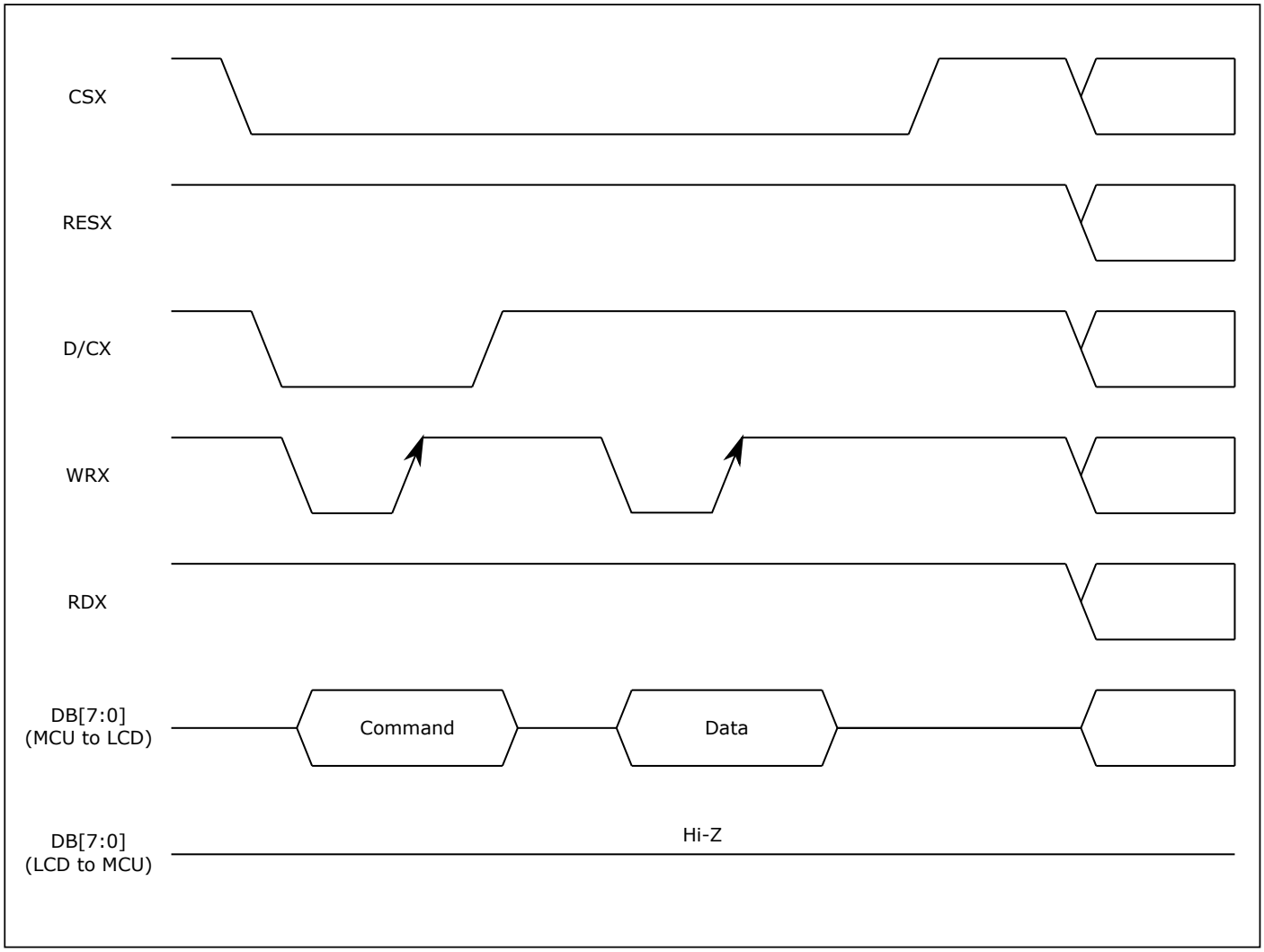


Fig. 10.2: Write Timing

- CSX: chip select signal. When the signal is Low, the display module will be selected; 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 will be reset.
- D/CX: data/command signal. When the signal is 0, the DB[7:0] bit is a command; when the signal is 1, the DB[7:0] bit is the RAM data or command parameter.
- WRX: parallel data write strobe signal. The signal is driven from high to low in the write cycle and then pulled back to high. The display module reads the information transmitted by MCU at the rising edge of the signal. This is an asynchronous signal and can be terminated if not in use.
- RDX: parallel data read strobe signal. The signal is driven from high to low and then pulled back to high in the read cycle. MCU reads the information transmitted by the display module at the rising edge of the signal. This is an asynchronous signal and can be terminated if not in use.
- DB[7:0]: 8-bit data signal, used to transmit commands, command parameters or data

10.3.1.2 Read Timing

The timing of MCU reading data from the display module is shown as follows:

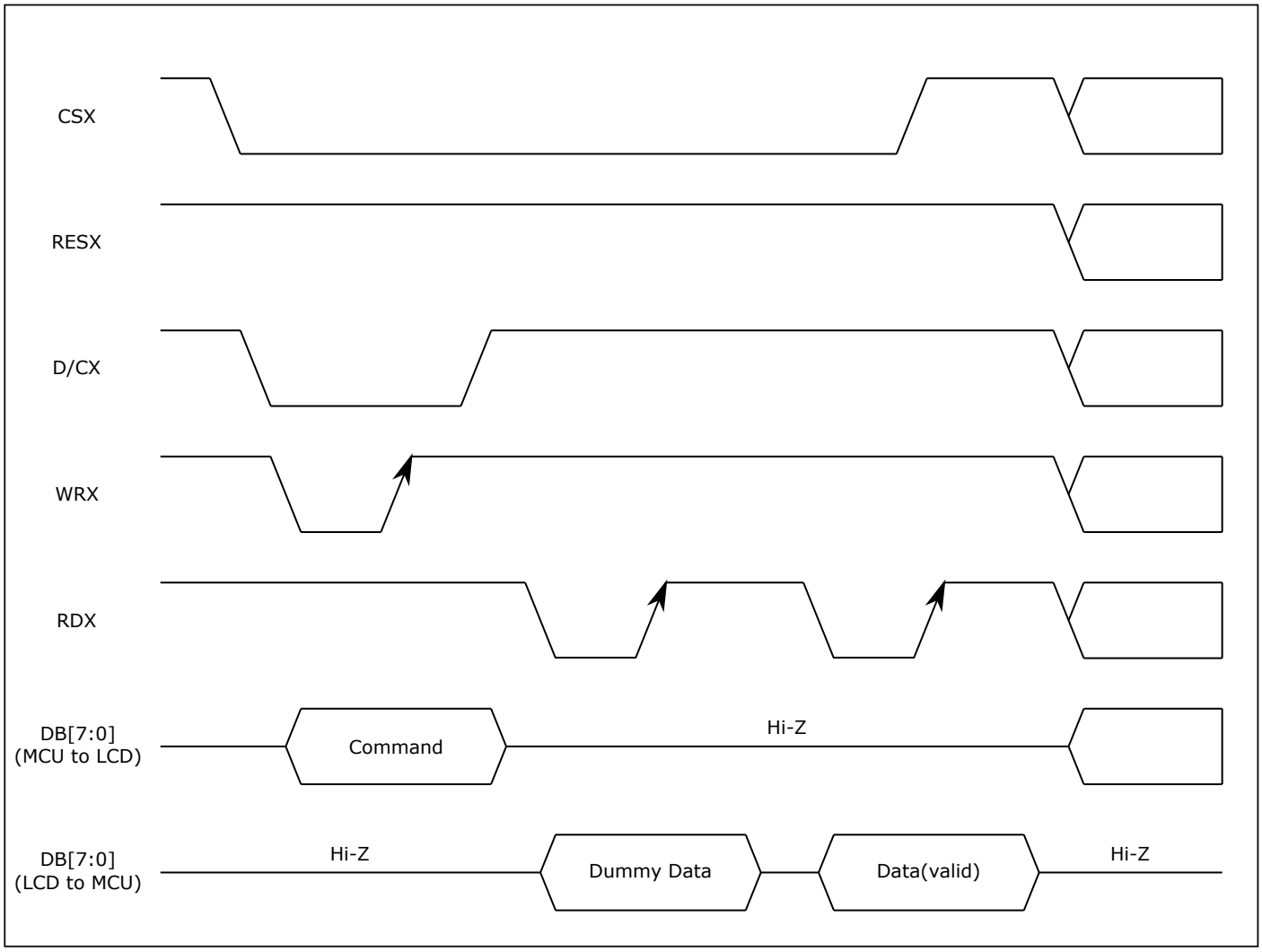


Fig. 10.3: Read Timing

10.3.1.3 Output RGB565

The data output in RGB565 format is shown as follows:

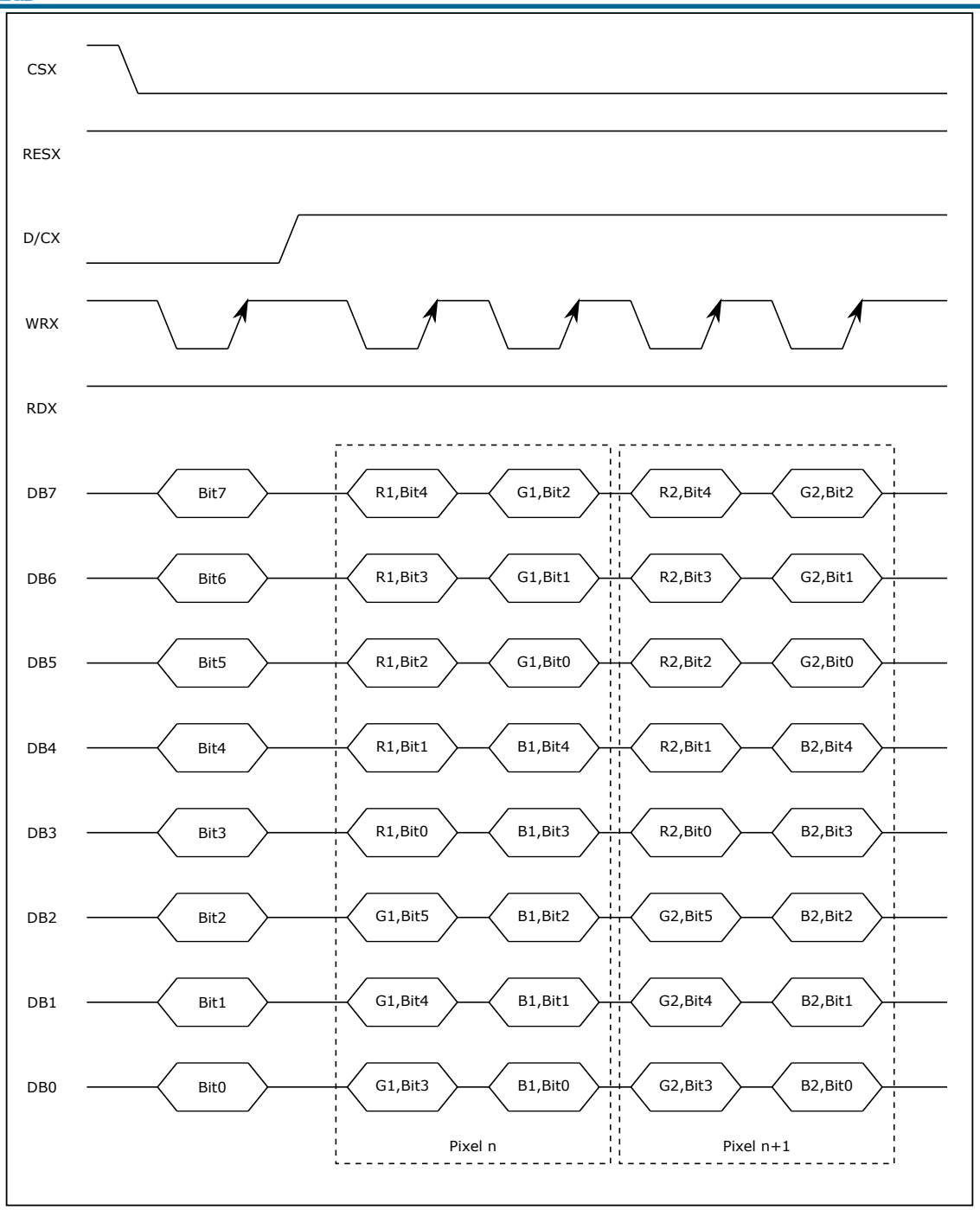


Fig. 10.4: RGB565 Output

10.3.1.4 Output RGB666

The data output in RGB666 format is shown as follows:

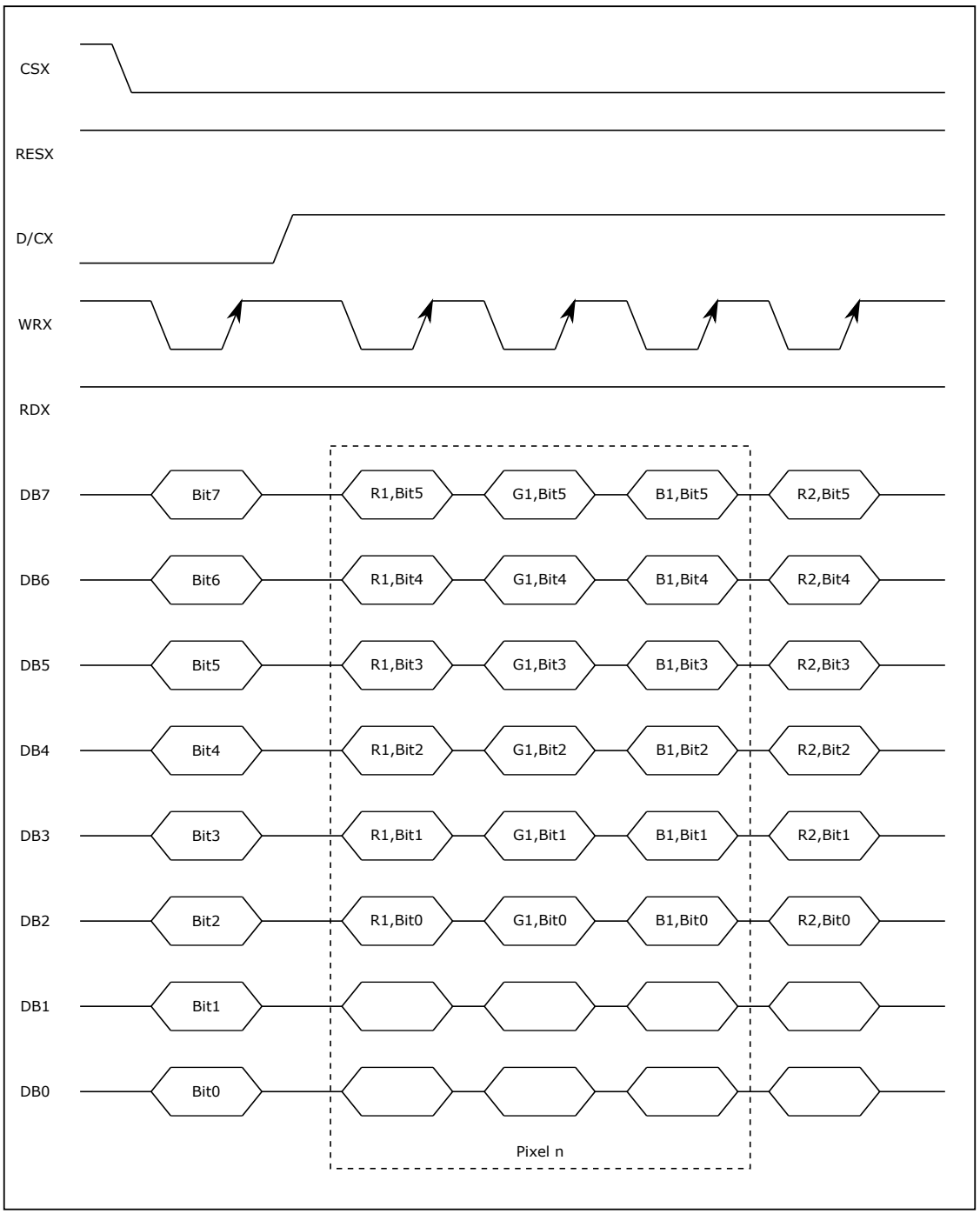


Fig. 10.5: RGB666 Output

10.3.1.5 Output RGB888

The data output in RGB888 format is shown as follows:

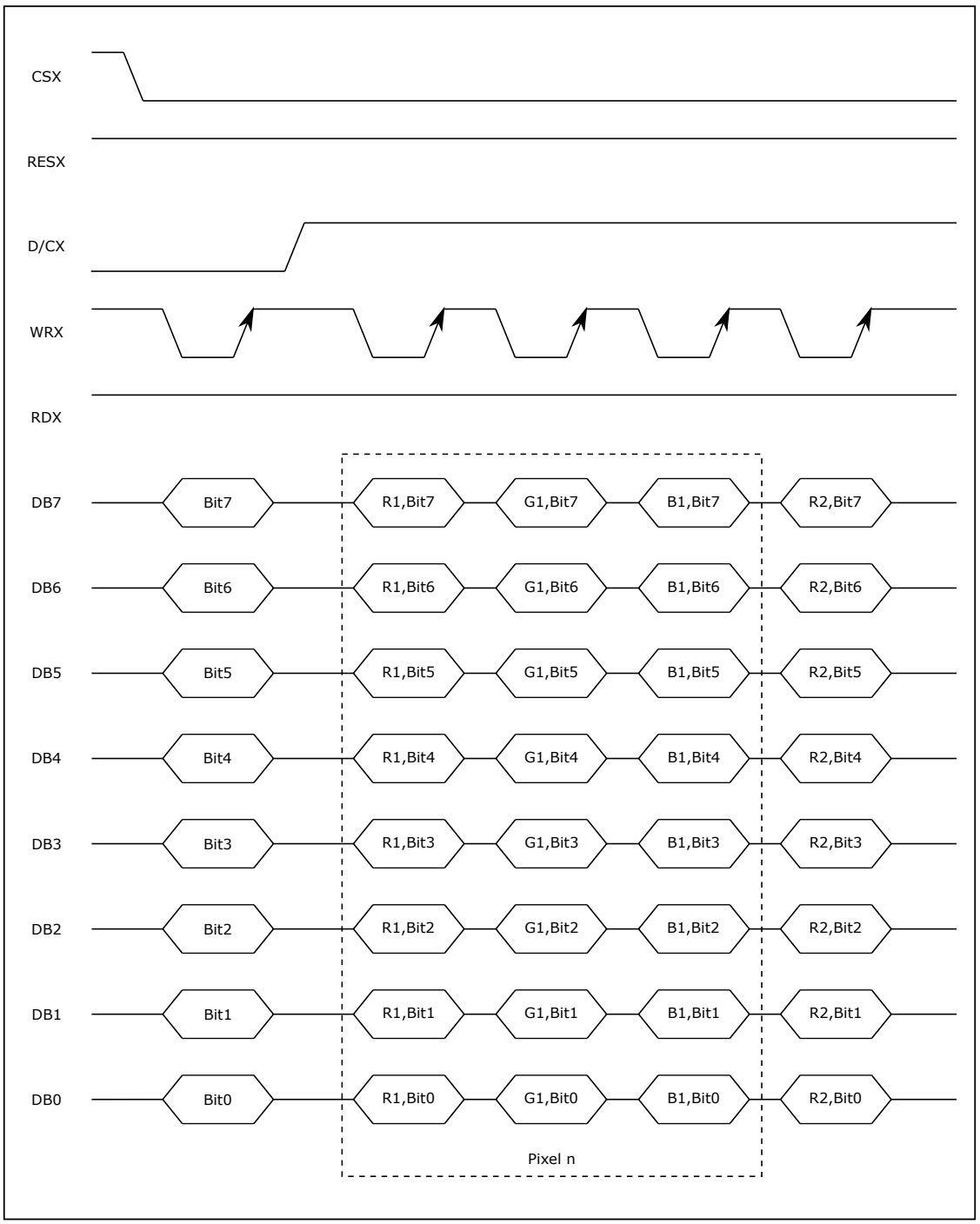


Fig. 10.6: RGB888 Output

10.3.2 DBI Type C 3-Line

10.3.2.1 Write Timing

The DBI Type C mode has 3-line 9-bit serial interface, and the timing of MCU writing data to the display module is shown as follows:

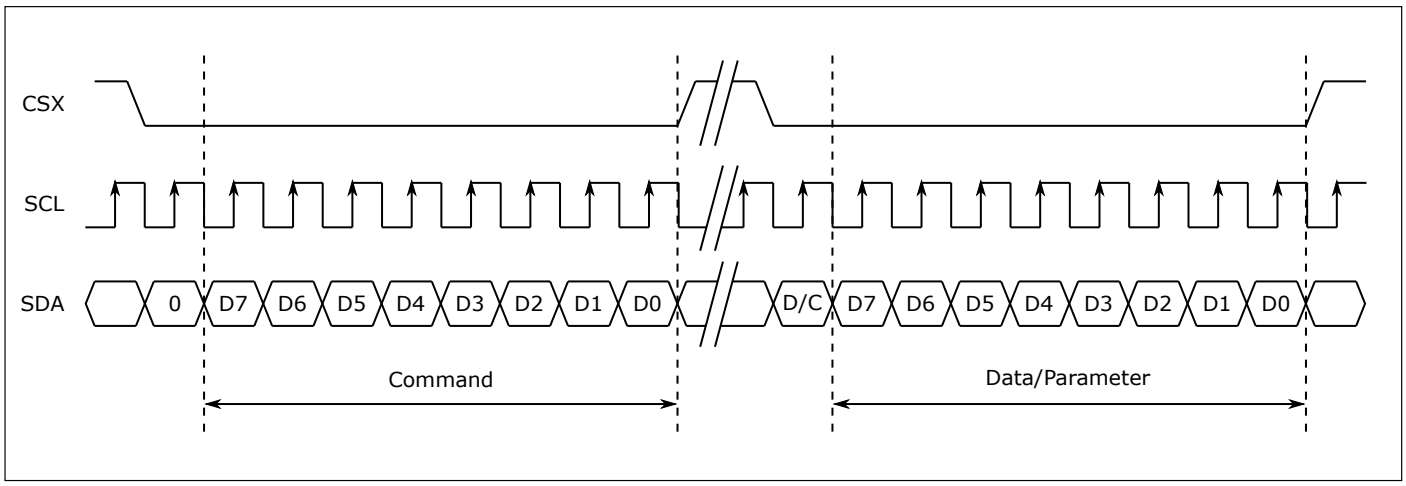


Fig. 10.7: Write Timing

- CSX: chip select signal. When the signal is Low, the display module will be selected; when it is High, the display module will ignore all other interface signals.
- SCL: serial clock line, which provides a clock signal for data transfer.
- SDA: serial data line. In a write operation, the serial data packet contains a D/CX (data/command) selection bit and a transfer byte. If the D/CX bit is Low, the transfer byte is a command. If the D/CX bit is High, that is the display data or command parameter.

10.3.2.2 Read Timing

The timing of MCU reading data from the display module is shown as follows:

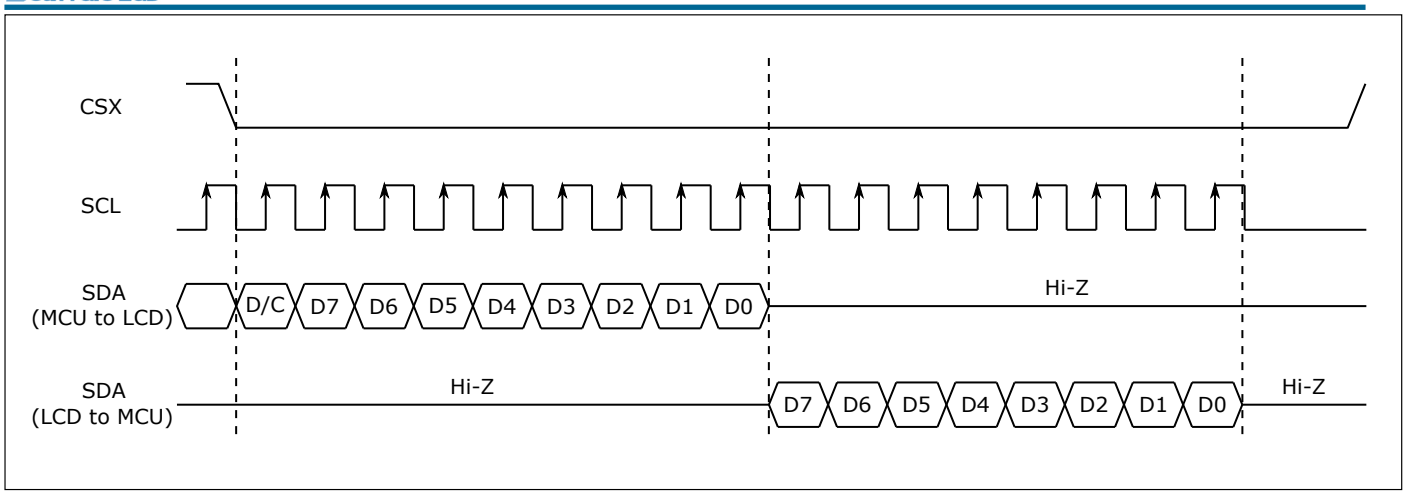


Fig. 10.8: Read Timing

10.3.2.3 Output RGB565

The data output in RGB565 format is shown as follows:

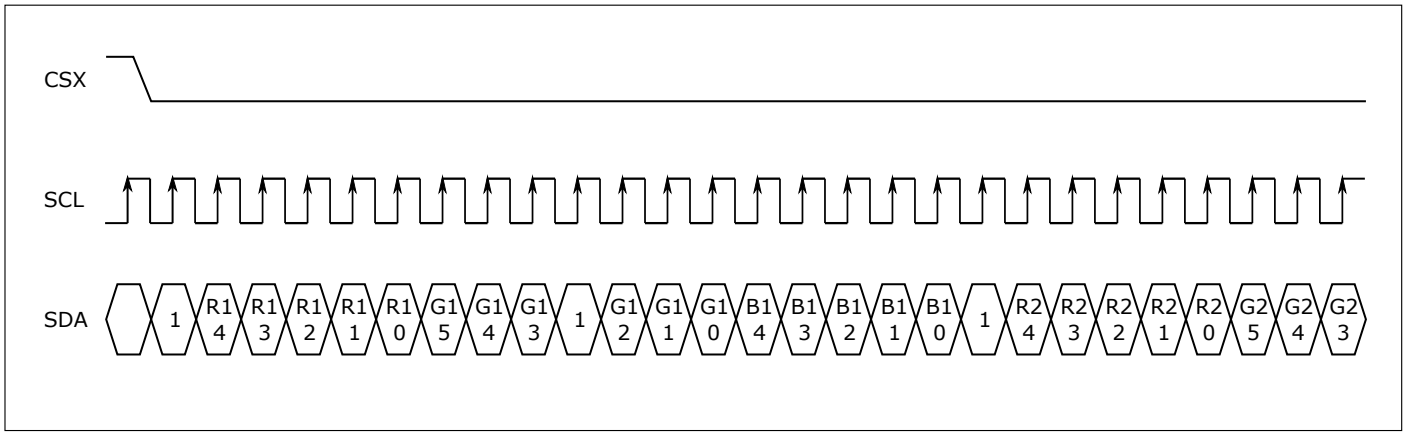


Fig. 10.9: RGB565 Output

10.3.2.4 Output RGB666

The data output in RGB666 format is shown as follows:

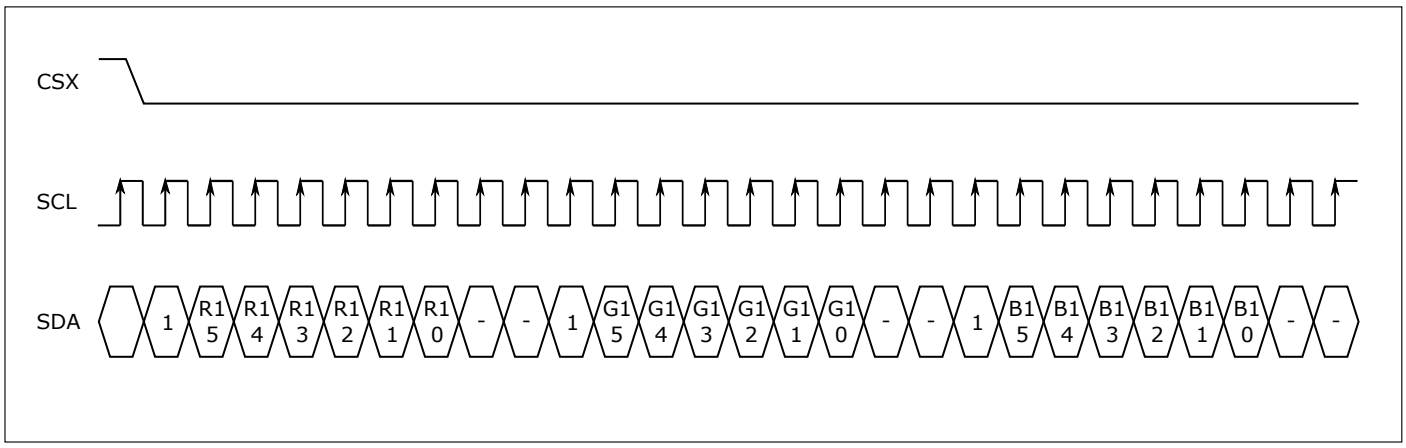


Fig. 10.10: RGB666 Output

10.3.2.5 Output RGB888

The data output in RGB888 format is shown as follows:

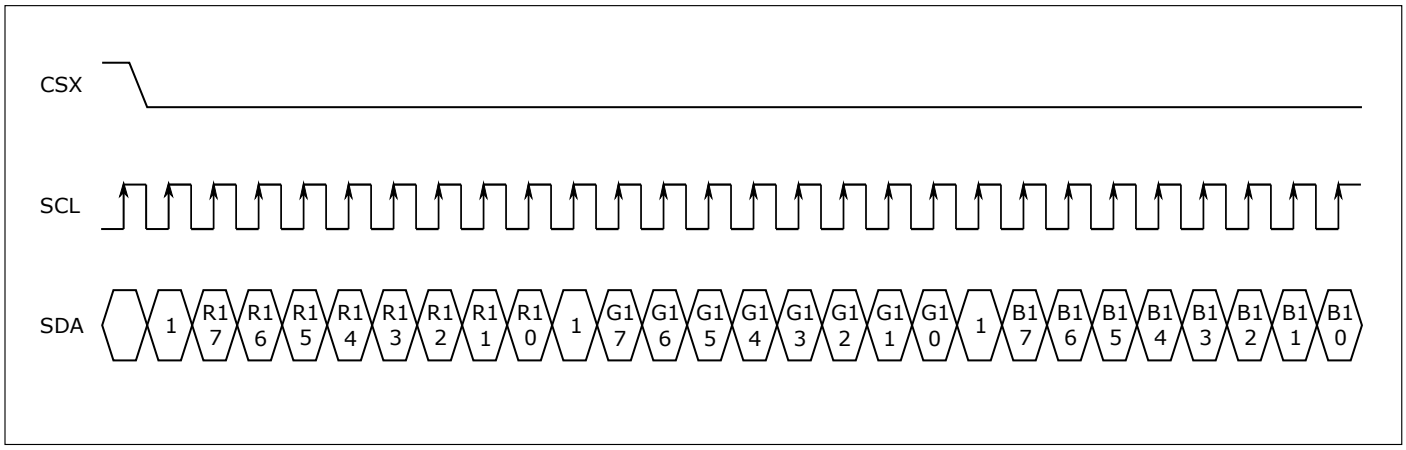


Fig. 10.11: RGB888 Output

10.3.3 DBI Type C 4-Line

10.3.3.1 Write Timing

The DBI Type C mode has 4-line 8-bit serial interface, and the timing of MCU writing data to the display module is shown as follows:

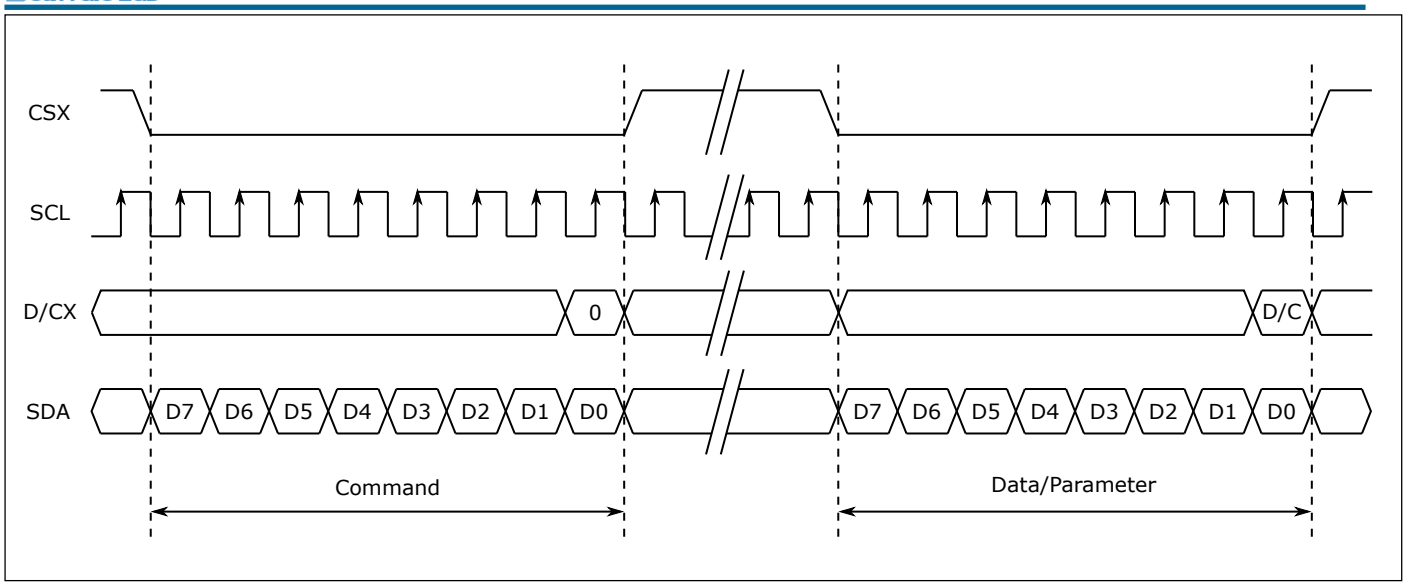


Fig. 10.12: Write Timing

- CSX: chip select signal. When the signal is Low, the display module will be selected; when it is High, the display module will ignore all other interface signals.
- D/CX: data/command signal. If the signal is Low, the information transferred by SDA is a command. If the signal is High, that is the display data or command parameter.
- SCL: serial clock line, which provides a clock signal for data transfer.
- SDA: serial data line. used to transmit commands, command parameters or data.

10.3.3.2 Read Timing

The timing of MCU reading data from the display module is shown as follows:

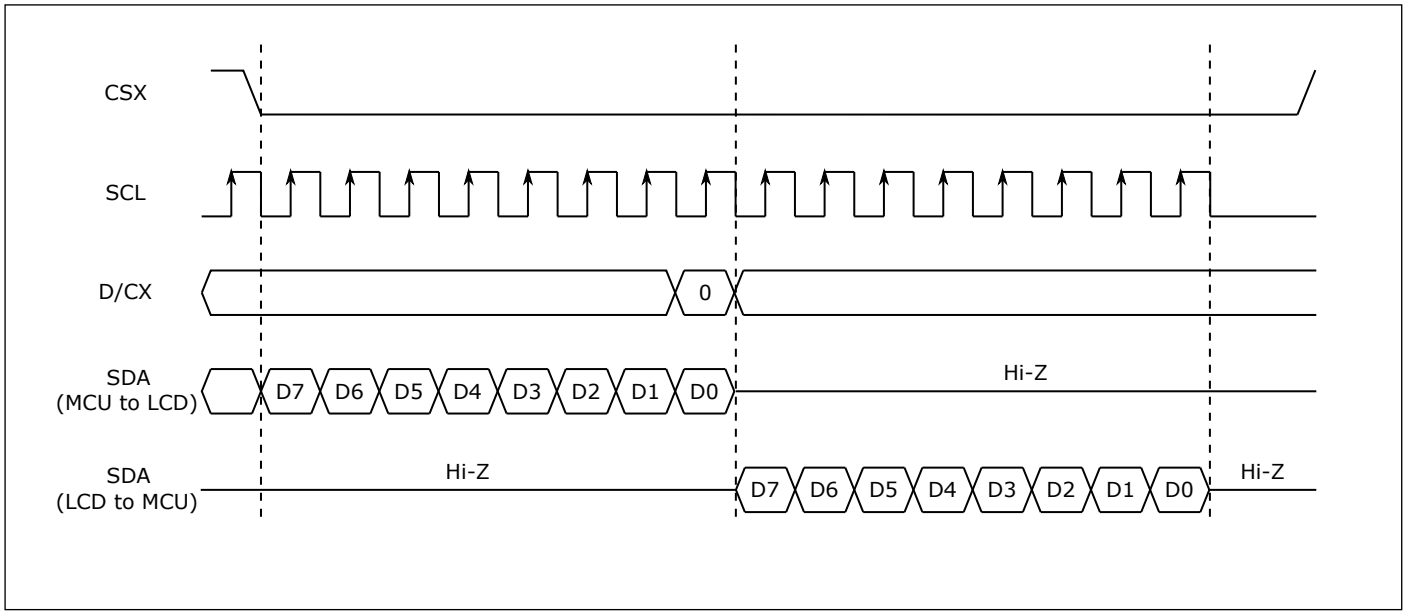


Fig. 10.13: Read Timing

10.3.3.3 Output RGB565

The data output in RGB565 format is shown as follows:

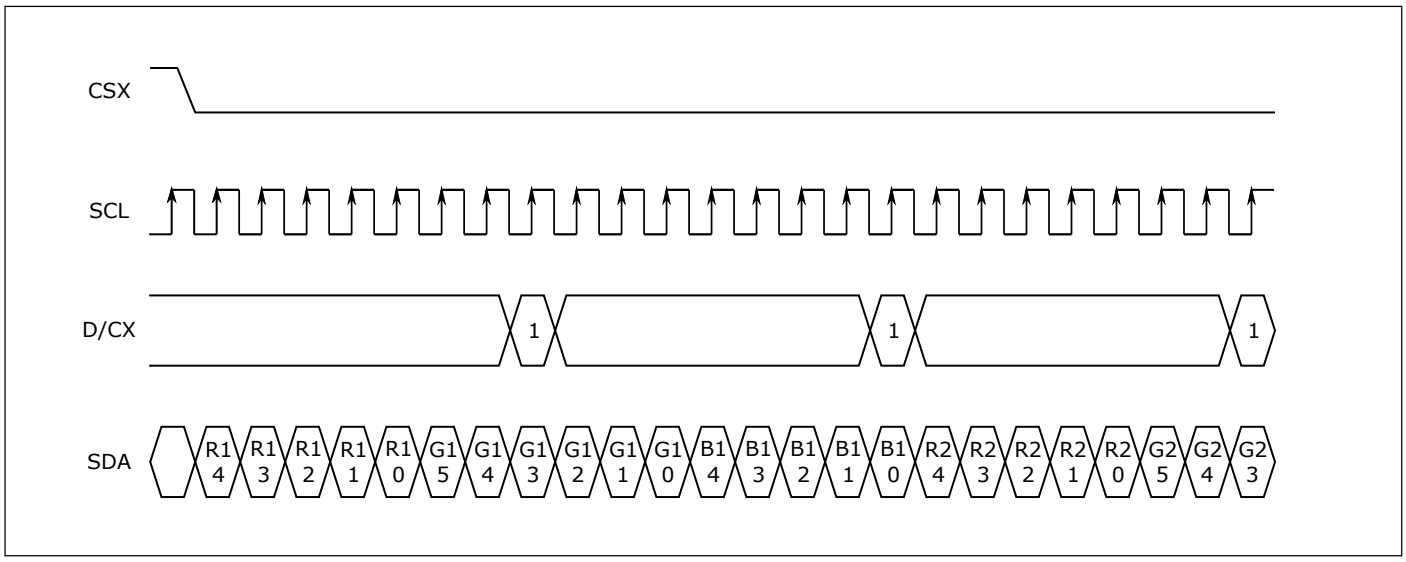


Fig. 10.14: RGB565 Output

10.3.3.4 Output RGB666

The data output in RGB666 format is shown as follows:

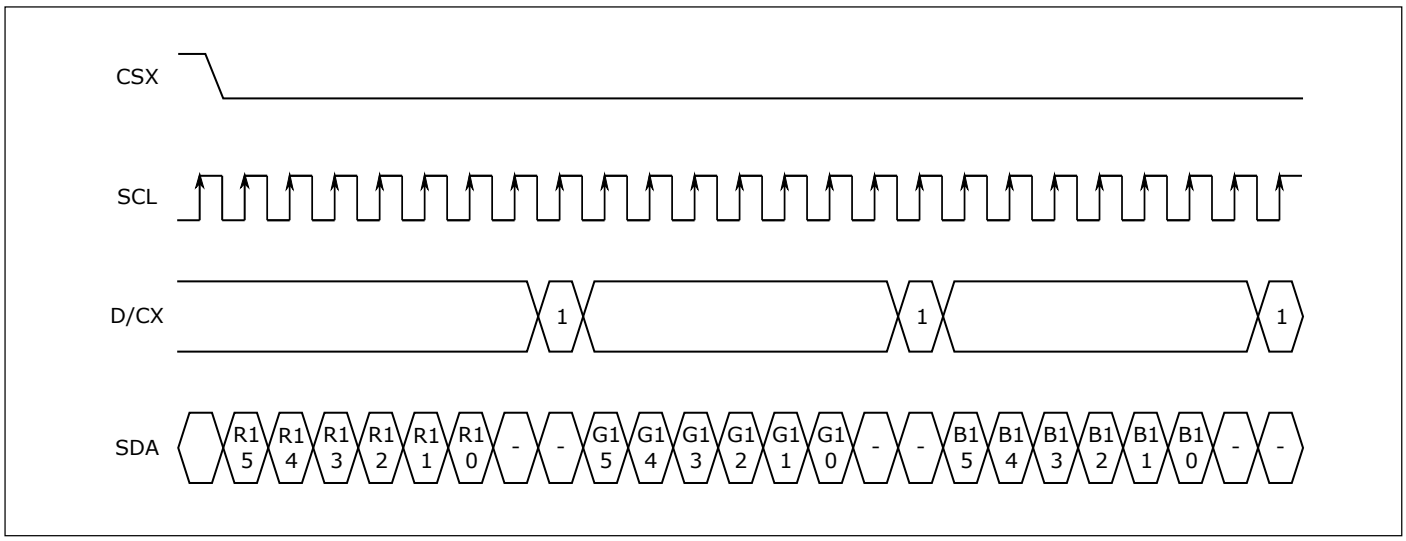


Fig. 10.15: RGB666 Output

10.3.3.5 Output RGB888

The data output in RGB888 format is shown as follows:

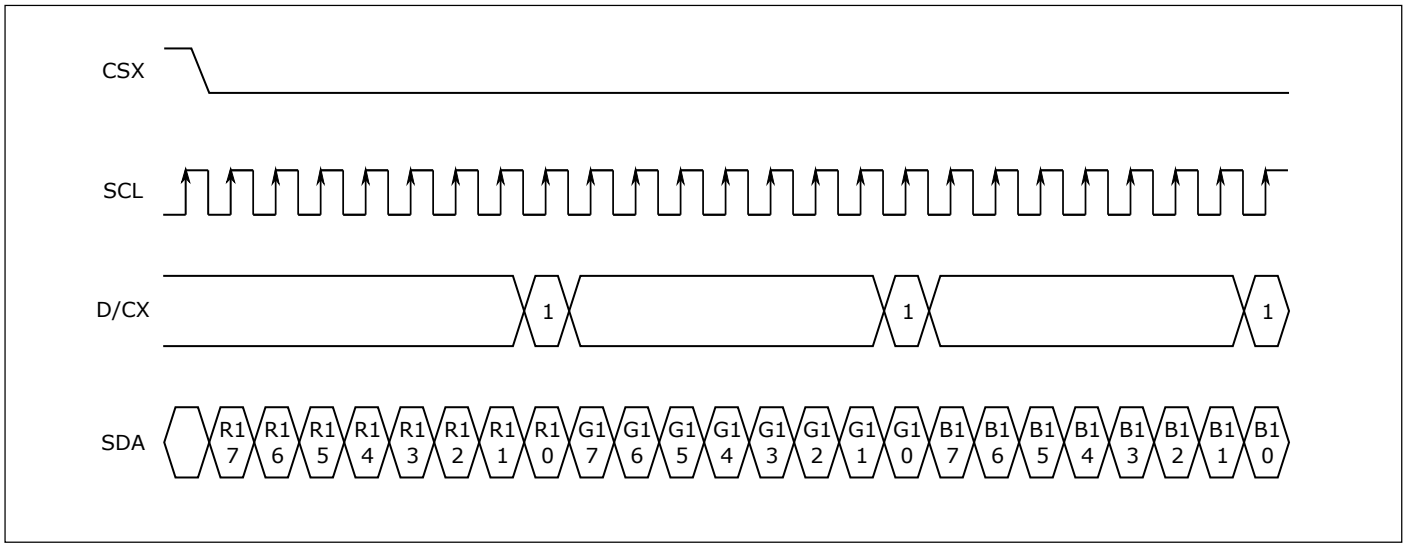


Fig. 10.16: RGB888 Output

10.3.4 Input Pixel Format

The DBI module supports 8 different input pixel formats, whose arrangement in the memory is shown as follows:

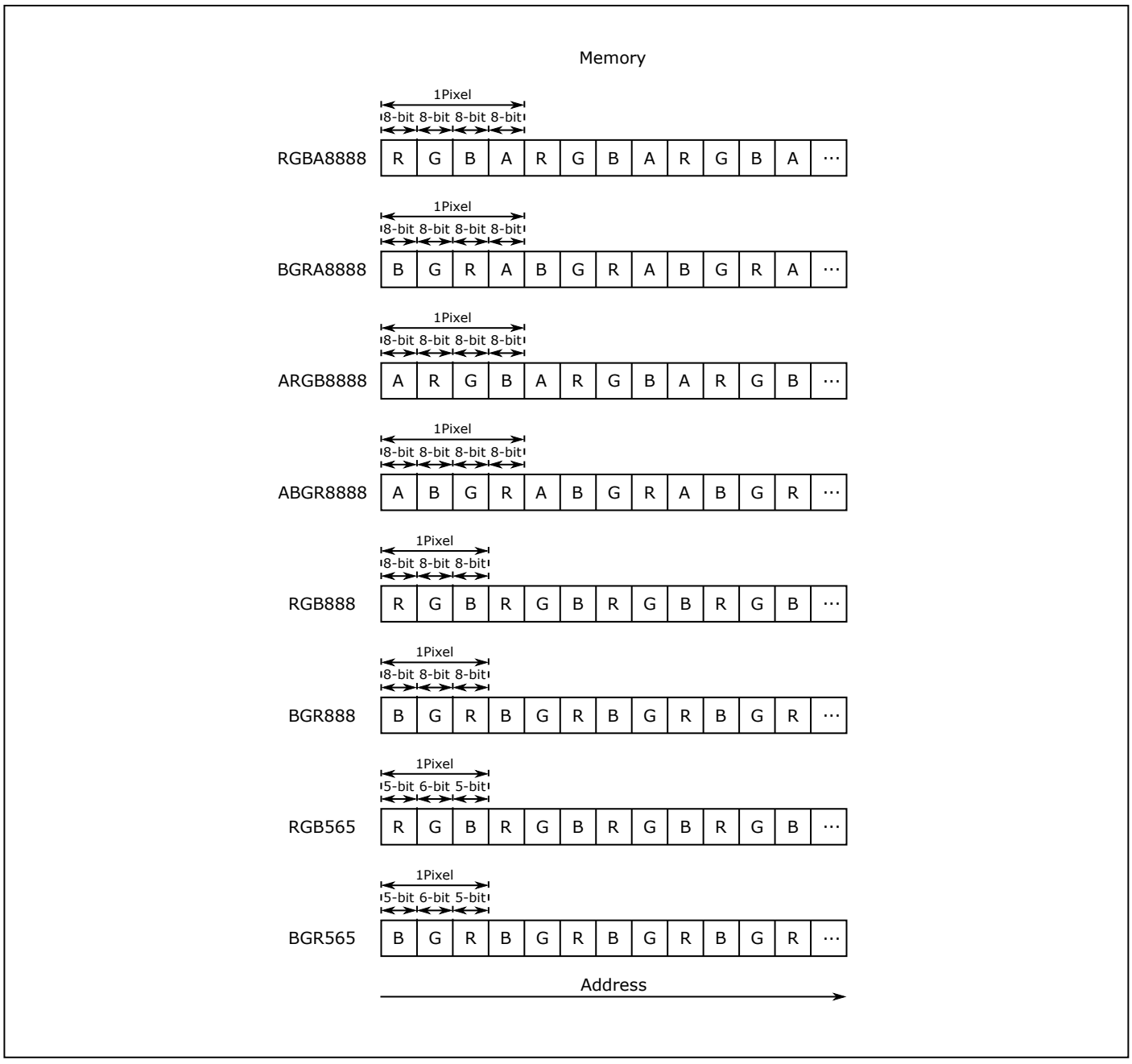


Fig. 10.17: Input Pixel Format

10.3.5 Interrupt

DBI has the following interrupts:

- End of transfer interrupt
- TX FIFO request interrupt
- FIFO overflow error interrupt

When a pixel count value is set, the end of transfer interrupt will be generated when the transferred 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, the TX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not met.

The FIFO overflow error interrupt will be generated in the case of Overflow or Underflow of TX.

10.3.6 DMA

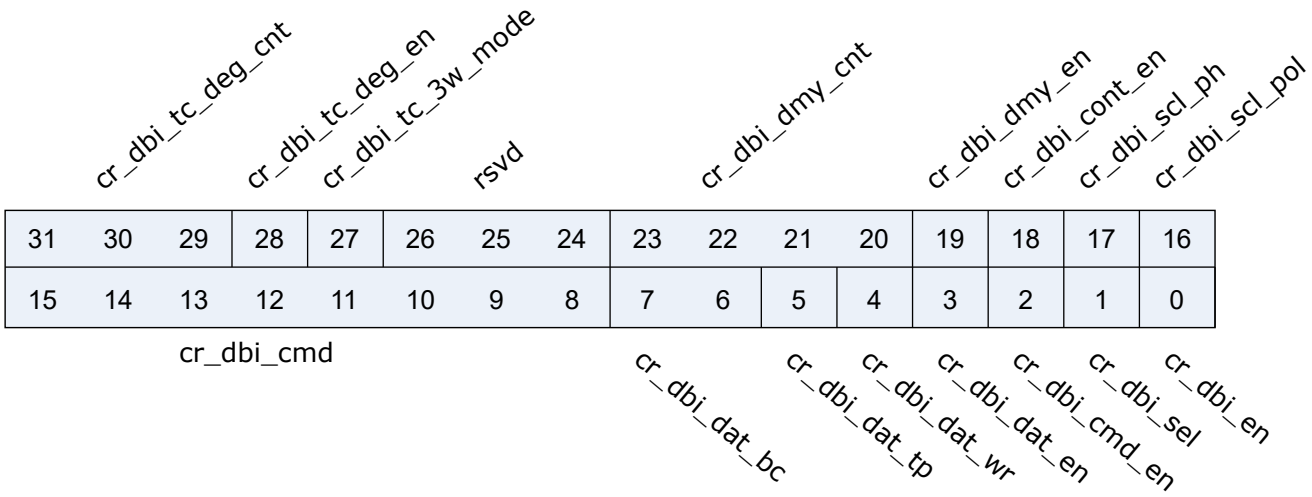
DBI TX supports the DMA transfer mode, which requires to set the threshold of TX FIFO by configuring the <TFITH> bit in the register DBI_FIFO_CONFIG_1. When this mode is enabled, if TFICNT is greater than TFITH, the DMA TX request will be triggered. After DMA is configured, when receiving this request, DMA will transfer data from memory to TX FIFO as configured.

10.4 Register description

Name	Description
dbi_config	
dbi_int_sts	
dbi_bus_busy	
dbi_pix_cnt	
dbi_prd	
dbi_wdata	
dbi_rdata	
dbi_fifo_config_0	
dbi_fifo_config_1	
dbi_fifo_wdata	

10.4.1 dbi_config

Address：0x3001b000

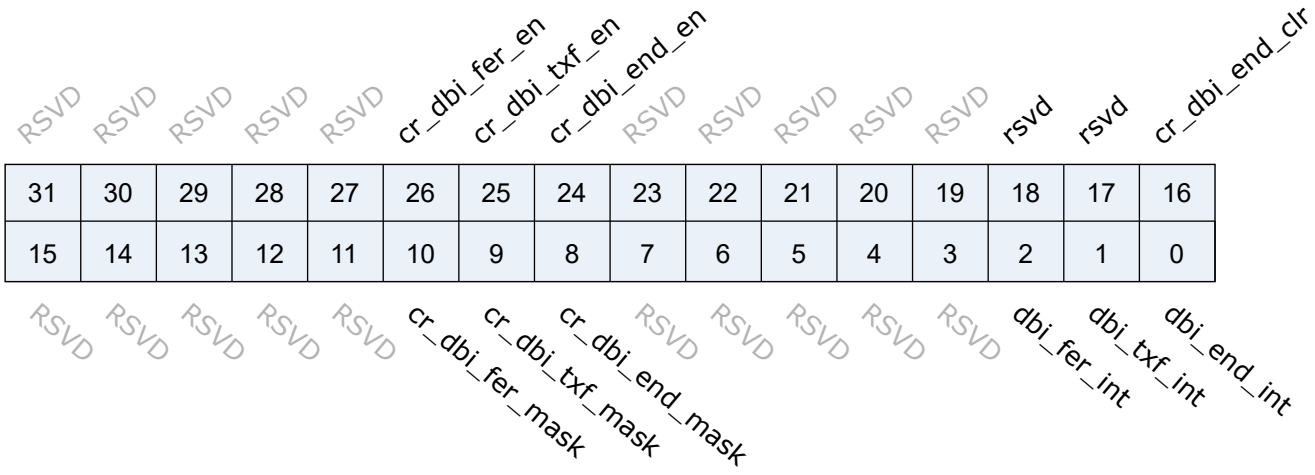


Bits	Name	Type	Reset	Description
31:29	cr_dbi_tc_deg_cnt	r/w	3'd0	De-glitch function cycle count
28	cr_dbi_tc_deg_en	r/w	1'b0	Enable signal of all input de-glitch function
27	cr_dbi_tc_3w_mode	r/w	1'b0	DBI Type C 3-wire mode enable 1'b0: 4-wire mode is enabled 1'b1: 3-wire mode is enabled
26:24	rsvd	rsvd	3'h0	
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
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
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_cmd	r/w	8'h2C	DBI Command

Bits	Name	Type	Reset	Description
7:6	cr_dbi_dat_bc	r/w	2'd0	Data byte count of normal data (pixel data count is determined by cr_dbi_pix_cnt)
5	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
4	cr_dbi_dat_wr	r/w	1'b1	Data phase Read/Write select 1'b0: Read data 1'b1: Write data
3	cr_dbi_dat_en	r/w	1'b1	Data enable signal 1'b0: Data phase disabled 1'b1: Data phase enabled
2	cr_dbi_cmd_en	r/w	1'b1	Command enable signal 1'b0: No command phase 1'b1: Command will be sent
1	cr_dbi_sel	r/w	1'b0	Select signal of DBI Type B or C 1'b0: DBI Type B 1'b1: DBI Type C
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

10.4.2 dbi_int_sts

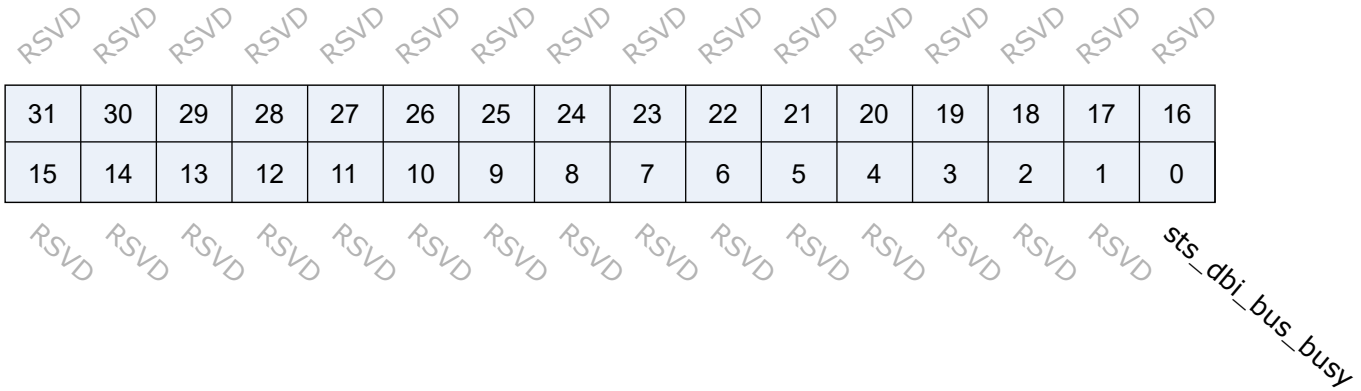
Address：0x3001b004



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

10.4.3 dbi_bus_busy

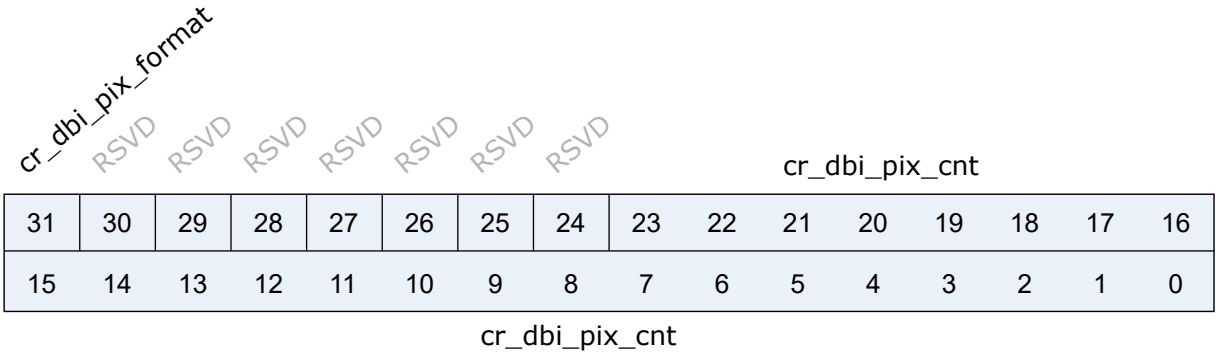
Address：0x3001b008



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	sts_dbi_bus_busy	r	1'b0	Indicator of dbi bus busy

10.4.4 dbi_pix_cnt

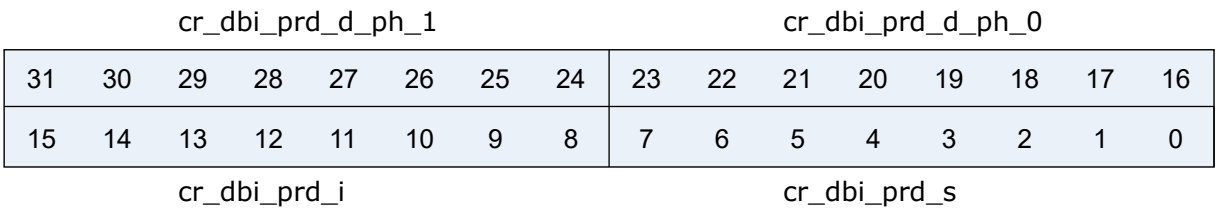
Address：0x3001b00c



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

10.4.5 dbi_prd

Address：0x3001b010



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)

Bits	Name	Type	Reset	Description
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)

10.4.6 dbi_wdata

Address：0x3001b018

cr_dbi_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

cr_dbi_wdata

Bits	Name	Type	Reset	Description
31:0	cr_dbi_wdata	r/w	32'h0	Data to be written into display IC using write command

10.4.7 dbi_rdata

Address：0x3001b01c

sts_dbi_rdata

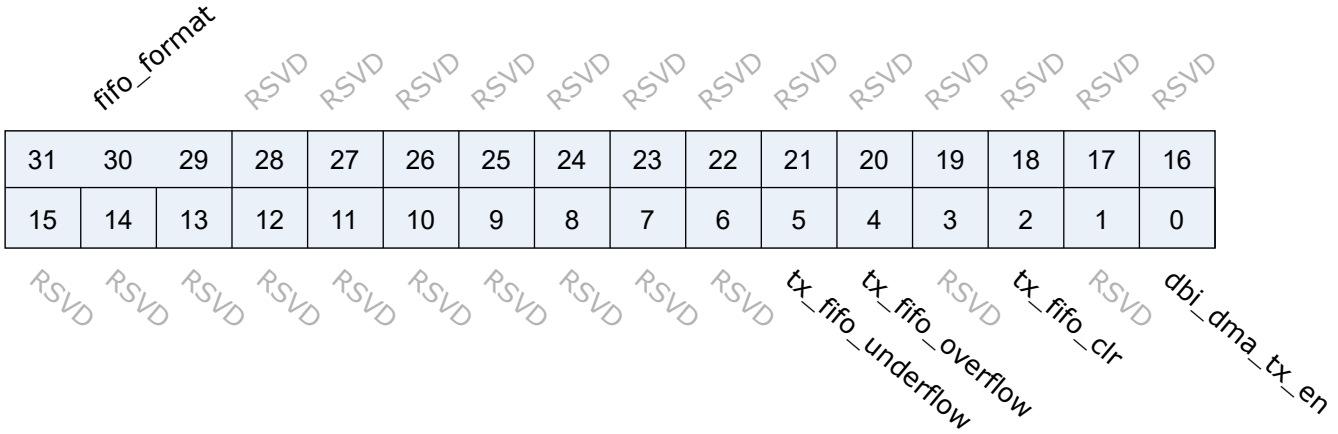
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

sts_dbi_rdata

Bits	Name	Type	Reset	Description
31:0	sts_dbi_rdata	r	32'h0	Data read from display IC using read command

10.4.8 dbi_fifo_config_0

Address：0x3001b080



Bits	Name	Type	Reset	Description
31:29	fifo_format	r/w	3'd0	FIFO 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[7:0], B[7:0], G[7:0], R[7:0] 3'd5: B[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]
28:6	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

10.4.9 dbi_fifo_config_1

Address：0x3001b084

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD tx_fifo_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

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD tx_fifo_cnt

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

10.4.10 dbi_fifo_wdata

Address：0x3001b088

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)

11 DPI

11.1 Overview

The Display Pixel Interface (DPI), also known as RGB interface, is an interface protocol defined by MIPI Alliance for communicating with display screens. This module has 16-bit parallel data lines, with the system memory acting as the display memory, and drives the display directly by continuously outputting data and synchronization signals.

11.2 Features

- Meets MIPI® Alliance standard
- Supports single buffer and ping-pong buffer serving as the display memory
- Supports hardware and software control for ping-pong operation
- Supports the Test Mode
- Programmable timing parameters
- Supported input pixel format (buffer data format):
  - YUV422 (interleaved)
  - RGB888
  - RGB565
  - RGBA8888
  - YUV420 (planar)
- Supports output pixel format of RGB565
- One programmable interrupt
- Use with OSD

11.3 Functional Description

The block diagram of DPI Module is shown as follows.

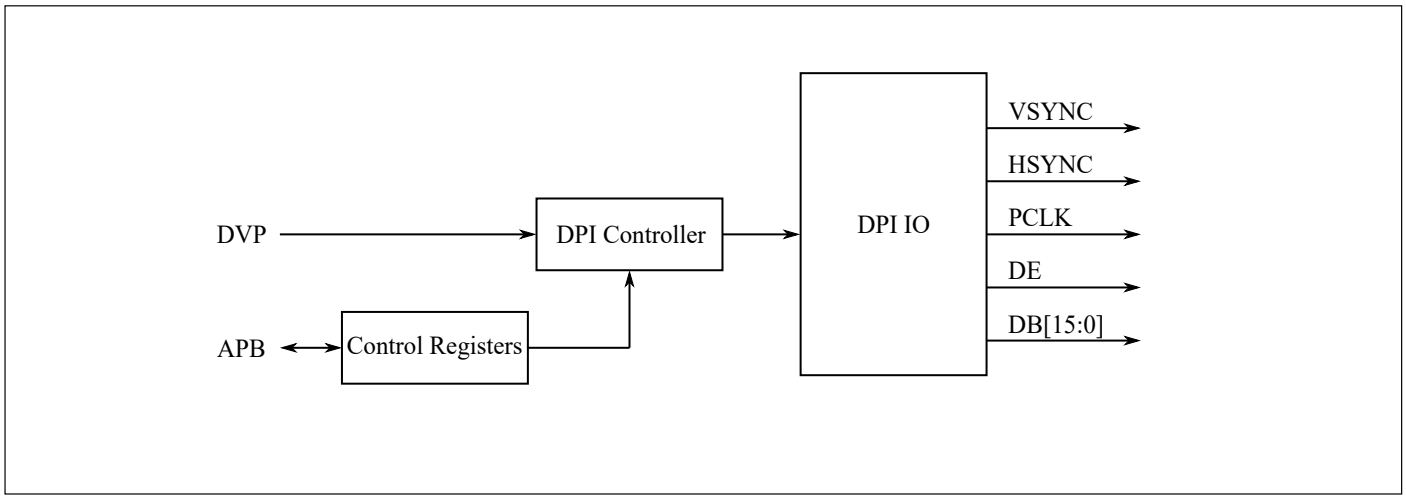


Fig. 11.1: Block Diagram of DPI

11.3.1 DPI Timing

The 16-bit DPI interface timing is shown as follows:

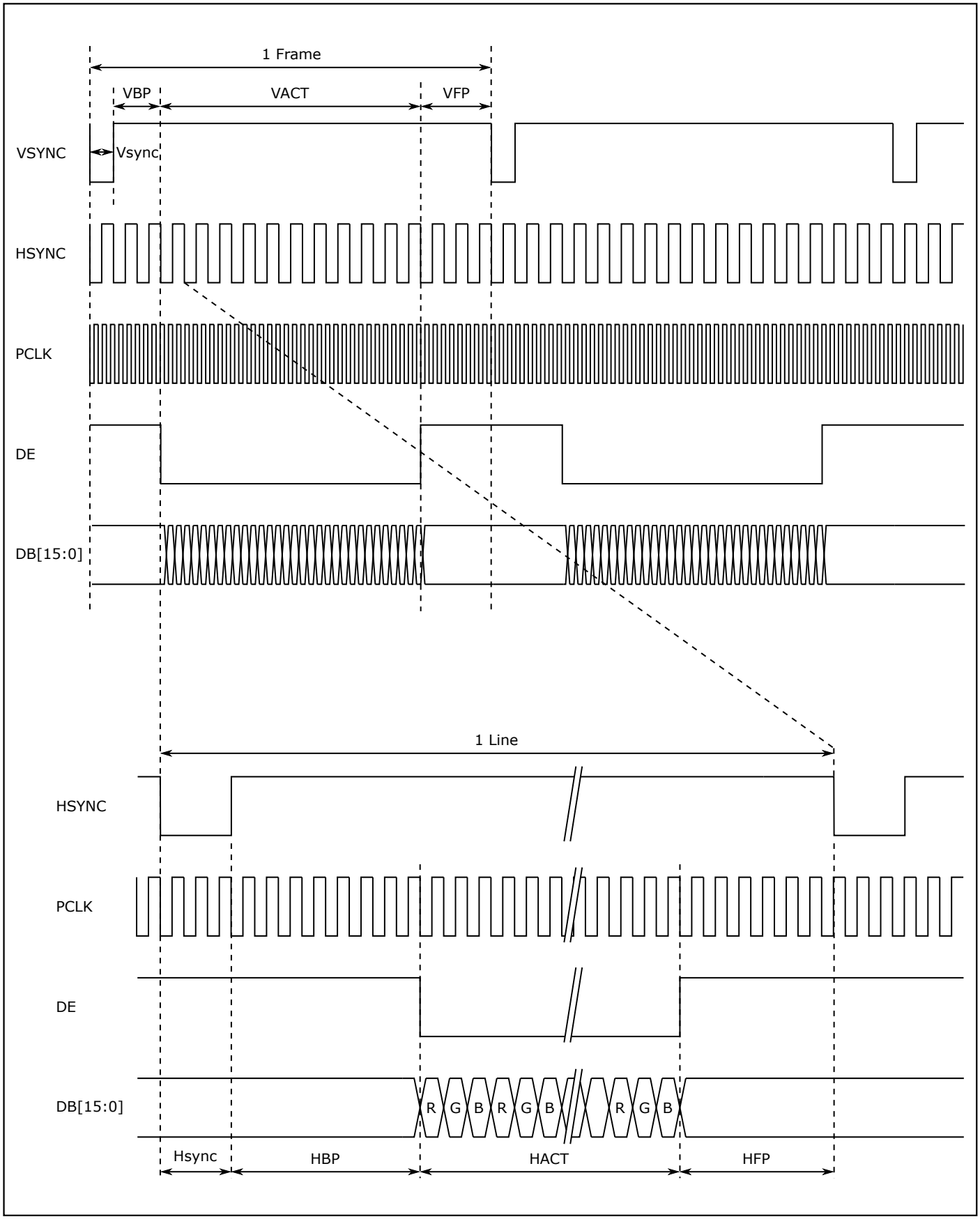


Fig. 11.2: DPI Timing

- Vertical Sync (VSYNC): Vertically synchronized timing signal, used to indicate the beginning of each displayed image frame
- Horizontal Sync (HSYNC): Horizontally synchronized timing signal, used to indicate the beginning of each horizontal pixel line
- Pixel Clock (PCLK): Pixel clock, continuously generated; VSYNC, HSYNC, DE, and DB [15:0] will be sampled at the rising edge of PCLK
- Data Enable (DE): Data enable signal, used to indicate valid pixel data
- Data Bit (DB) [15:0]: Parallel data bits –16 bits in total, used to transfer pixel data.

11.3.2 Programmable timing parameters

The timing parameters of DPI are shown as follows:

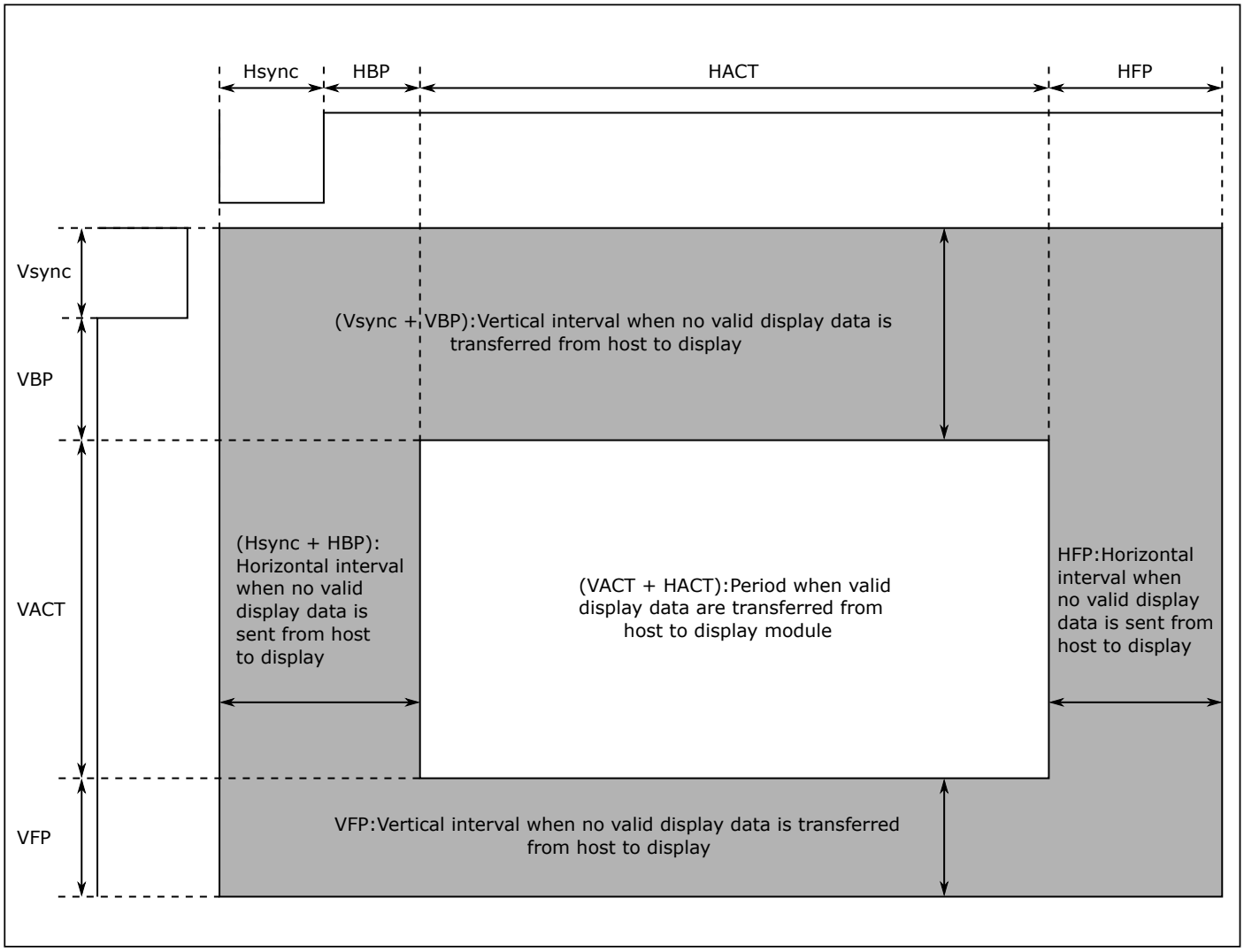


Fig. 11.3: Timing Parameters

The vertical period (one frame) is equal to the sum of Vsync, VBP, VACT, and VFP. The horizontal period (one line) is equal to the sum of Hsync, HBP, HACT, and HFP. The programmable ranges of the timing parameters are shown in the following table:

Parameters	Symbols	Min.	Step	Max.	Unit
Horizontal Synchronization	Hsync	1	1	255	PCLK Cycle
Horizontal Back Porch	HBP	1	1	-	PCLK Cycle
Horizontal Active	HACT	176	1	800	PCLK Cycle
Horizontal Front Porch	HFP	1	1	255	PCLK Cycle
Vertical Synchronization	Vsync	1	1	255	Line
Vertical Back Porch	VBP	1	1	-	Line
Vertical Active	VACT	208	1	480	Line
Vertical Front Porch	VFP	1	1	255	Line

Fig. 11.4: Ranges of Timing Parameters

11.3.3 Single Buffer

The DPI Module can use one memory with a size of "display resolution × number of bytes per pixel" as the display memory. When it starts working, DPI transfers the data in this memory to the Display Module repeatedly and continuously. If changes are made to the data in this memory, the changes may take effect on the current or next frame, depending on whether the changed data is located in memory after or before the location in memory of the data currently being transferred by the DPI.

11.3.4 Ping-Pong Buffer

The DPI Module can use two memories with each size of "display resolution × number of bytes per pixel" as the display memory. DPI transfers the data in one memory to the Display Module repeatedly and continuously. The control mode for switching the current memory for data transfer to the other memory can be set to hardware or software control. When it is set to hardware control, the data for DPI is provided by the Cam Module (see Cam Module section); DPI switches the current memory in use to another memory after the Cam Module has written a complete frame of data to the memory not involved in transfer, and the data in the current memory is completely transferred. When it is set to software control, the memory currently used by DPI for transfer is set by <SWAP_IDX_SWV>, with "0" and "1" corresponding to two memory blocks respectively.

11.3.5 Test Mode

In Test Mode, DPI can output images with the same pixel color in the same row and a gradient color between different rows. The Test Mode requires three parameters, namely the minimum value of pixel data VAL_MIN (16 bit), the maximum value of pixel data VAL_MAX (16 bit), and the step size STEP (8 bit). DPI uses VAL_MIN as the value of each pixel in the 1st row, (VAL_MIN+STEP) as the one in the 2nd row, and (VAL_MIN+STEP*2) as the one in the 3rd row…and so on, and returns to VAL_MIN when the pixel value of the Nth row (VAL_MIN+STEP*(N1)) is greater than VAL_MAX. This cycle repeats until the number of output rows reaches the set value. Taking VAL_MIN=0x00E0, VAL_MAX=0x012F, and STEP=0x10 as an example, the display in the Test Mode is shown as follows:

VAL_MIN=0x00E0, VAL_MAX=0x012F, STEP=0x10

0x 00E0 00E0 00E0 00E0 00E0 … 00E0 00E0	Line 1
0x 00F0 00F0 00F0 00F0 00F0 … 00F0 00F0	Line 2
0x 0100 0100 0100 0100 0100 … 0100 0100	Line 3
0x 0110 0110 0110 0110 0110 … 0110 0110	Line 4
0x 0120 0120 0120 0120 0120 … 0120 0120	Line 5
0x 00E0 00E0 00E0 00E0 00E0 … 00E0 00E0	Line 6
0x 00F0 00F0 00F0 00F0 00F0 … 00F0 00F0	Line 7
0x 0100 0100 0100 0100 0100 … 0100 0100	Line 8
0x 0110 0110 0110 0110 0110 … 0110 0110	Line 9
0x 0120 0120 0120 0120 0120 … 0120 0120	Line 10
0x 00E0 00E0 00E0 00E0 00E0 … 00E0 00E0	Line 11
0x 00F0 00F0 00F0 00F0 00F0 … 00F0 00F0	Line 12
0x 0100 0100 0100 0100 0100 … 0100 0100	Line 13
⋮	⋮

Display(RGB565)

Fig. 11.5: Test Mode

11.3.6 Input Pixel Format

DPI supports input in multiple pixel formats. Internally, it will convert these formats to RGB888 uniformly and then convert RGB888 to RGB565 for sending out.

11.3.6.1 YUV422 (Interleaved)

The processing of input data in YUV422 (interleaved) format is shown as follows:

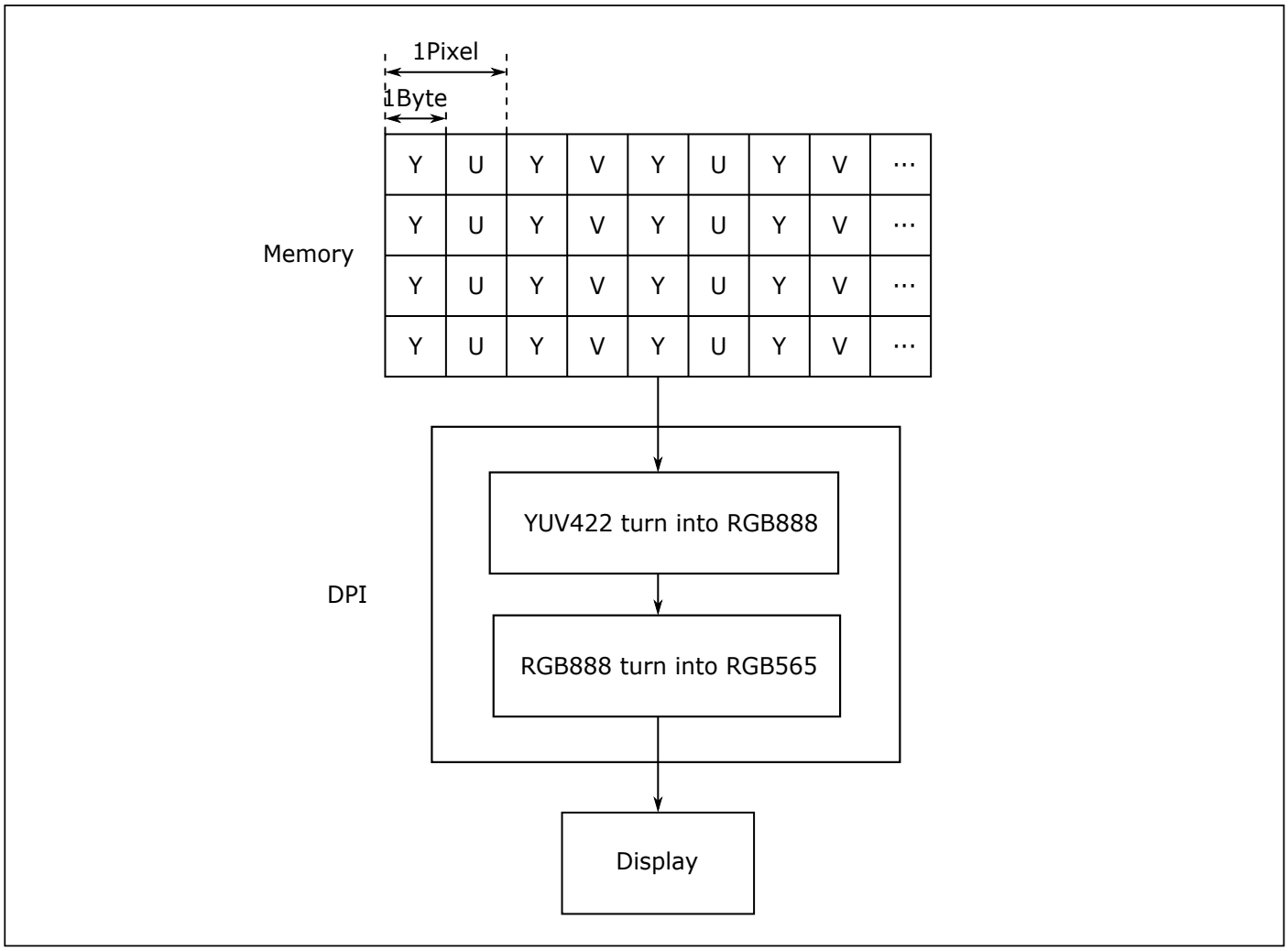


Fig. 11.6: YUV422 Processing Process

11.3.6.2 RGB888

The processing of input data in RGB888 format is shown as follows:

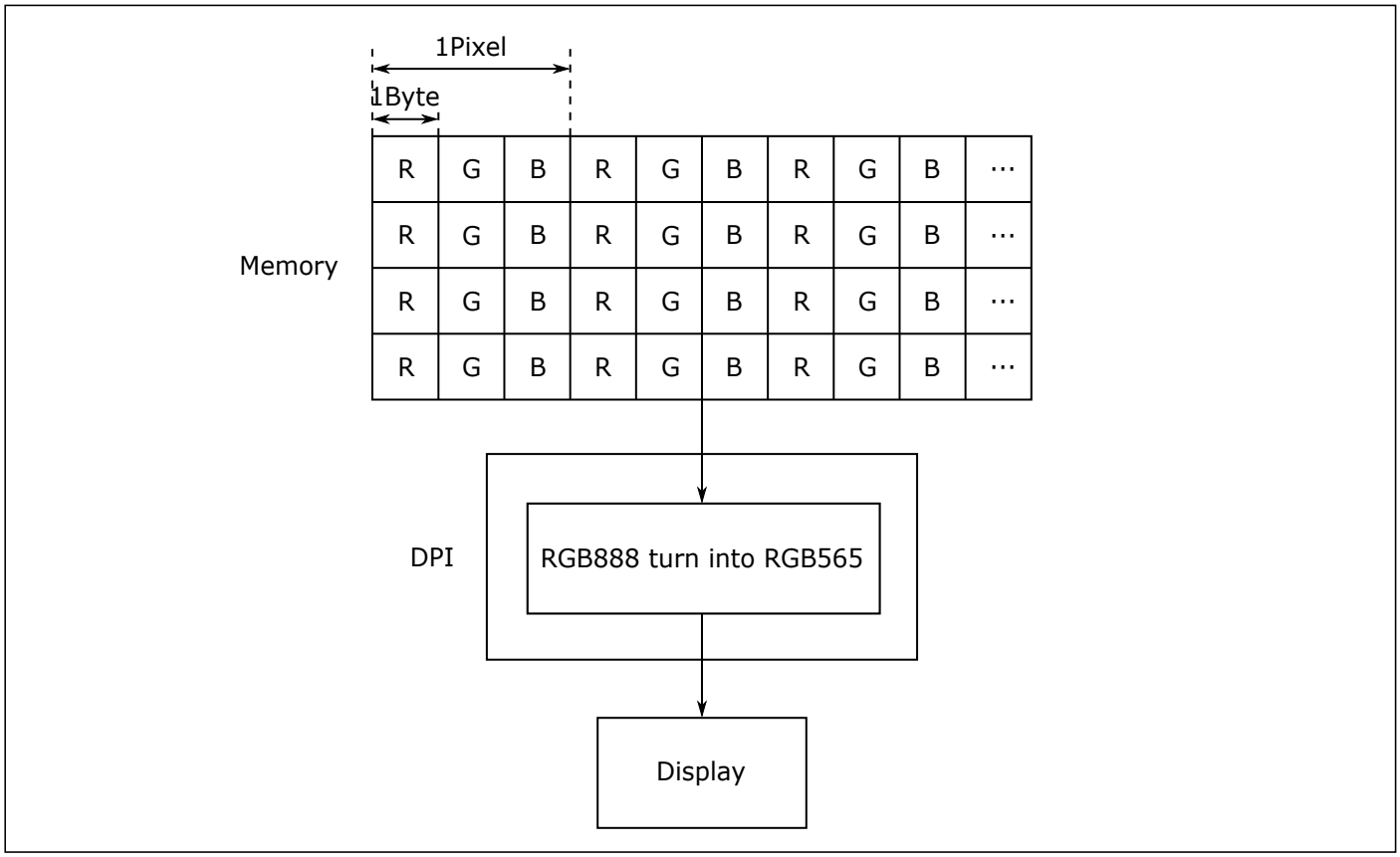


Fig. 11.7: RGB888 Processing Process

11.3.6.3 RGB565

The processing of input data in RGB565 format is shown as follows:

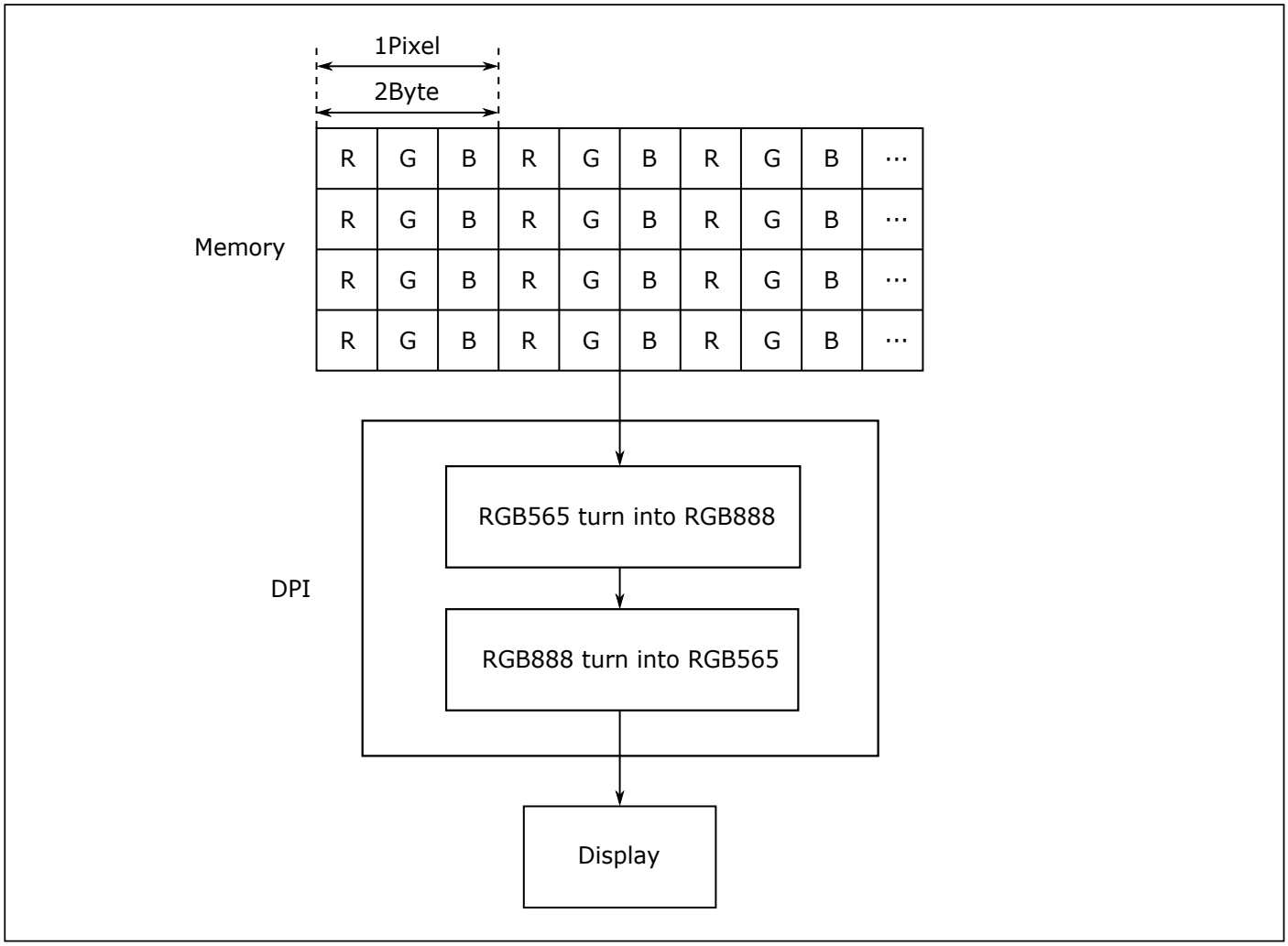


Fig. 11.8: RGB565 Processing Process

11.3.6.4 RGBA8888

The processing of input data in RGBA8888 format is shown as follows:

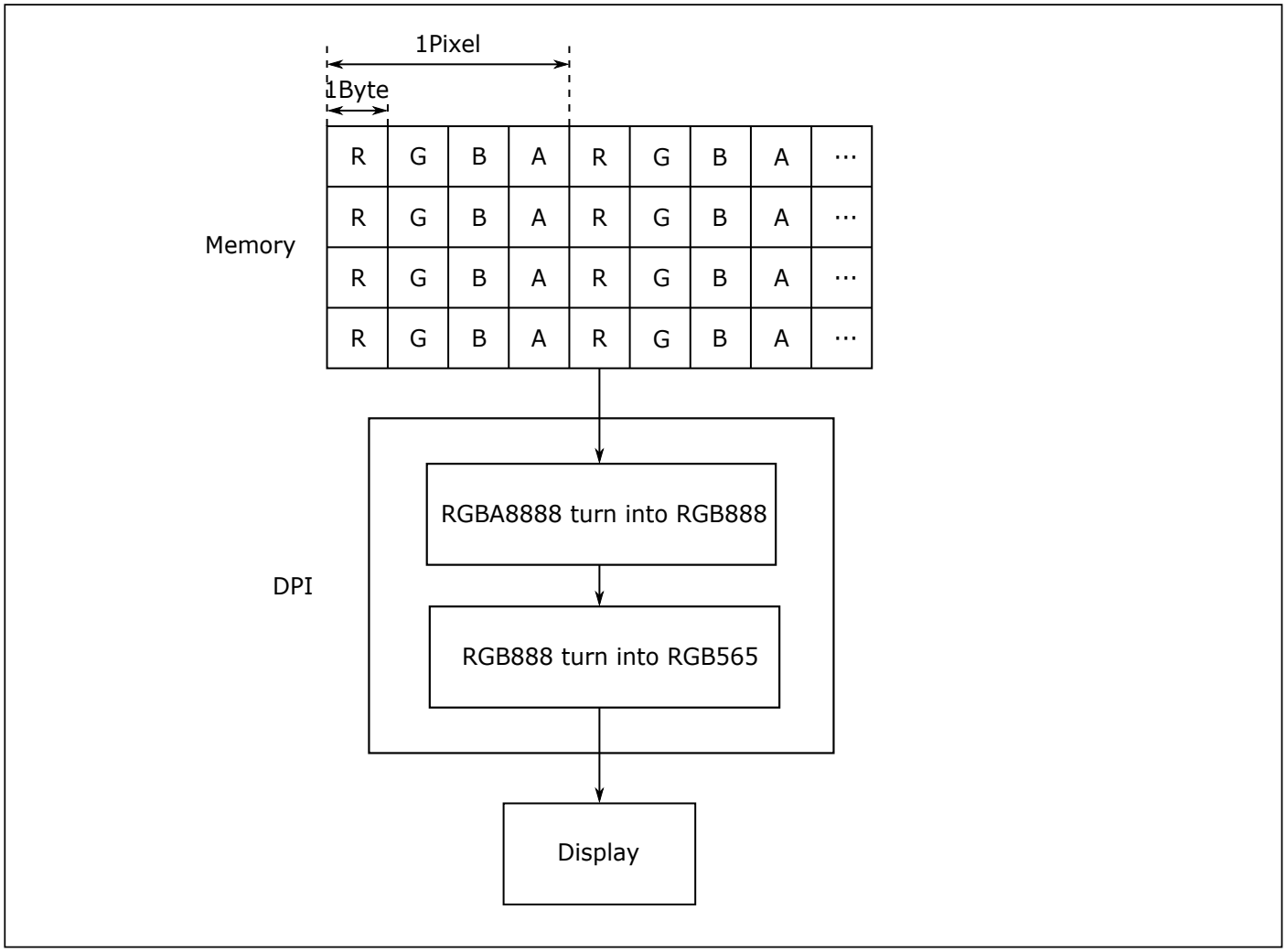


Fig. 11.9: RGBA8888 Processing Process

11.3.6.5 YUV420 (Planar)

The processing of input data in YUV420 (planar) format is shown as follows:

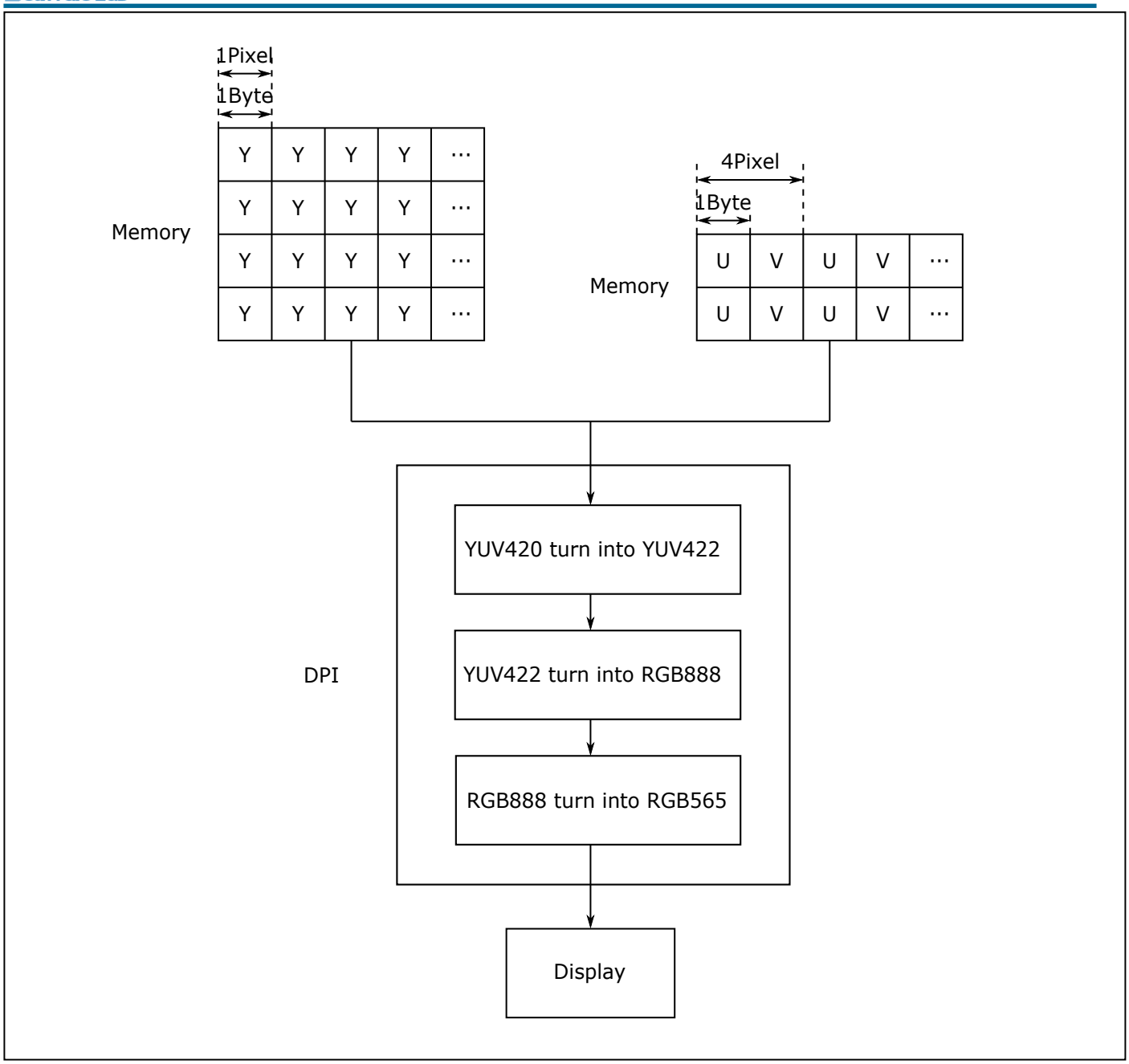


Fig. 11.10: YUV420 Processing Process

11.3.7 Programmable Interrupt

The DPI has a programmable interrupt. In configuration, the trigger source of the interrupt can be set to the input Vsync or output Vsync of the DPI, and the hopping edge can be the rising or falling edge. For example, if the interrupt is configured to be triggered on the rising edge of the input Vsync, DPI will generate an interrupt request on each Vsync rising edge of its input module. If that is configured to be triggered on the falling edge of the output Vsync, DPI generates an interrupt request on each Vsync falling edge of its output module.

11.3.8 Use with OSD

The input data of DSI can be configured to be processed by the OSD first. See the OSD Module section for functional description.

12 DSI

12.1 Overview

The Display Serial Interface (DSI) is an interface protocol defined by MIPI Alliance for communicating with display screens. It sends pixels or commands to peripherals and can read status or pixel information from peripherals. All data lanes of DSI are unidirectional in High-Speed Mode, while in Low-Speed Mode only the first data lane is bidirectional and other lanes are unidirectional. The clock lane is dedicated to transmitting synchronous clock signals during high-speed data transmission. In addition, one master side allows communication with multiple slave sides at the same time. This chip has a built-in DSI controller and DPHY, which enable communication with DSI-compatible displays so that data can be displayed on the screen.

12.2 Features

- Meets MIPI® Alliance standard
- MIPI® DPHY interface with maximum speed of 800 Mbps
- One Clock Lane, up to 4 Data Lanes
- Four sequences for Data Lanes
- Supports Ultra Low Power Mode and High Speed Mode
- Data Lane 0 supports bidirectional communication and Escape Mode
- Supports LPDT/ULPS/Trigger in the Escape Mode
- Supports short and long packet sending in the LPDT Mode
- Supports error correction code (ECC) and check function
- Supports data length error and data overflow error interrupt
- Supports Command Mode and Video Mode
- Supports Event and Pulse Modes in the Video Mode

- Data formats supported by the display interface:
  - YUV422, 8 bit
  - RGB565
  - RGB666, loosely packed
  - RGB888
- Rich and diverse interrupt mechanisms and status query
- Supports DMA transmission in the LPDT Mode
- Use with OSD

12.3 Functional Description

The block diagram of the DSI Module is shown as follows:

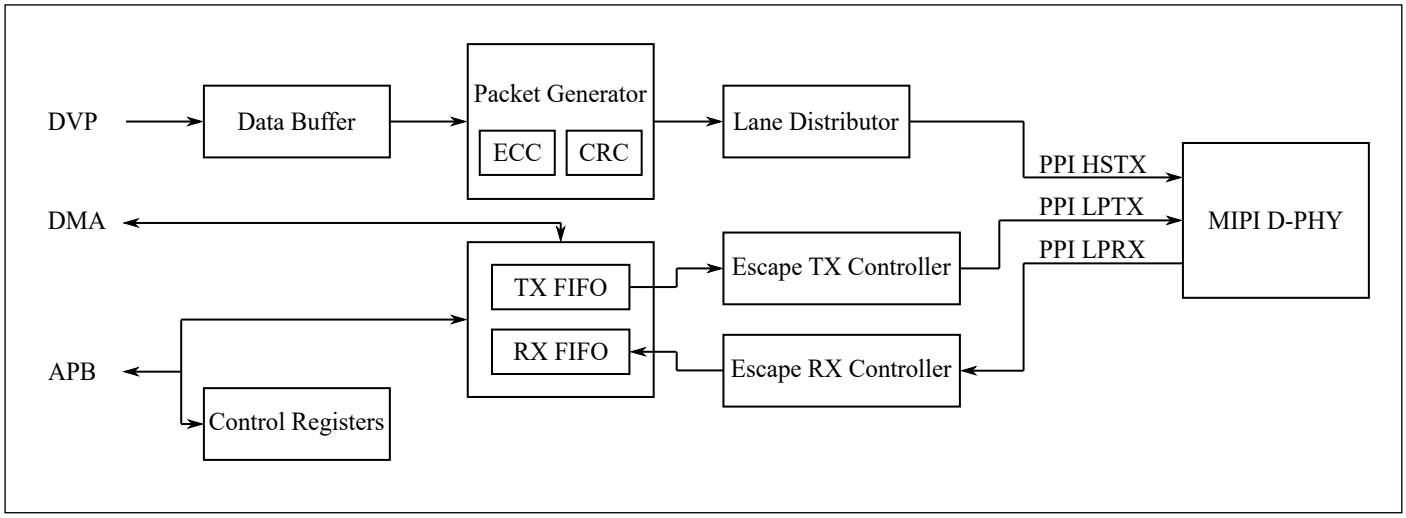


Fig. 12.1: Block Diagram of DSI

12.3.1 DSI Protocol Layering

The DSI protocol is divided into 4 layers from bottom to top, namely, physical layer, lane management layer, protocol layer, and application layer, as shown below:

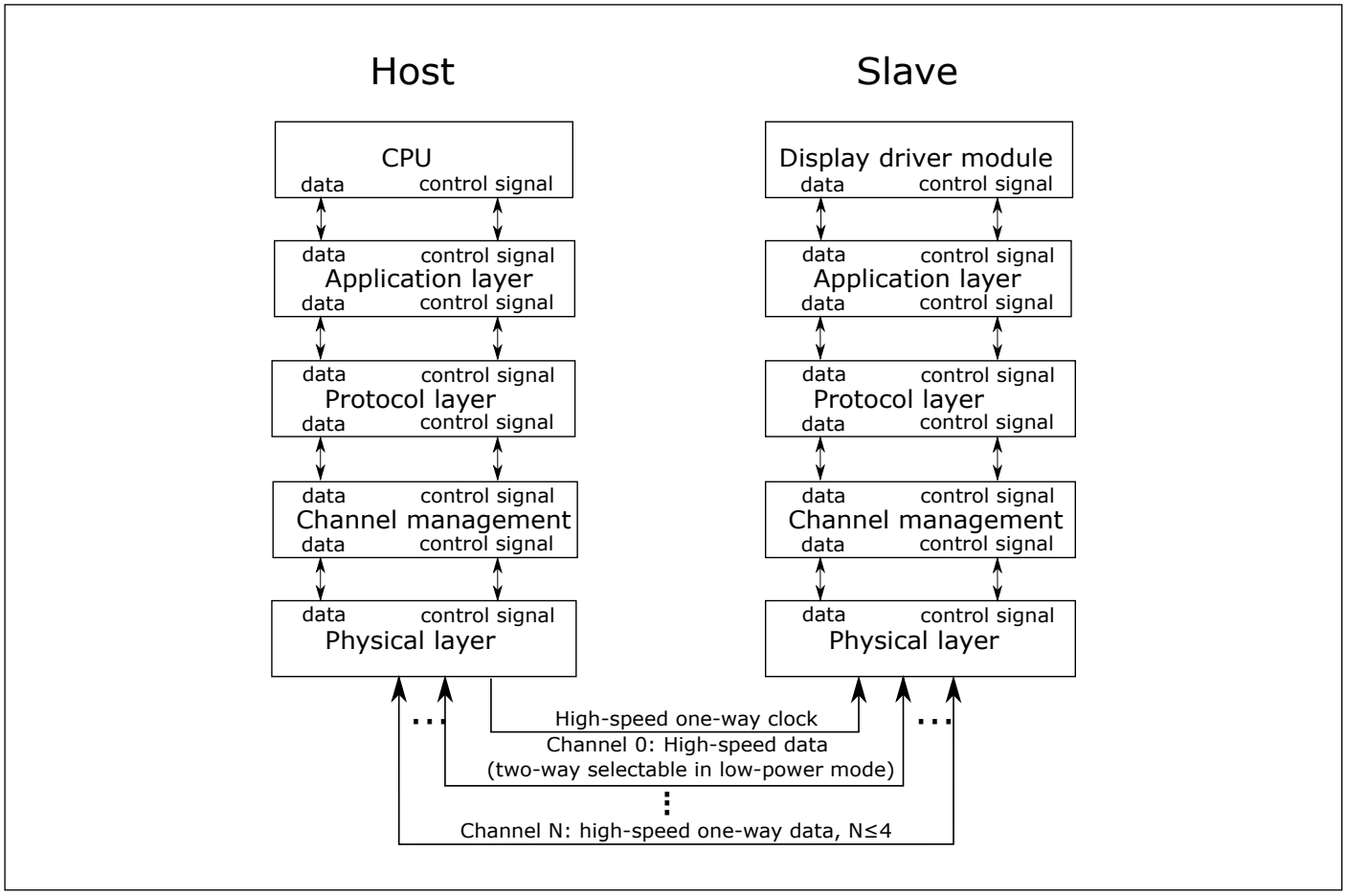


Fig. 12.2: Protocol Layering

Physical layer: It specifies the transmission medium (electrical conductor), the input/output circuit, and the clock mechanism for capturing “1” and “0” from the serial bit stream. In addition, it also specifies the signaling mechanism of start of transmission (SoT) and end of transmission (EoT) and other “out-of-band” information that can be transmitted between transmitting and receiving PHY.

Lane management layer: DSI can expand lanes to improve performance. The number of data lanes can be 1, 2, 3 or 4 depending on the bandwidth requirements of the application. The lane management layer aims to transfer the to-be-sent data from the transmitter (TX) end to the corresponding data lanes by groups according to the lane order, collect the data from each lane at the receiver (RX) end, and then merge them into a recombined data stream, thus recovering the original stream sequence.

Protocol layer: The DSI protocol defines packet formats, including short and long packets. The protocol layer aims to pack packets at the TX end according to the type and content of data, add the ECC and CRC codes, and transmit them to the lane management layer. At the RX end, it checks and corrects the received packet based on the ECC and CRC codes, decodes the packet header and data content, and transmits them to the application layer.

Application layer: Depending on the needs of application modules, at the TX end, it initially encodes the to-be-sent commands and data and converts them into the format specified by DSI; at the RX end, it restores the received data to the data format and timing supported by application modules.

12.3.2 Physical Layer

According to the DPHY protocol, synchronous connection is used between the master and the slave in the physical layer. The clock lane is used to transmit high-speed clocks, and one or more data lanes are used to transmit low-power or high-speed data signals. Each lane uses two interconnected lines to connect the master and the slave, and supports both High-Speed (HS) and Low-Power (LP) Modes. In the HS Mode, TX simultaneously drives two interconnected lines of the lane to output a low-swing differential signal, such as 200 mV. In the LP Mode, TX drives the interconnected lines respectively, to output a single-ended signal with a relatively large swing respectively, such as 1.2 V. The levels of interconnected lines in the two modes are shown as follows:

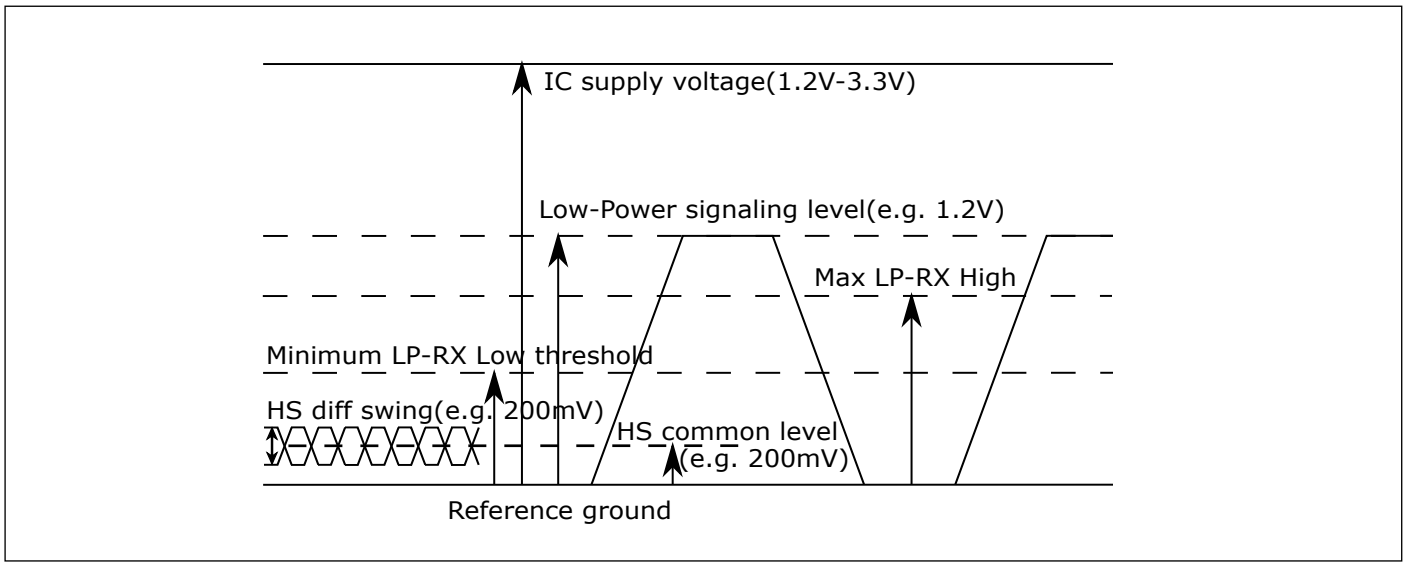


Fig. 12.3: Levels in HS and LP Modes

The two ends of the interconnected line are a driving unit and a receiving unit. The driving unit includes a differential transmitting module (HS_TX) and a low-power single-ended transmitting module (LP_TX), and the receiving unit includes a differential receiving module (HS_RX) and a low-power single-ended receiving module (LP_RX). HS_TX drives the interconnected line with differential signals, and there are two statuses on the high-speed lane: Differential0 and Differential1. LP_TX independently drives two interconnected lines, and there are four statuses on the lane: LP00, LP01, LP10, and LP11. The protocol clearly defines the line levels and configures the HS Mode, Control Mode, and Escape Mode, as shown below:

Lane Pair State Code	Line DC Voltage Levels		High Speed(HS)	Low Power	
	DATA_P	DATA_N	Burst Mode	Control Mode	Escape Mode
HS-0	Low(HS)	High(HS)	Differential-0	Note 1	Note 1
HS-1	High(HS)	Low(HS)	Differential-1	Note 1	Note 1
LP-00	Low(LP)	Low(LP)	Not Defined	Bridge	Space
LP-01	Low(LP)	High(LP)	Not Defined	HS-Request	Mark-0
LP-10	High(LP)	Low(LP)	Not Defined	LP-Request	Mark-1
LP-11	High(LP)	High(LP)	Not Defined	Stop	Note 2

Note:

1. When the lane is in high-speed mode, the low-power receiver (LP_Rx) of each lane will check the LP-00 status code;
2. If the low power receiver (LP-Rx) of each lane recognizes the LP-11 status code, the lane will return to control mode.

Fig. 12.4: Lane Status Code

12.3.3 Clock Lane

The clock lane can be driven to Low Power Mode (LPM), UltraLow Power Mode (ULPM), and High Speed Clock Mode (HSCM). The flow chart of switching between different power modes of the clock lane is as follows:

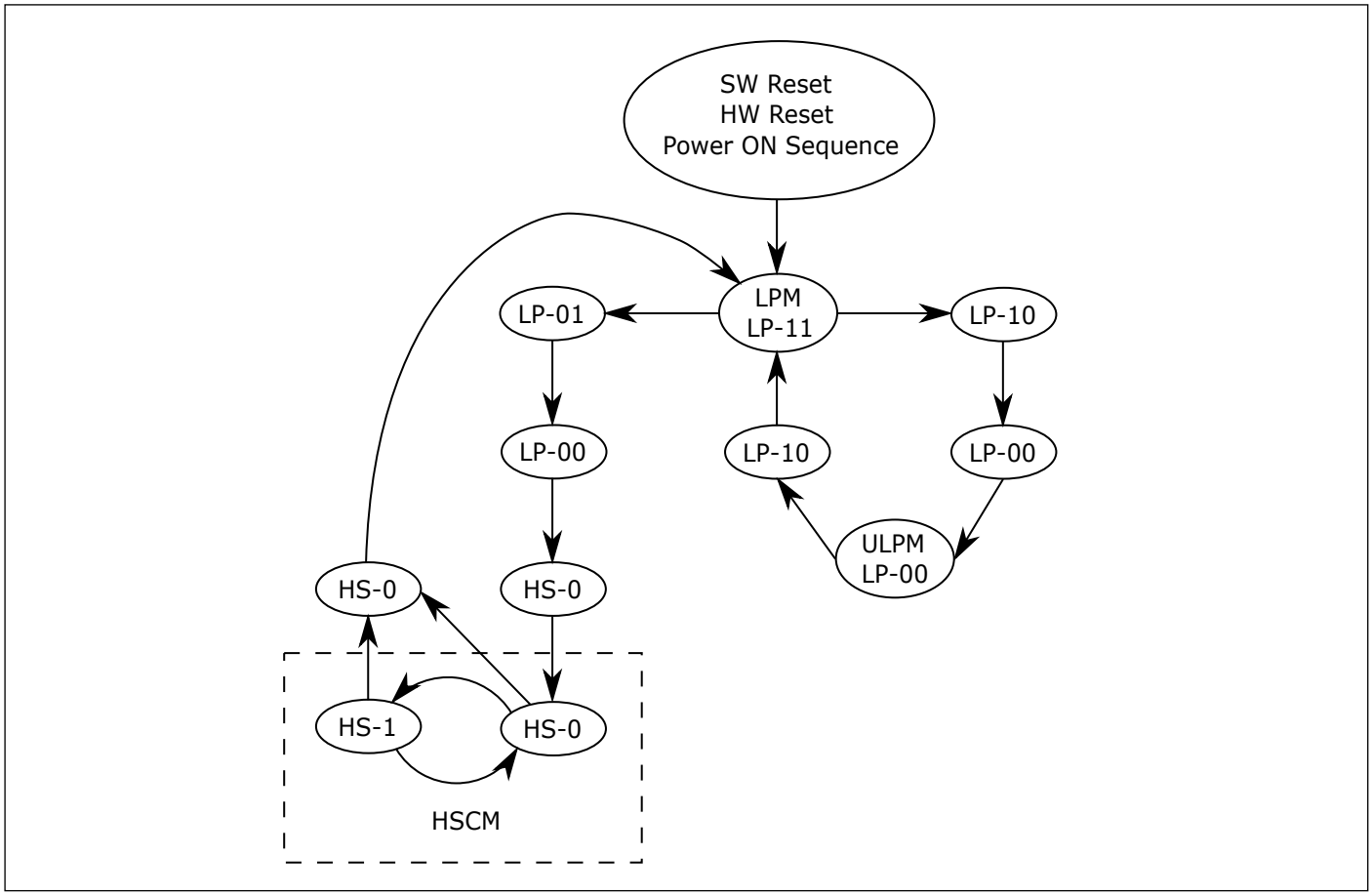


Fig. 12.5: Clock Lane Mode Switching

12.3.3.1 Low Power Mode (LPM, LP11)

Three ways for switching:

1. After the software reset, hardware reset and power-on sequence, the clock lane automatically changes to LP11 (LPM).
2. After leaving LP00 (ULPM), the clock lane changes to LP10 first and then to LP11 (LPM), as shown below:

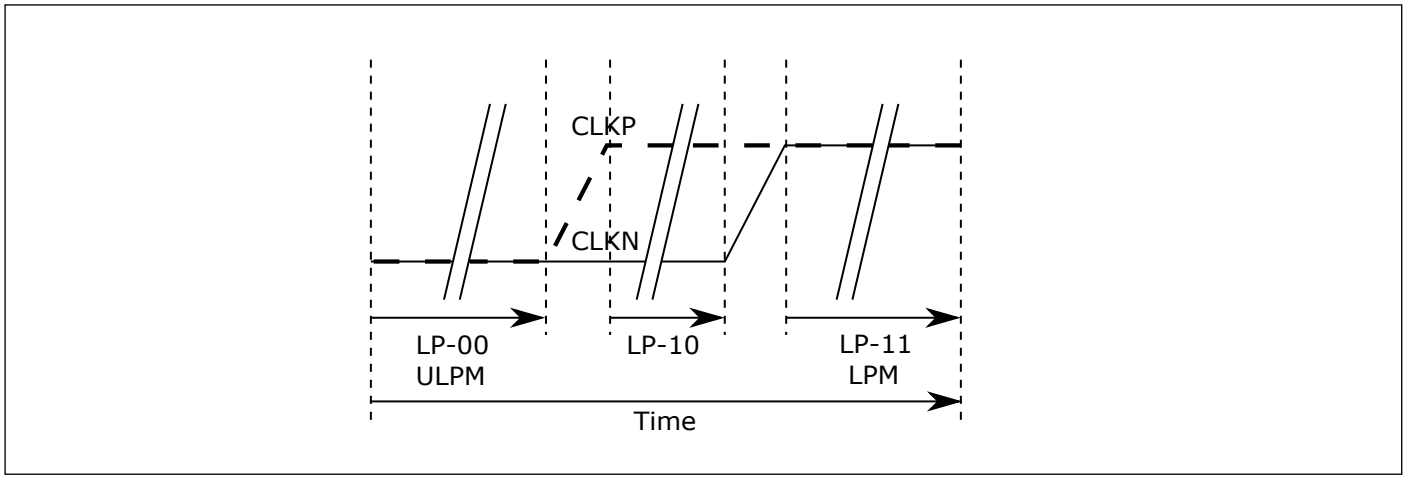


Fig. 12.6: Switching from ULPM to LPM

3. After leaving HS0 or HS1 (HSCM), the clock lane changes to HS0 first and then to LP11 (LPM), as shown below:

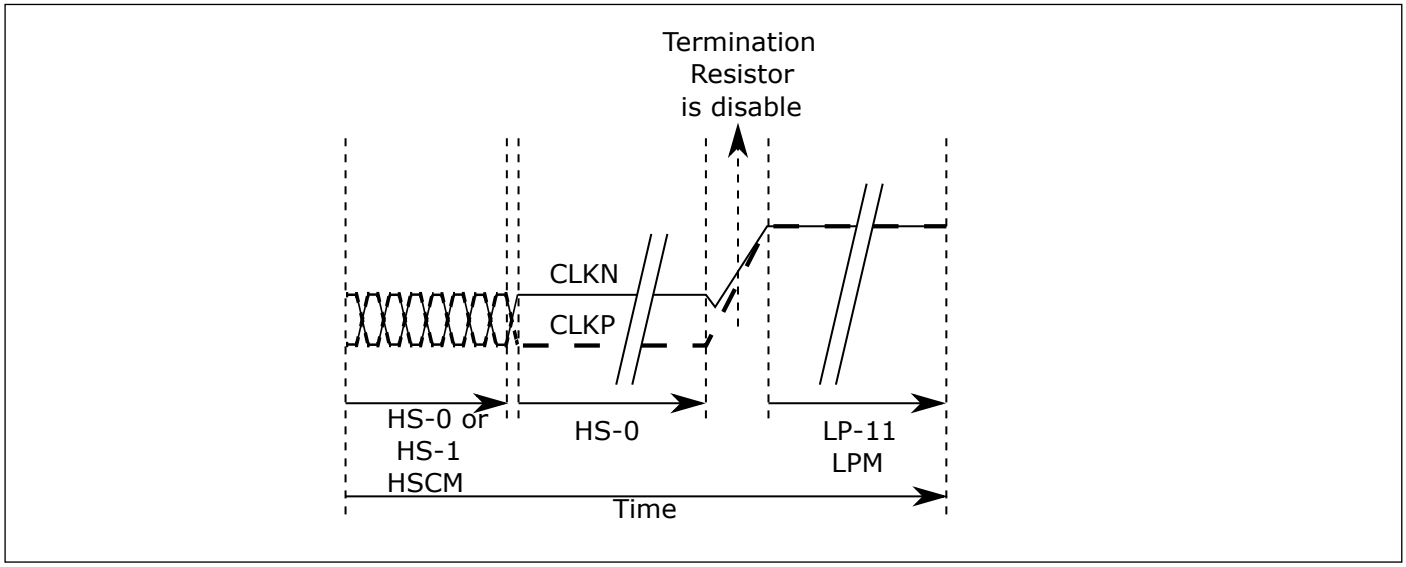


Fig. 12.7: Switching from HSCM to LPM

12.3.3.2 UltraLow Power Mode (ULPM, LP00)

One way for switching: The clock lane switches from LP11 (LPM) to LP10 first and then to LP00 (ULPM), as shown below:

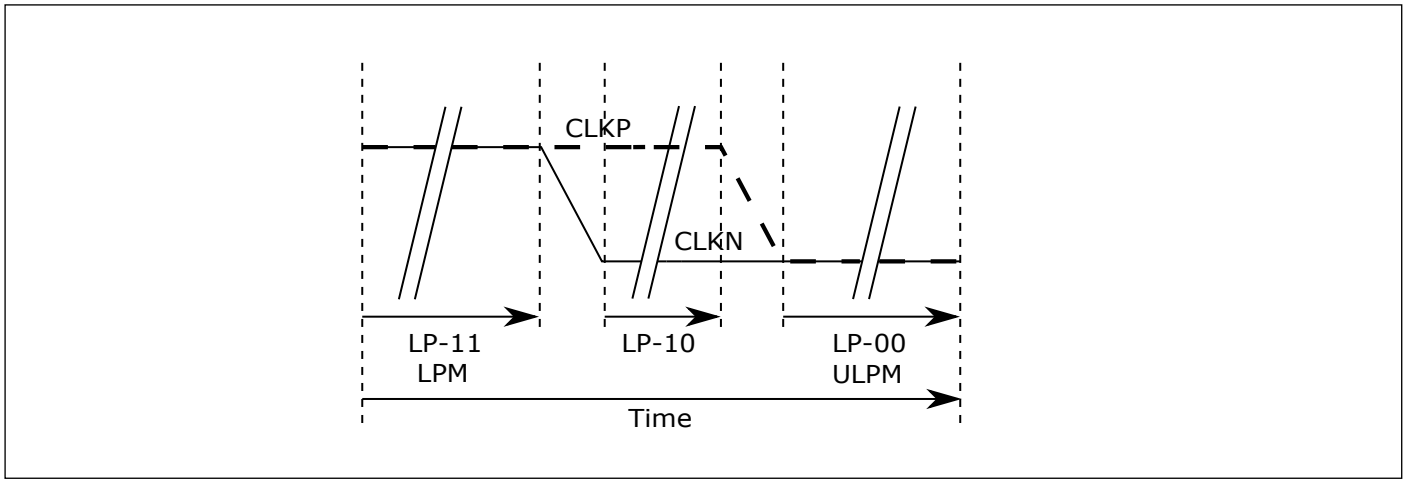


Fig. 12.8: Switching from LPM to ULPM

12.3.3.3 High Speed Clock Mode (HSCM, HS0/1)

One way for switching: The clock lane switches from LP11 (LPM) to LP01, LP00, HS0, and HS0/1 (HSCM) in sequence, as shown below:

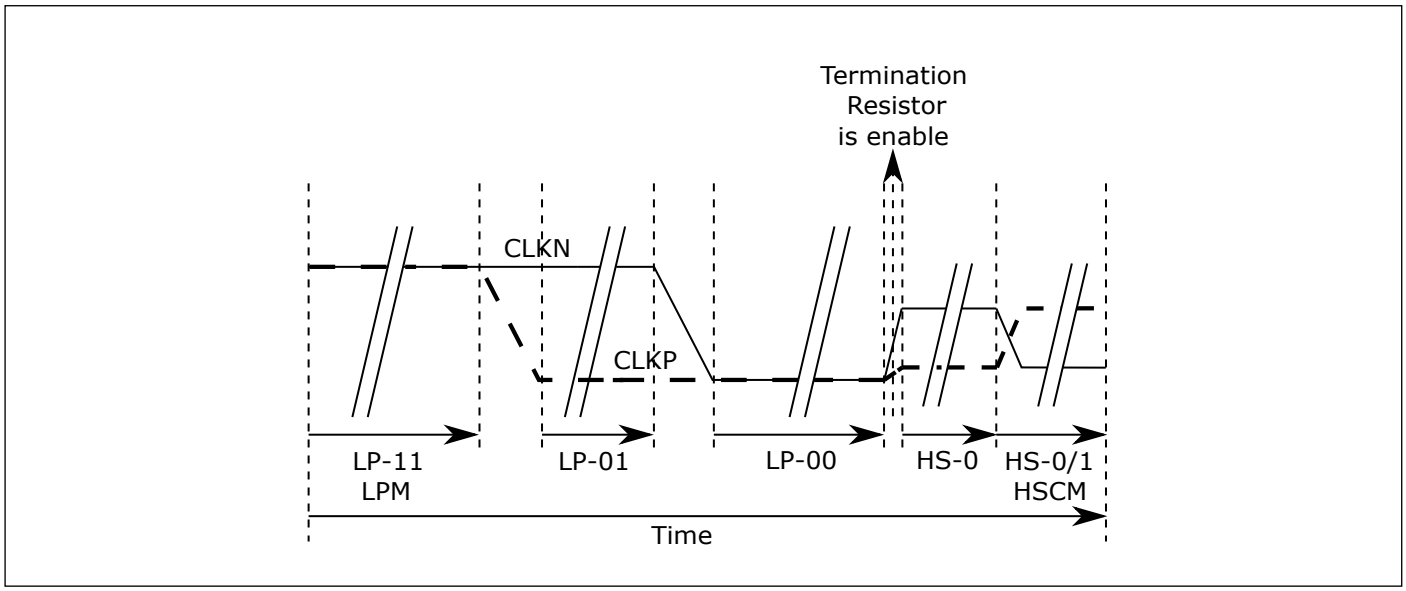


Fig. 12.9: Switching from LPM to HSCM

12.3.4 Data Lanes

All data lanes (D3P/N, D2P/N, D1P/N, and D0P/N) can be driven to HighSpeed Data Transmission (HSDT) Mode, but only data lane 0 (D0P/N) can switch to the Escape Mode and Bus TurnAround Request (BTA) Mode. The entering and leaving sequences of the three modes are shown as follows:

Mode	Entering Mode Sequence	Leaving Mode Sequence
Escape Mode	LP-11 -> LP-10 -> LP-00 -> LP-01 -> LP-00	LP-00 -> LP-10 -> LP-11(Mark-1)
High-Speed Data Transmission	LP-11 -> LP-01 -> LP-00 -> HS-0	(HS-0 or HS-1) -> LP-11
Bus Turnaround Request	LP-11 -> LP-10 -> LP-00 -> LP-10 -> LP-00	Hi-Z

Fig. 12.10: Entering and Leaving Sequences of Three Modes of Data Lanes

12.3.4.1 HighSpeed Data Transmission (HSDT)

When the clock lane of the MCU has switched to the HSCM Mode, the Display Module can switch to the HSDT Mode. The sequence of switching to the HSDT Mode is shown as follows:

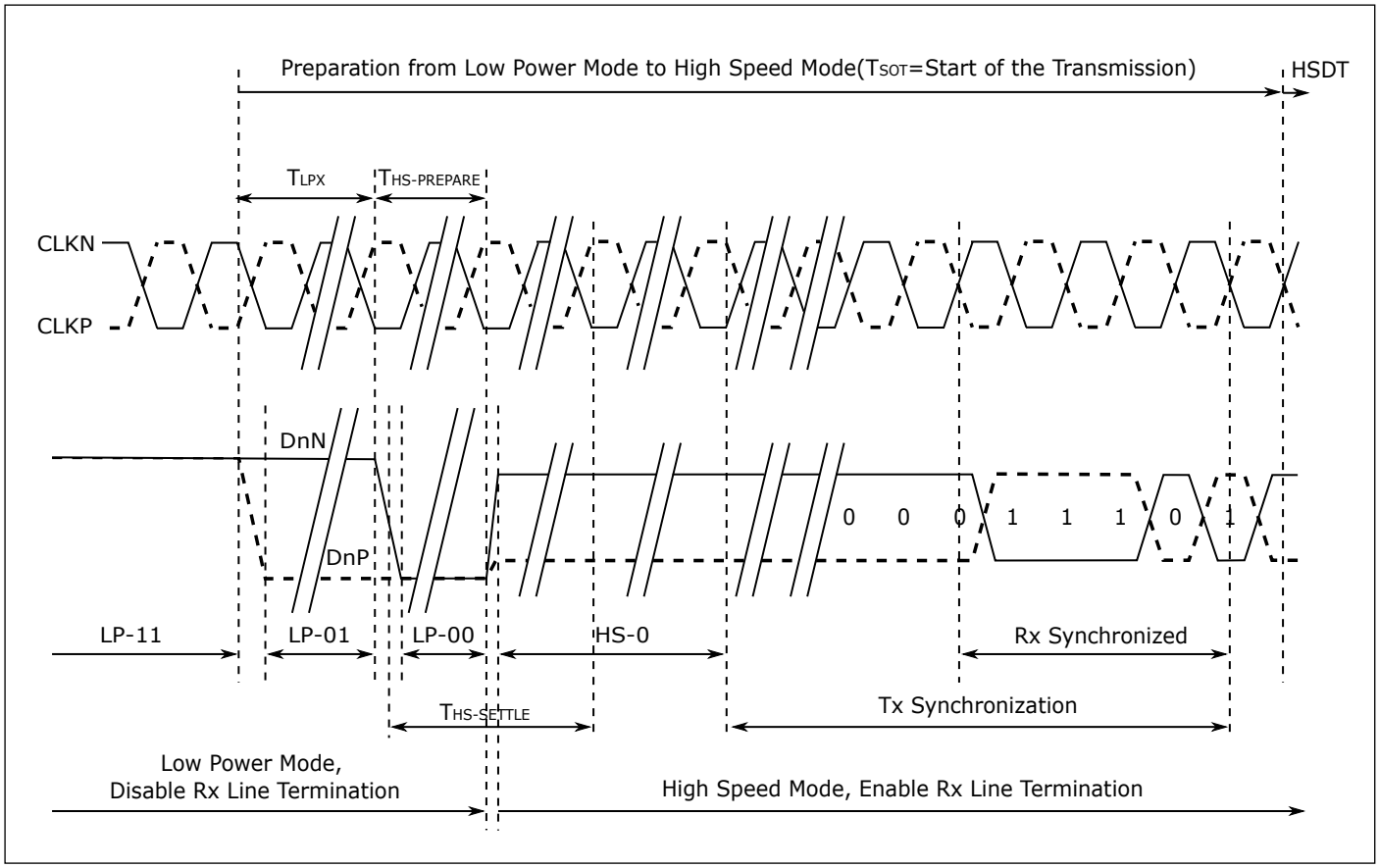


Fig. 12.11: Sequence of Entering HSDT Mode

Sequence:

- Start: LP-11
- HS request: LP01
- HS determination: LP-00=>HS-0 (RX: enabled lane terminal)
- RX synchronization: 011101 (by order of transmitted bits); TX synchronization: 00011101 (by order of transmitted bits)
- End: HSDT, ready to receive loaded high-speed data.

When the clock lane of the MCU is in the HSCM Mode, the Display Module can leave the HSDT Mode, and the clock lane must stay in HSCM until all data lanes switch to the LP11 Mode. The sequence of leaving the HSDT Mode is shown as follows:

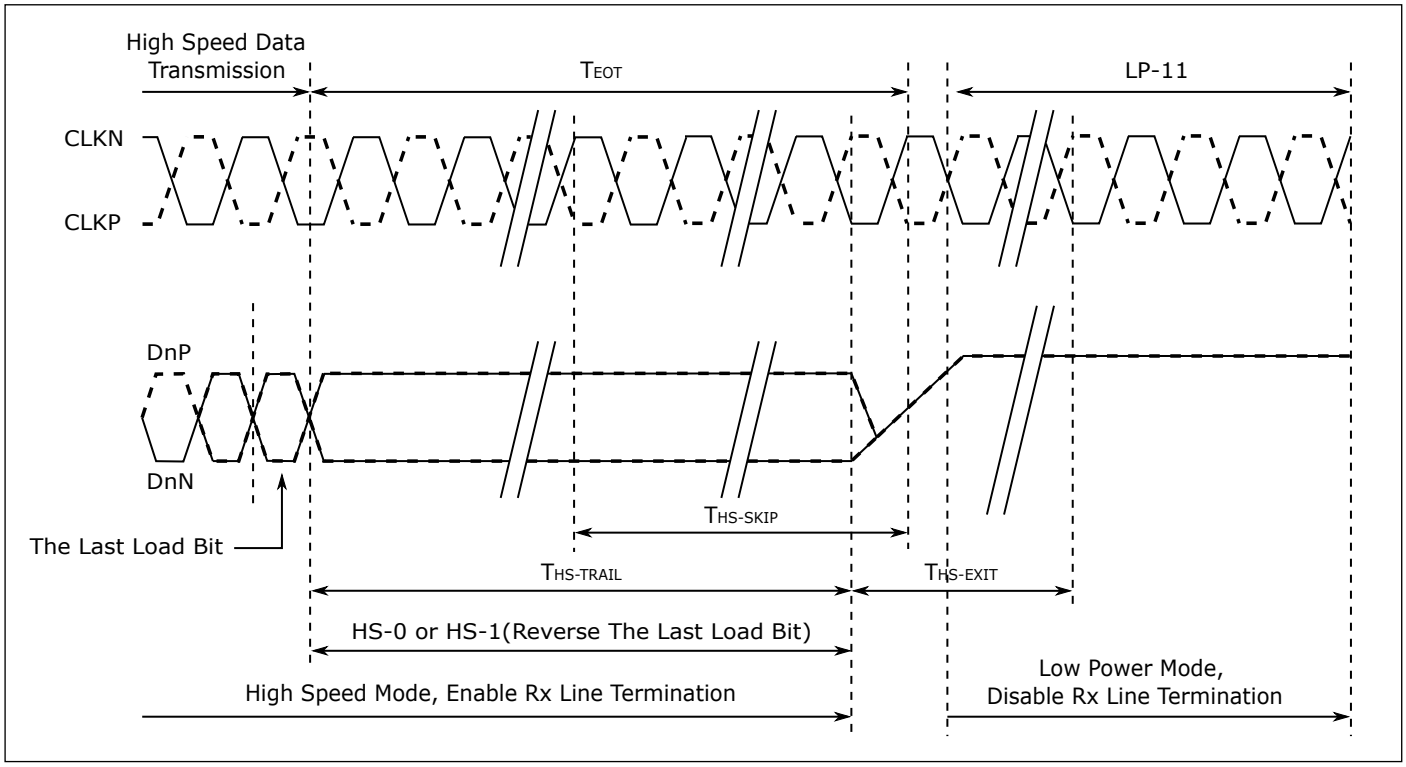


Fig. 12.12: Sequence of Leaving HSDT Mode

Sequence:

- Start: HSDT.
- Stop HSDT: If the last loading bit is HS-0, the MCU changes to HS-1; if the last loading bit is HS-1, MCU changes to HS-0.
- End: LP-11 (RX: disabled lane terminal).

The burst operation of HSDT can be composed of one or multiple packets, which can be either short or long packets.

Here are some different burst examples of HSDT:

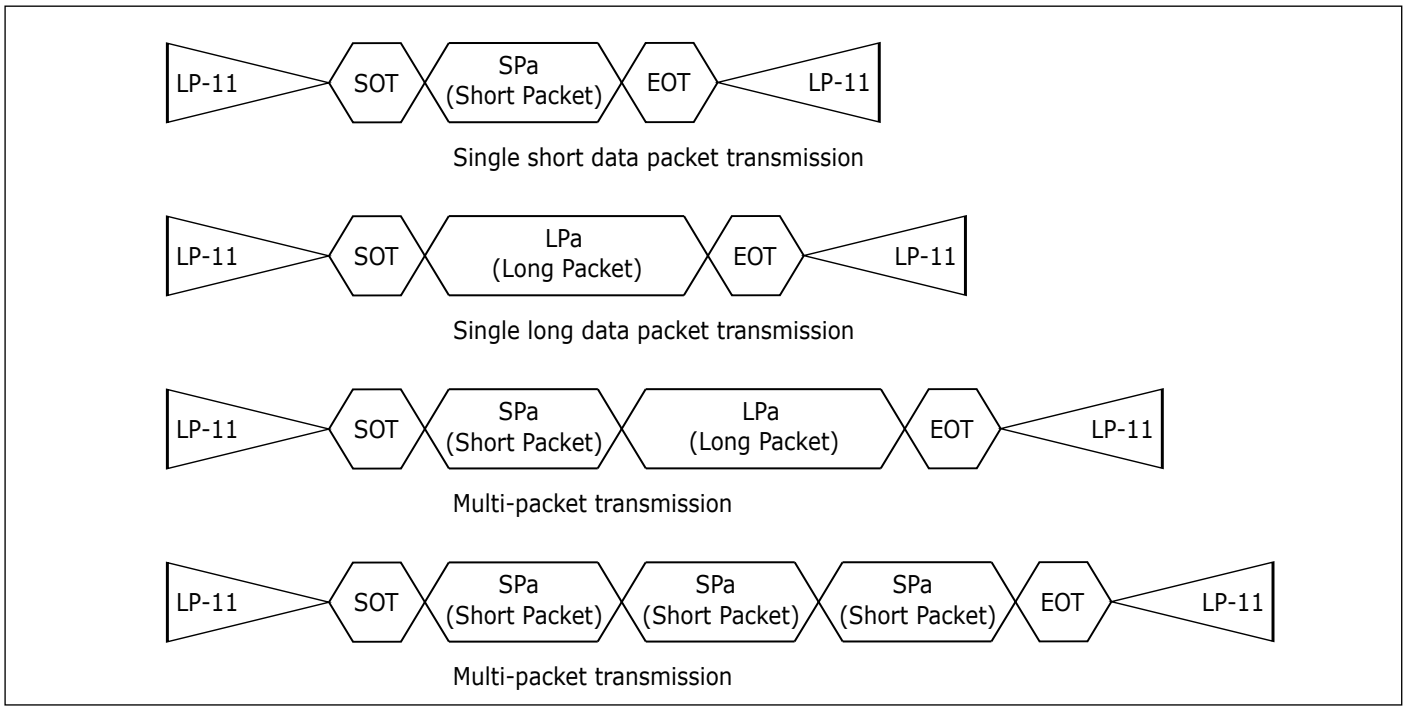


Fig. 12.13: HSDT Burst Sequence

12.3.4.2 Bus Turn-Around(BTA)

When the MCU or Display Module that is controlling the lane D0P/N wants to get information from the RX end, it can initiate a bus turnaround request to reverse the bus control. The MCU and the Display Module initiate the bus turnaround request using the same sequence, as shown below:

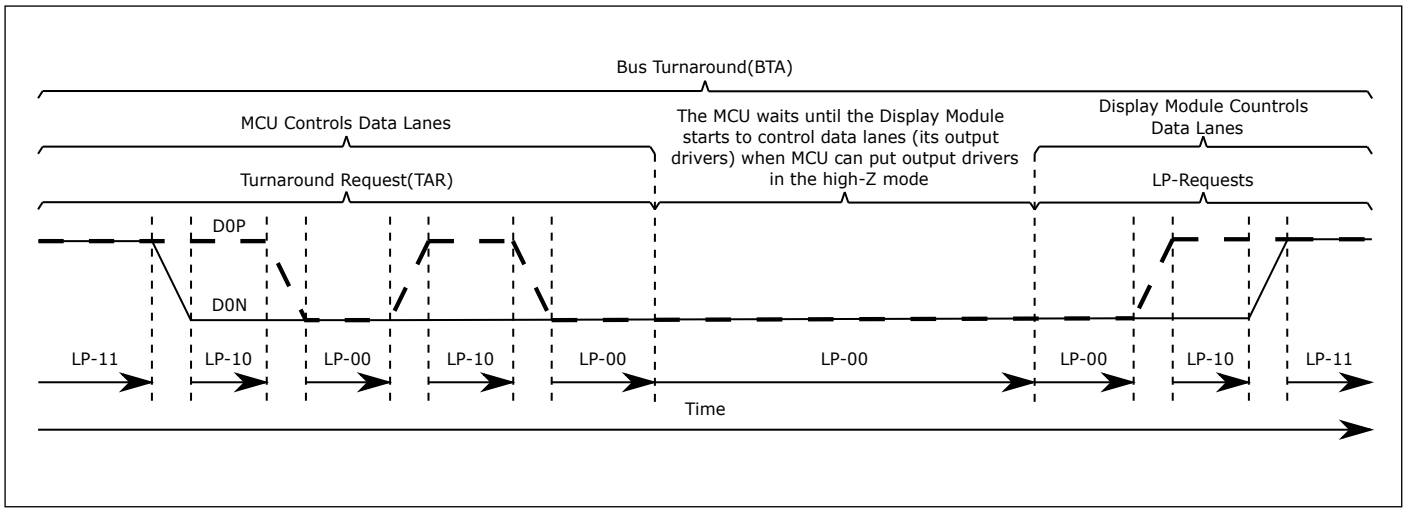


Fig. 12.14: Bus Turnaround Sequence

Sequence:

- Start: LP-11
- Turnaround request (MCU): LP-11=>LP-10=>LP-00=>LP-10=>LP-00
- MCU waits until the Display Module starts to control the lane D0P/N, while the MCU stops controlling the lane D0P/N and sets it to the high-impedance state.
- The Display Module changes to the Stop Mode: LP-00=>LP-10=>LP-11

12.3.4.3 Escape Mode

The Escape Mode, a special mode under LPM, has the following features:

1. Send LPDT command from the MCU to the Display Module.
2. Drive data lanes to the ULPS state.
3. Send a RAR command to reset the Display Module.
4. Send an ACK signal for transmitting non-error events from the Display Module to the MCU.

The basic sequence of the Escape Mode is shown as follows:

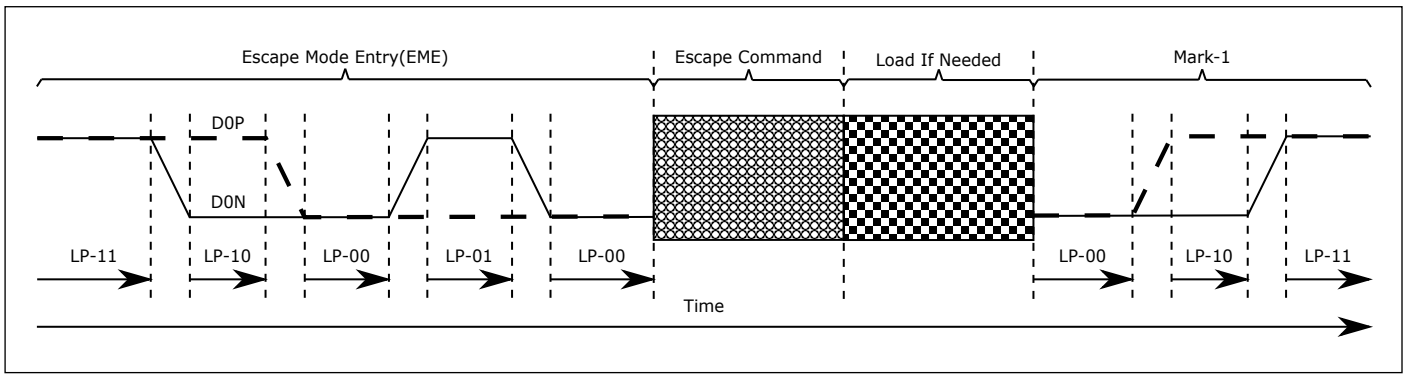


Fig. 12.15: Escape Mode Sequence

Sequence:

- Start: LP-11
- Switch to the Escape Mode (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00
- Escape command (EC): When one of the data lanes changes from Low to High and then goes Low, this change shows the value of the current data bit (D0P = 1, D0N = 0)
- When the data lane 0 changes from Low to High and then goes Low, the receiver latches 1 bit data, that is, logic 0. The receiver takes the transmission from Low to High and then to Low
- as its internal clock

- If necessary, load the data
- Leave the Escape Mode: LP-00=>LP-10=>LP-11
- End: LP-11

Escape command types are shown in the following table:

Escape Command	Command Type	Entry Command Pattern (First Bit->Last Bit Transmitted)	Dn	D0
Low-Power Data Transmission	Mode	1110 0001 b	×	√
Ultra-Low Power Mode	Mode	0001 1110 b	√	√
Note 1	Mode	1001 1111 b	×	×
Note 1	Mode	1101 1110 b	×	×
Remote Application Reset	Trigger	0110 0010 b	×	√
Tearing Effect	Trigger	0101 1101 b	×	×
Acknowledge	Trigger	0010 0001 b	×	√
Note 1	Trigger	1010 0000 b	×	×

Note:
1. The support of this command has not been implemented on the display module;
2. The value of n is 1, 2 and 3;
3. "√" means support;
4. "×" means not supported.

Fig. 12.16: Escape Commands

12.3.4.4 Low Power Data Transmission (LPDT)

When the data lane changes to the Escape Mode and the MCU has sent the LPDT command to the Display Module, the MCU can send the data to the Display Module in the LPM. The Display Module also uses the same sequence when sending data to the MCU. The LPDT sequence is shown as follows:

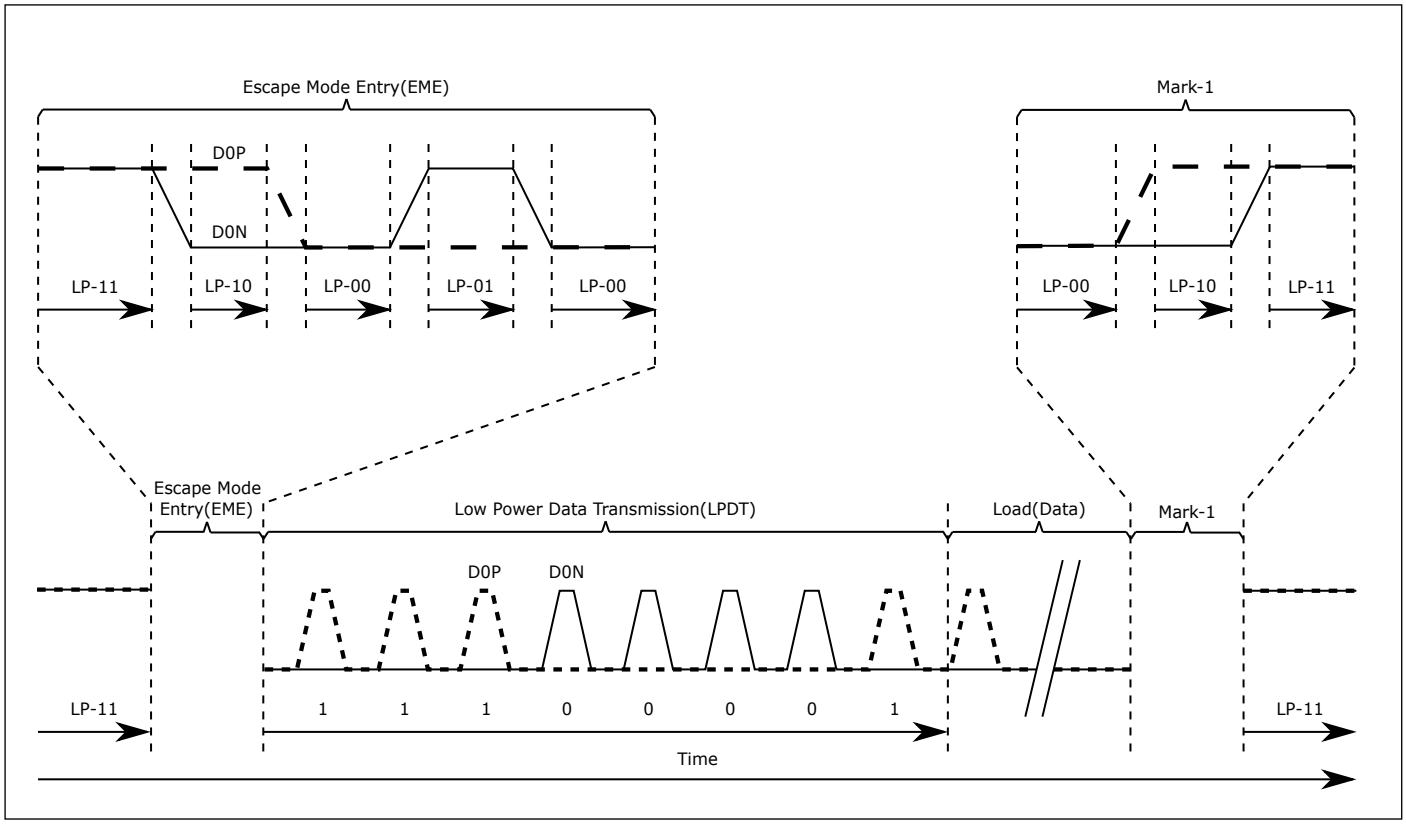


Fig. 12.17: LPDT Sequence

Sequence:

- Start: LP-11
- Switch to the Escape Mode (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00
- Send the LPDT command in the Escape Mode: 0x87 (LSBfirst).
- Data loading: One or more bytes, which are in a suspended state when the data lane stops between bytes (all lanes are Low).
- Leave the Escape Mode: LP-00=>LP-10=>LP-11
- End: LP-11

12.3.4.5 UltraLow Power State (ULPS)

When the data lane switches to the Escape Mode, MCU can mandatorily change the data lane to the ULPS. The ULPS sequence is shown as follows:

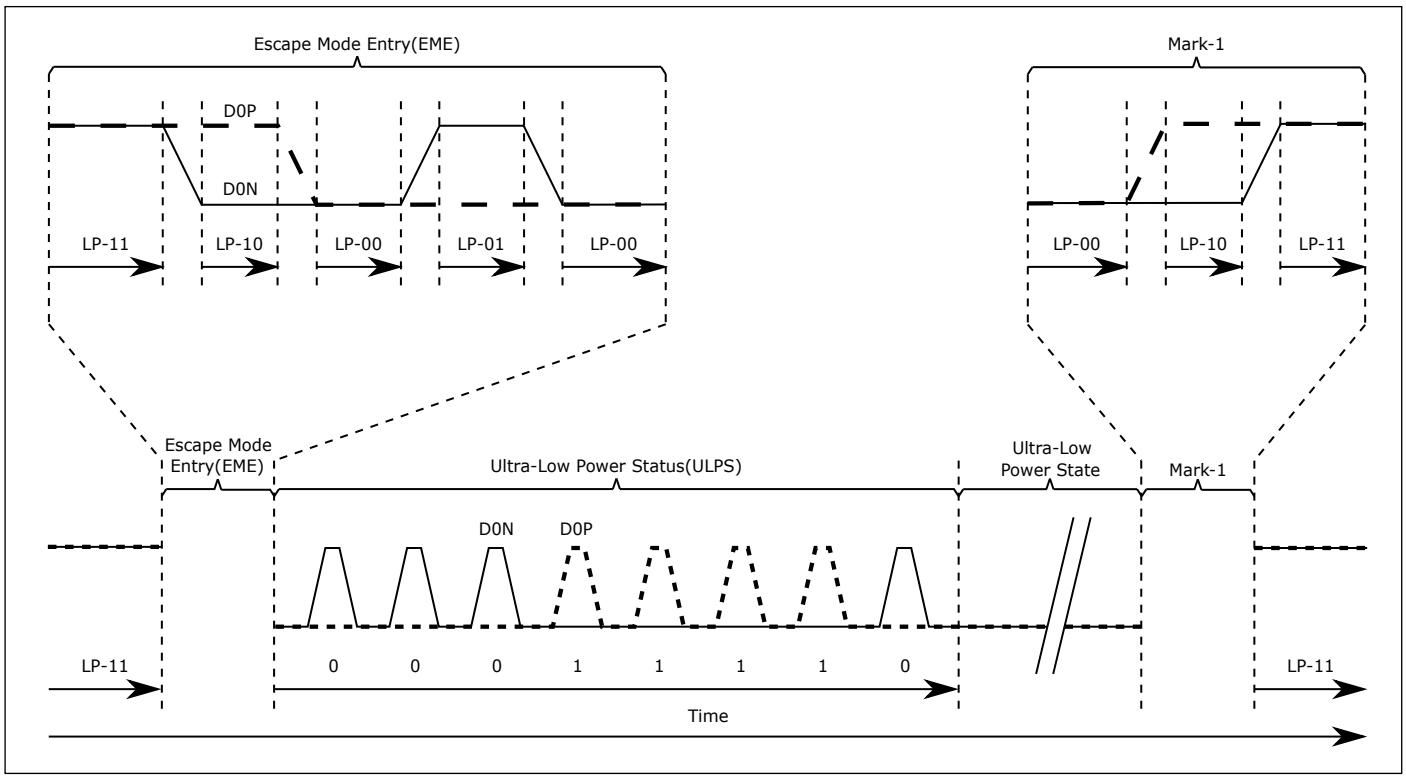


Fig. 12.18: ULPS Sequence

Sequence:

- Start: LP-11
- Switch to the Escape Mode (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00
- Send the ULPS command in the Escape Mode: 0x78 (LSBfirst)
- ULPS: The MCU keeps the data lane Low
- Leave the Escape Mode: LP-00=>LP-10=>LP-11
- End: LP-11(after the data lane exits ULPS, wait 100 us before sending the next command)

12.3.4.6 Remote Application Reset (RAR)

RAR is one of the Trigger commands. When the data lane switches to the Escape Mode, the MCU can inform the Display Module to reset in the remote application reset trigger. The RAR sequence is shown as follows:

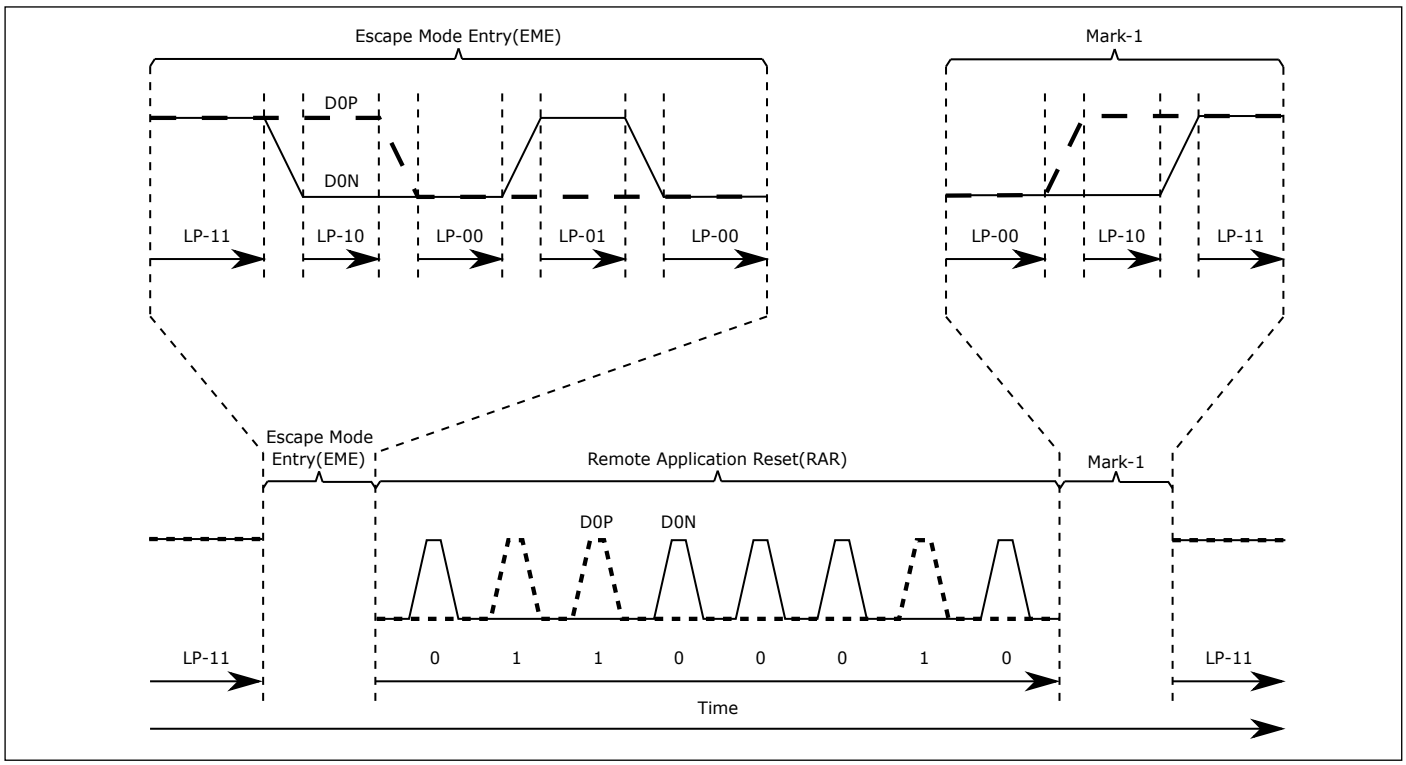


Fig. 12.19: RAR Sequence

Sequence:

- Start: LP-11
- Switch to the Escape Mode (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00
- Send the RAR command in the Escape Mode: 0x46 (LSBfirst)
- Leave the Escape Mode: LP-00=>LP-10=>LP-11
- End: LP-11

12.3.4.7 Acknowledgement (ACK)

ACK is one of the Trigger commands. The Display Module can notify the MCU of unrecognized errors through ACK. The Display Module sends ACK in a sequence shown in the following figure:

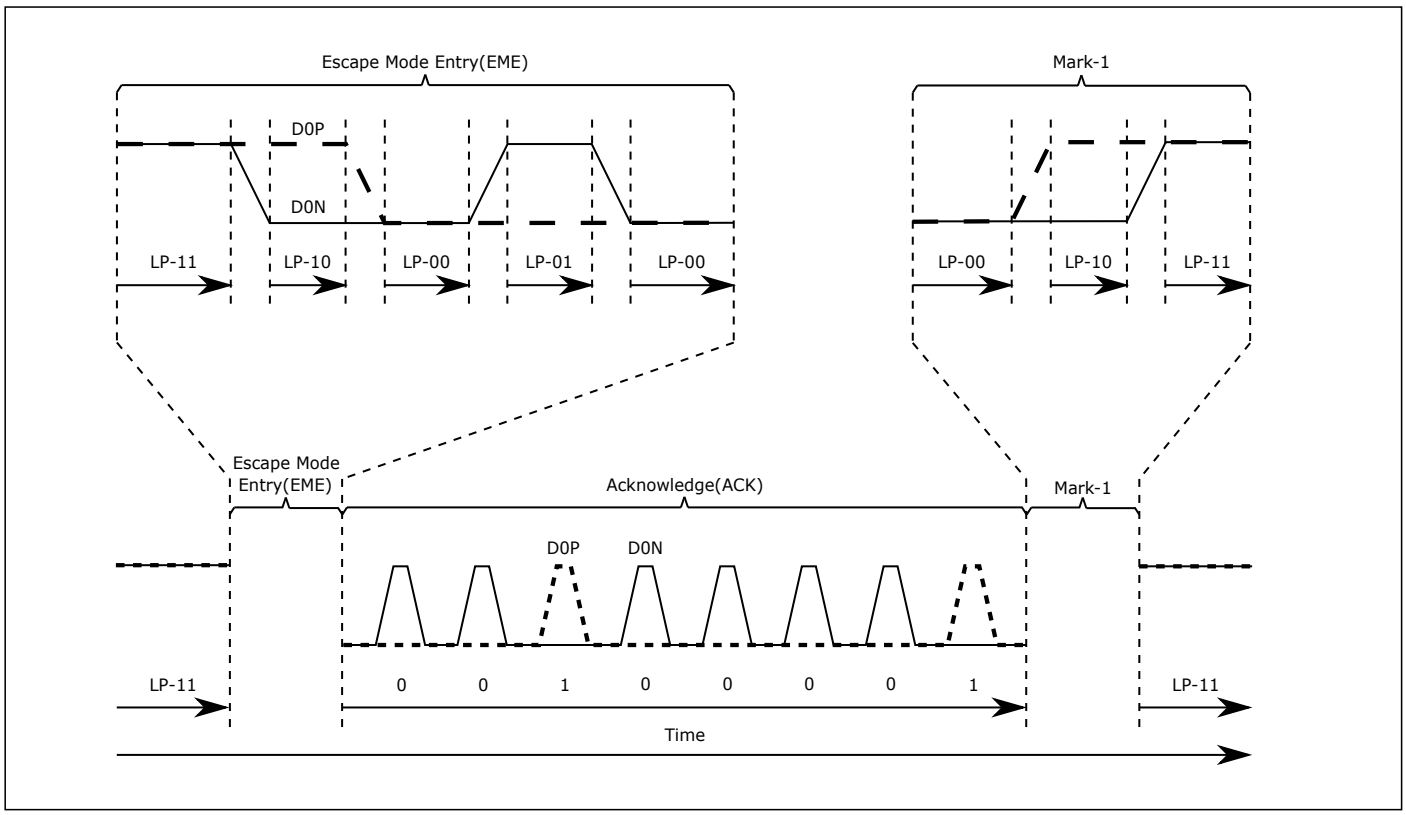


Fig. 12.20: ACK Sequence

Sequence:

- Start: LP-11
- Switch to the Escape Mode (EME): LP-11=>LP-10=>LP-00=>LP-01=>LP-00
- Send the ACK command in the Escape Mode: 0x84 (LSBfirst)
- Leave the Escape Mode: LP-00=>LP-10=>LP-11
- End: LP-11

12.3.5 Lane Management Layer

The lanes of DSI are expandable. The number of data lanes can be one, two, three, or four as expanded based on the bandwidth requirements of practical applications, and the peak bus bandwidth can gain an approximate linear increase. At the TX end, this layer aims to distribute the data from the protocol layer to each data lane in a certain order. At the RX end, this layer aims to collect transmitted bytes from each lane and merge them into a complete packet. All data lanes share a clock signal, but data transfer on each lane may not be completed simultaneously. When the number of bytes transmitted is not an integral multiple of the number of lanes, some lanes will finish data transfer before others. Except for Lane0, the order of the other three lanes can be configured by <CR_LANE_MUX_SEL>.

There are four orders:

- Lane0, Lane1, Lane2, Lane3
- Lane0, Lane3, Lane1, Lane2
- Lane0, Lane2, Lane3, Lane1
- Lane0, Lane2, Lane1, Lane3

Taking the single lane and four lanes (Lane0, Lane1, Lane2, Lane3) as examples, the TX end is shown as follows:

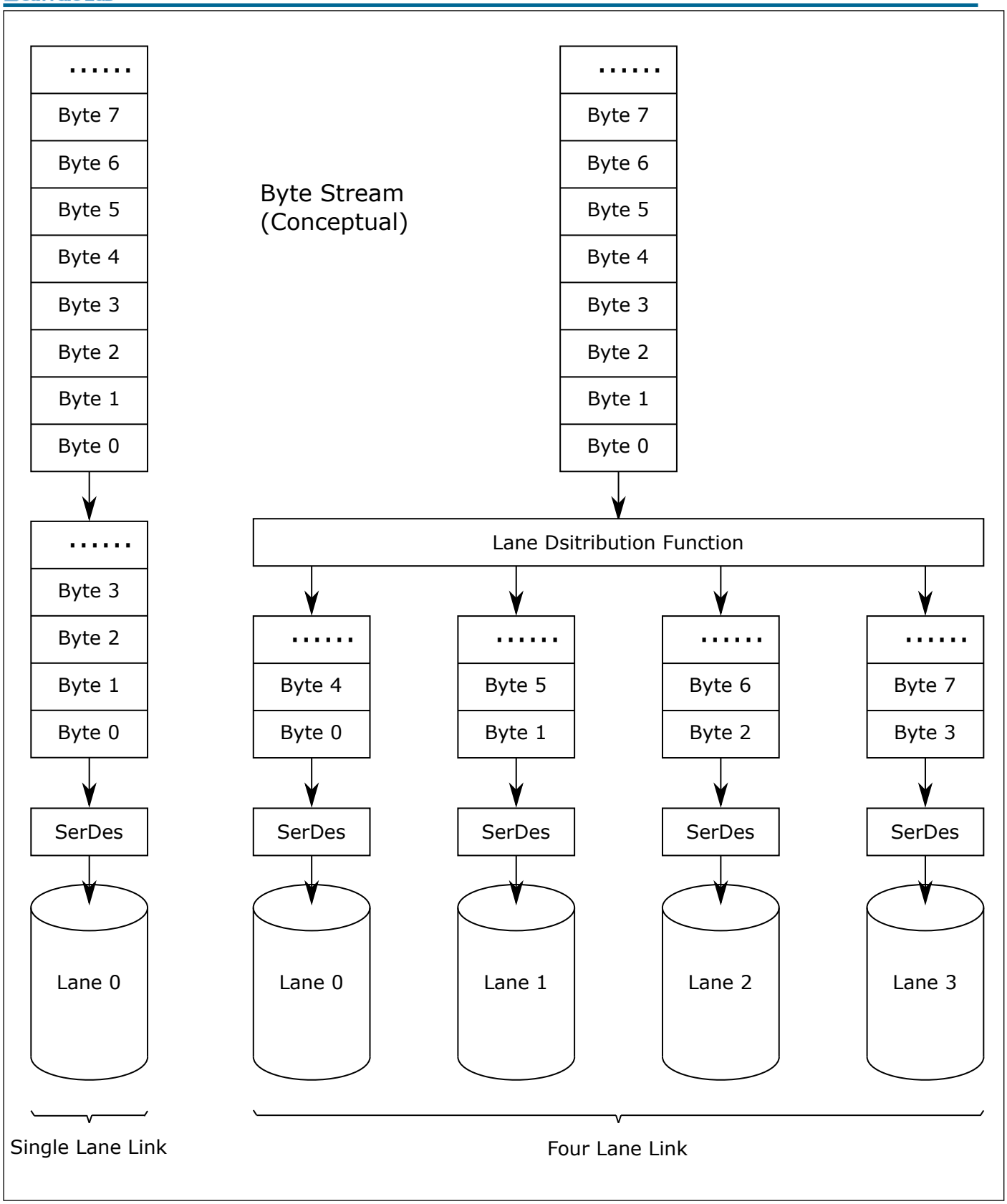


Fig. 12.21: TX End Distribution

The RX end is shown as follows:

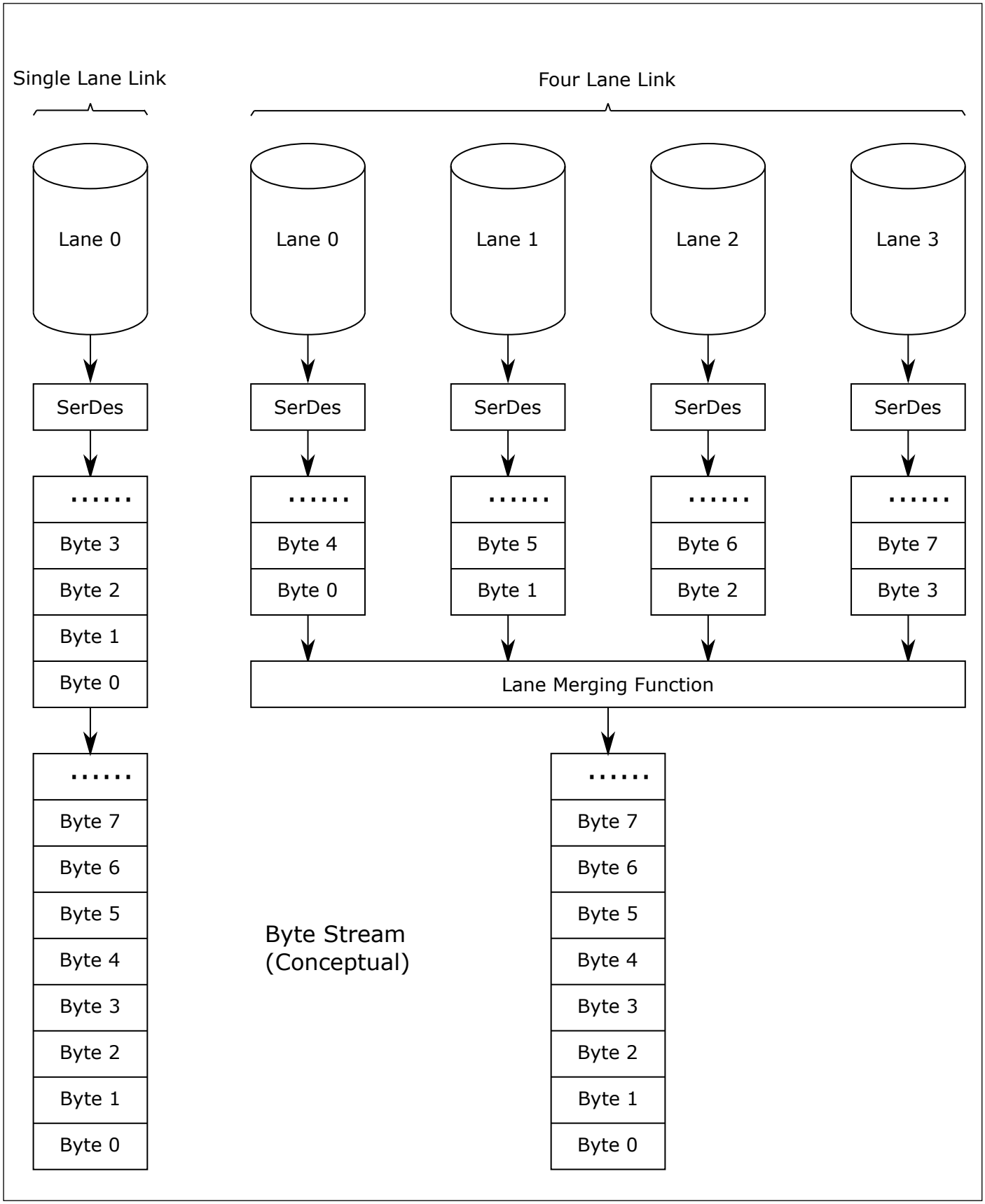


Fig. 12.22: RX End Merging

The lane distributor accepts HS transmission with arbitrary byte length, buffers N bytes, where N is the number of lanes implemented in the interface, and sends N byte arrays on N lanes in parallel. Before sending data, all lanes execute the SoT sequence in parallel to indicate the start of the first byte of the packet to the corresponding receiving unit. After the SoT, the lane sends N byte arrays from the first packet in parallel through a cyclic process. For a four-lane system, Byte 0 of the packet enters Lane 0, Byte 1 enters Lane 1, Byte 2 enters Lane 2, Byte 3 enters Lane 3, Byte 4 enters Lane 0, and so on. As HS transmission involves any number of bytes, which may not be an integral multiple of the number of lanes, some lanes will finish data transfer before others. Although all lanes start transmission at the same time by executing SoT in parallel, each lane runs independently and some lanes may complete HS transmission before others and send EoT one byte in advance. It is as shown below:

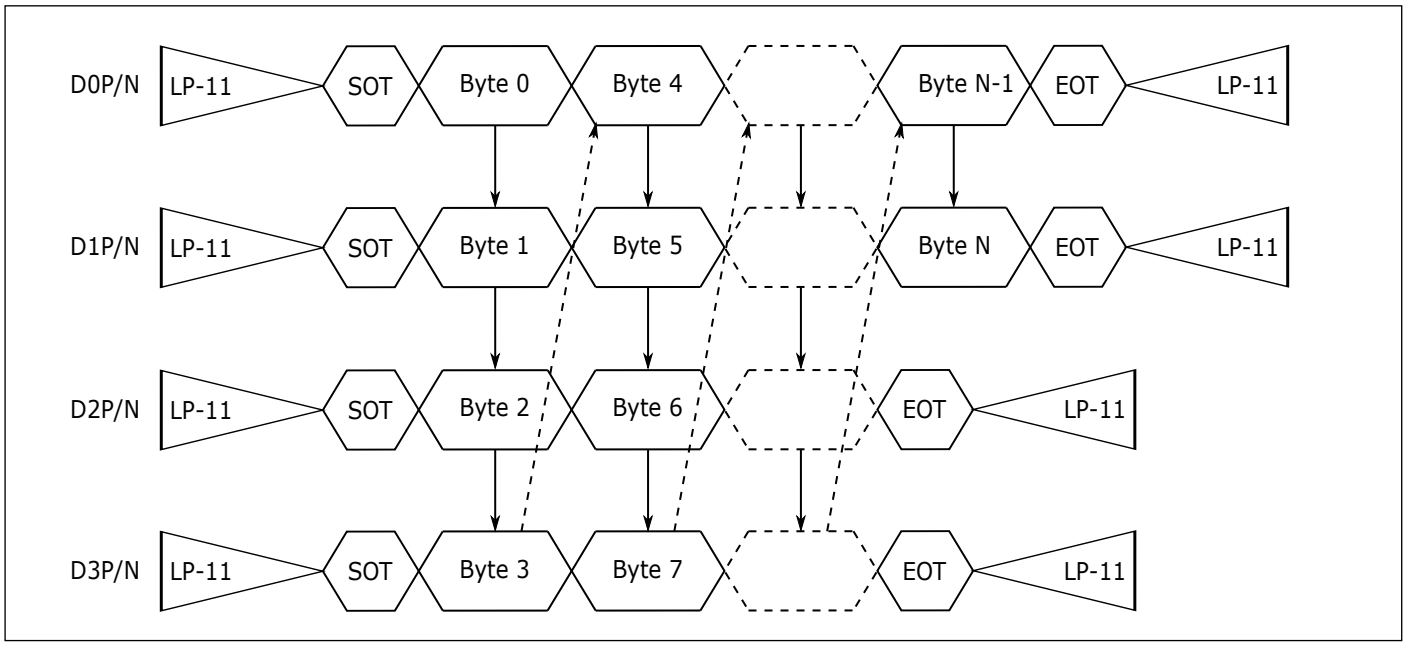


Fig. 12.23: Transmission of Bytes (Non-integral Multiple of the Number of Lanes)

12.3.6 Protocol Layer

DSI is a communication protocol based on packet transmission. Essentially, the commands and data are transmitted between the MCU and the Display Module in the packet format. DSI defines short packet (SPa) and long packet (LPa). The type of packet (SPa or LPa) can be identified from its header (PH).

12.3.6.1 Short Packet

A short packet consists of 8-bit data identification (DI), two bytes of commands or data, and 8-bit ECC. The length of a short packet is 4 bytes including ECC. Short packets are mainly used for most command pattern commands and related parameters and also used to transmit events like H-sync and V-sync edges. Due to short length, the timing information can be accurately transmitted to the peripheral logic. The structure of a short packet is shown as follows:

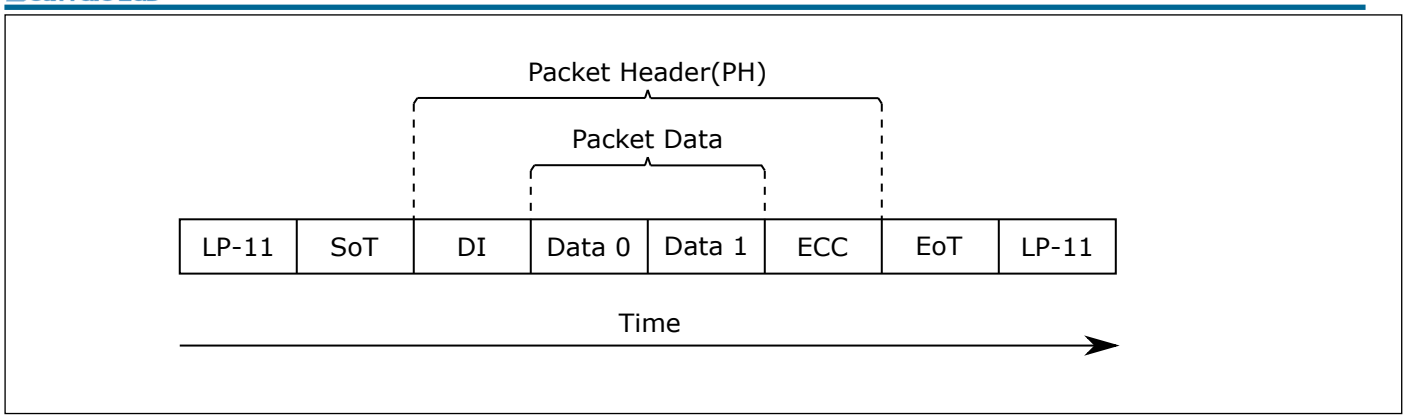


Fig. 12.24: Structure of Short Packet

- LP-11: low power stop state
- SoT: start of transmission
- DI: 8-bit, data identification
- Data 0: 8-bit, packet data 0
- Data 1: 8-bit, packet data 1
- ECC: 8-bit, error correction code
- EoT: end of transmission

12.3.6.2 Long Packet

The long packet consists of a 32-bit packet header (PH), a data payload with a variable number of bytes, and a 16-bit packet footer (PF). The packet header consists of 8-bit data identification, 16-bit word count (WC), and 8-bit ECC. Since the WC that records the payload length is two bytes, and the value ranges from 0 to 65,535, the length of a long packet ranges from 6 to 65,541 bytes. Long packets are mainly used to transmit a large amount of image data or some control commands. The structure of a long packet is shown as follows:

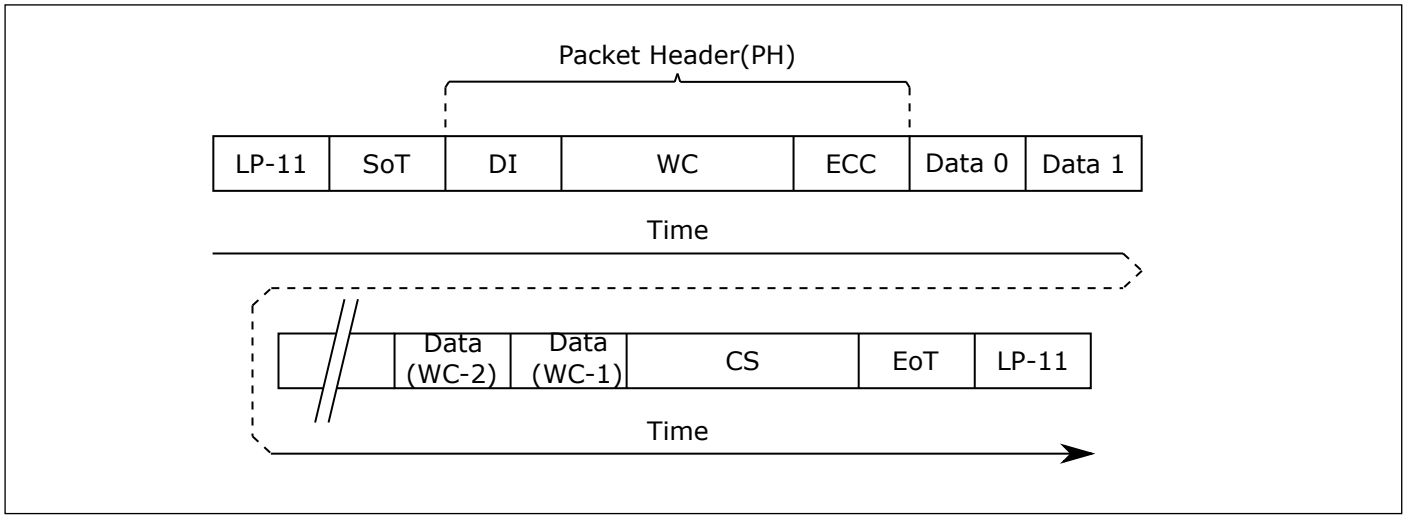


Fig. 12.25: Structure of Long Packet

- LP-11: low power stop state
- SoT: start of transmission
- DI: 8-bit, data identification
- WC: 16-bit, word count
- ECC: 8-bit, error correction code
- Data 0, Data 1⋯: packet data (0–65535 bytes)
- CS: 16-bit, checksum
- EoT: end of transmission

12.3.6.3 Multiple Packets per Transmission

In the above structural diagrams of short and long packets, the transmission contains only one packet. For multiple packets, if only one packet is sent in each transmission, the overhead of frequent switching between LPS and HS Mode will greatly limit the bandwidth. DSI allows multiple packets to be concatenated and sent without SoT, EoT and LP11 between packets, as shown below:

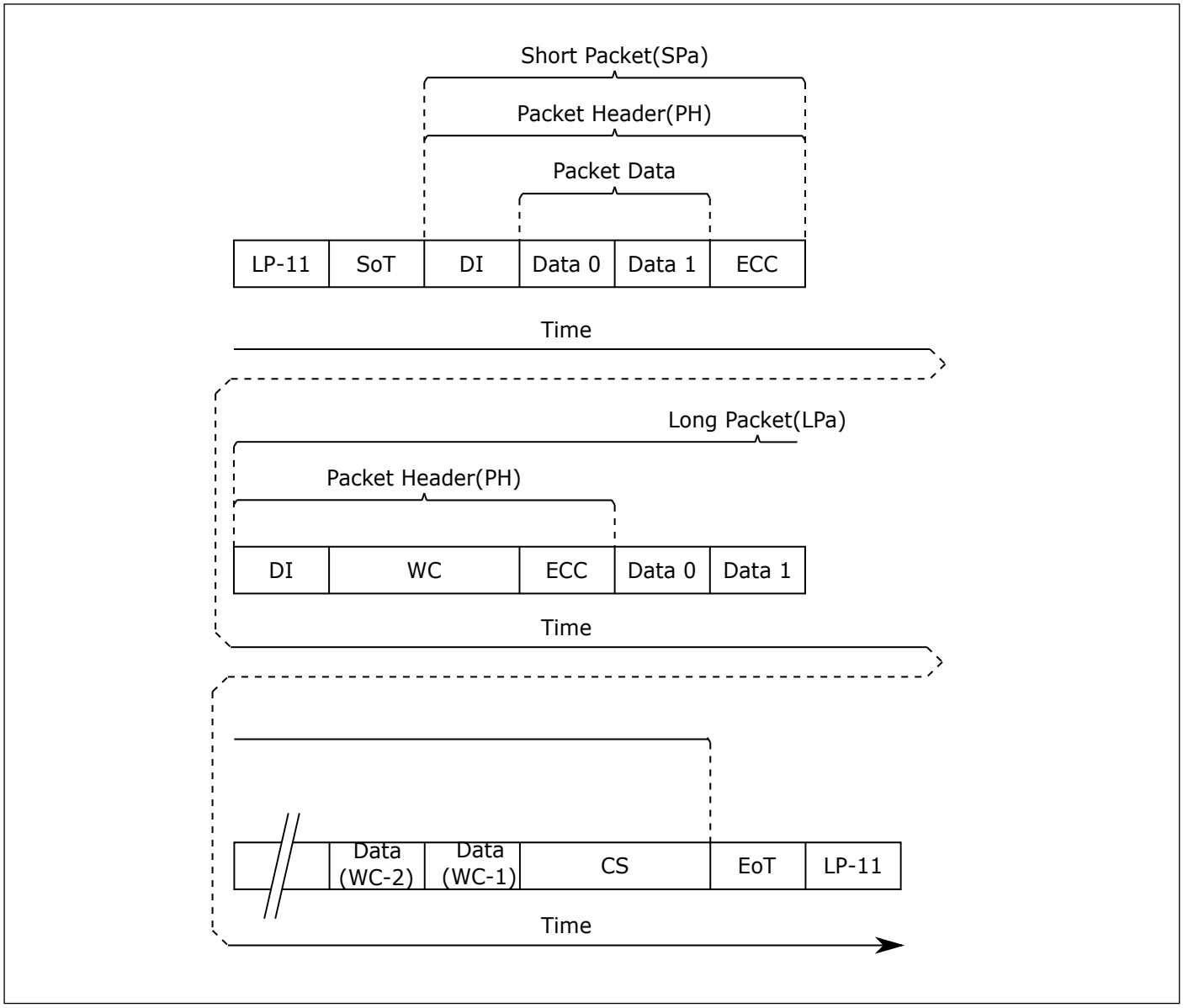


Fig. 12.26: Multiple Packets per Transmission

12.3.6.4 Bit Order of Bytes in a Packet

The bit order of the bytes used in a packet is that the LSB is sent first and the MSB is sent last. The following figure shows an example:

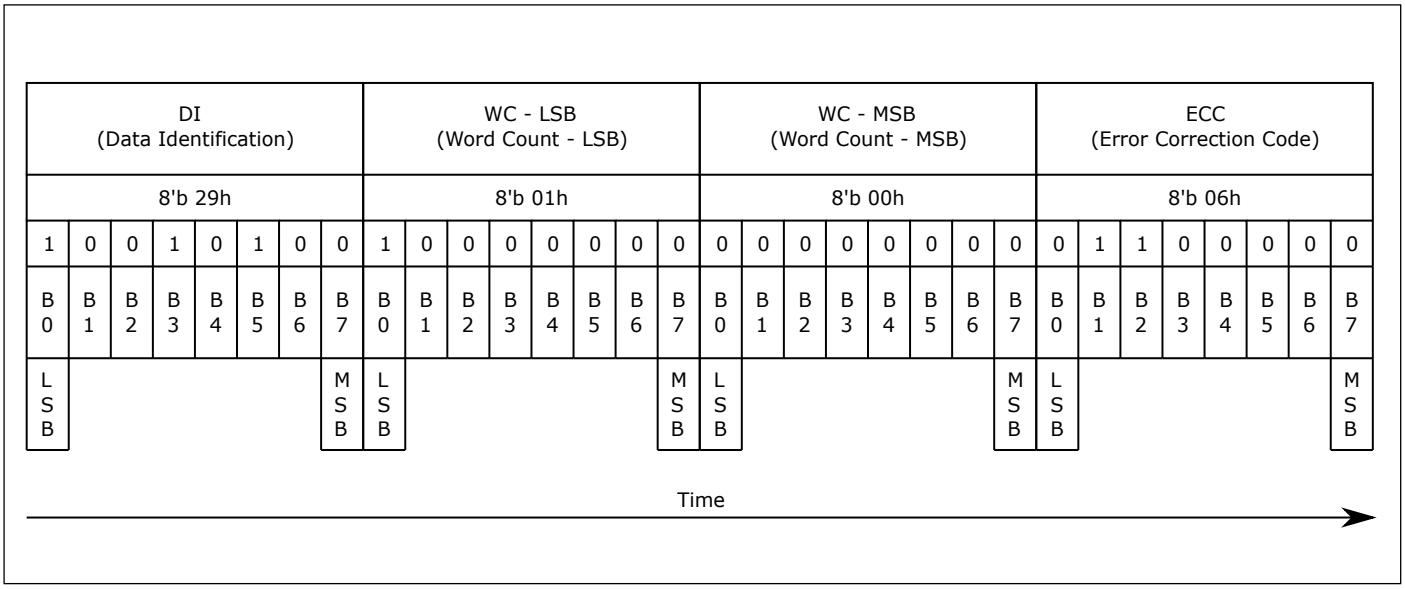


Fig. 12.27: Bit Order of Bytes in a Packet

12.3.6.5 Byte Order in a Packet

The byte order of multi-byte information used in a packet is that the LSB is sent first and the MSB is sent last. For example, the word count (WC) consists of 2 bytes. LSB is sent before the MSB, as shown below:

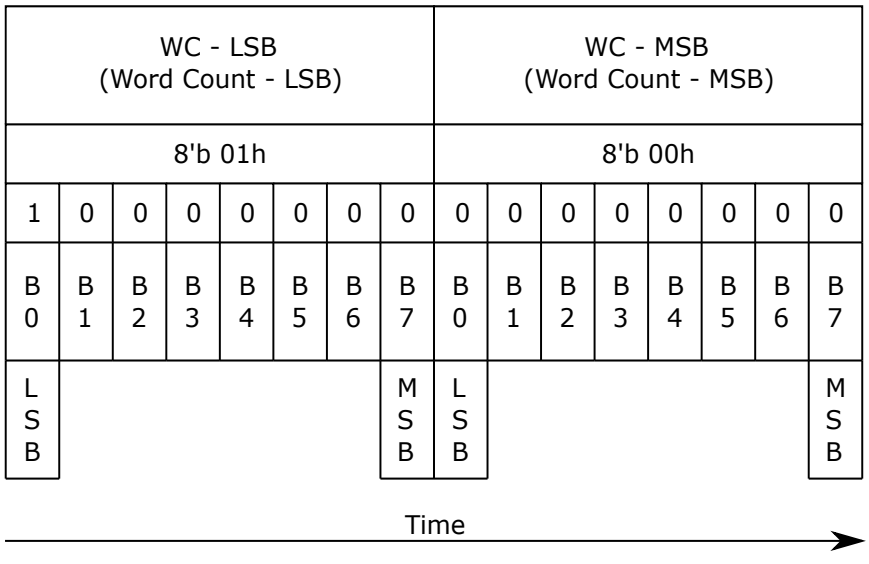


Fig. 12.28: Byte Order in a Packet

12.3.6.6 Packet Header (PH)

The packet header always consists of 4 bytes, and the contents of the 4 bytes are different for short and long packets. The short packet consists of 1-byte data identification (DI), 2-byte packet data (PD0, PD1), and 1-byte ECC. The long packet consists of 1-byte data identification (DI), 2-byte word count (WC), and 1-byte ECC.

12.3.6.7 Data Identification (DI)

DI, a part of the PH, consists of two parts:

- Virtual channel (VC), 2-bit, DI[7:6]
- Data type (DT), 6-bit, DI[5:0]

12.3.6.8 Virtual Channel (VC)

VC, a part of data identification (DI[7:6]) structure, is used to determine the destination to which MCU packets are sent. The virtual channel can allocate 4 different lanes to 4 different Display Modules, and each module uses the same virtual channel used by the MCU to send packets to it.

12.3.6.9 Data Type (DT)

DT, a part of data identification (DI[5:0]) structure, is used to define a short or long packet type and the packet format. The data types from the MCU to the Display Module are defined in the following table:

From the MCU to the Display Module								
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Hex	Description	Short/Long Packet
0	0	0	0	0	1	01	Sync Event, V Sync Start	SPa (Short Packet)
0	1	0	0	0	1	11	Sync Event, V Sync End	SPa (Short Packet)
1	0	0	0	0	1	21	Sync Event, H Sync Start	SPa (Short Packet)
1	1	0	0	0	1	31	Sync Event, H Sync End	SPa (Short Packet)
0	0	1	0	0	0	08	End of Transmission Packet (EoTP), Note 1	SPa (Short Packet)
0	0	0	0	1	0	02	Color Mode Off Command	SPa (Short Packet)
0	1	0	0	1	0	12	Color Mode On Command	SPa (Short Packet)
1	0	0	0	1	0	22	Shut Down Peripheral Command	SPa (Short Packet)
1	1	0	0	1	0	32	Turn On Peripheral Command	SPa (Short Packet)
0	0	0	0	1	1	03	Generic Short Write, No Parameter	SPa (Short Packet)
0	1	0	0	1	1	13	Generic Short Write, 1 Parameter	SPa (Short Packet)
1	0	0	0	1	1	23	Generic Short Write, 2 Parameters	SPa (Short Packet)
0	0	0	1	0	0	04	Generic Short Read, No Parameter	SPa (Short Packet)
0	1	0	1	0	0	14	Generic Short Read, 1 Parameter	SPa (Short Packet)
1	0	0	1	0	0	24	Generic Short Read, 2 Parameters	SPa (Short Packet)
0	0	0	1	0	1	05	DCS Write, No Parameter	SPa (Short Packet)
0	1	0	1	0	1	15	DCS Write, 1 Parameter	SPa (Short Packet)
0	0	0	1	1	0	06	DCS Read, No Parameter	SPa (Short Packet)
1	1	0	1	1	1	37	Set Maximum Return Packet Size	SPa (Short Packet)
0	0	1	0	0	1	09	Null Packet, No Data, Note 2	LPa (Long Packet)
0	1	1	0	0	1	19	Blanking Packet, No Data	LPa (Long Packet)
1	0	1	0	0	1	29	Generic Long Write	LPa (Long Packet)
1	1	1	0	0	1	39	DCS Write Long	LPa (Long Packet)
0	0	1	1	1	0	0E	Packed Pixel Stream, 16-bit RGB, 5-6-5 Format	LPa (Long Packet)
0	1	1	1	1	0	1E	Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	LPa (Long Packet)
1	0	1	1	1	0	2E	Loosely Packed Pixel Stream, 18-bit RGB, 6-6-6 Format	LPa (Long Packet)
1	1	1	1	1	1	3E	Packed Pixel Stream, 24-bit RGB, 8-8-8 Format	LPa (Long Packet)
x	x	0	0	0	0	x0	Do Not Use	
x	x	1	1	1	1	xF	All Unspecified Codes Are Reserved	

Note:
1. This item can be used when the MCU wants to confirm the end of the transmission in the high-speed data transmission mode;
2. This item can be used when the data channel remains in high-speed data transfer mode.

Fig. 12.29: Data Types from the MCU to the Display Module

The data types from the Display Module to the MCU are defined in the following table:

From the Display Module to the MCU								
Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Hex	Description	Short/Long Packet
0	0	0	0	1	0	02	Acknowledge with Error Report	SPa (Short Packet)
0	0	1	0	0	0	08	End of Transmission Packet (EoTP)	SPa (Short Packet)
0	1	0	0	0	1	11	Generic Short Read Response, 1 Byte Returned	SPa (Short Packet)
0	1	0	0	1	0	12	Generic Short Read Response, 2 Bytes Returned	SPa (Short Packet)
0	1	1	0	1	0	1A	Generic Long Read Response	LPa (Long Packet)
0	1	1	1	0	0	1C	DCS Read Long Response	LPa (Long Packet)
1	0	0	0	0	1	21	DCS Read Short Response, 1 Byte Returned	SPa (Short Packet)
1	0	0	0	1	0	22	DCS Read Short Response, 2 Bytes Returned	SPa (Short Packet)

Fig. 12.30: Data Types from the Display Module to the MCU

12.3.6.10 Packet Data (PD)

For short packets, PD is a part of the PH, located after the data identification, and is fixed to 2 bytes, namely data 0 and data 1. The order of sending PD is sending data 0 before data 1. If the length of information is 1 byte, the value of data 1 is 0. For long packets, PD is located after the PH, and the byte length depends on the WC.

12.3.6.11 Word Count (WC)

WC is a part of the header of a long packet and located after the data identification. It is used to determine the total number of bytes in the PD of a long packet. WC is fixed to 2 bytes. The LSB is sent before the MSB.

12.3.6.12 Error Correction Code (ECC)

ECC, a part of the PH, can identify one or more errors and correct only one error. ECC protects the following fields:

- short packet: DI (8-bit), PD (16-bit) and ECC (8-bit)
- long packet: DI (8-bit), WC (16-bit) and ECC (8-bit)

Taking short packets as an example, the ECC calculation method for each byte is shown as follows (" ^" indicates exclusive OR operation).

DI (Data Identification)								Data 0 (Packet Data)								Data 1 (Packet Data)								ECC (Error Correction Code)							
8'b 05h								8'b 10h								8'b 00h								8'b 2Ch							
1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
D0	D1	D2		D4	D5		D7			D10	D11		D13			D16				D20	D21	D22	D23	P0							
D0	D1		D3	D4		D6		D8		D10		D12		D14			D17			D20	D21	D22	D23		P1						
D0		D2	D3		D5	D6			D9		D11	D12			D15			D18		D20	D21	D22				P2					
	D1	D2	D3				D7	D8	D9				D13	D14	D15				D19	D20	D21		D23				P3				
				D4	D5	D6	D7	D8	D9							D16	D17	D18	D19	D20		D22	D23					P4			
										D10	D11	D12	D13	D14	D15	D16	D17	D18	D19		D21	D22	D23						P5		
B0	B1	B2	B3	B4	B5	B6	B7	B0	B1	B2	B3	B4	B5	B6	B7	B0	B1	B2	B3	B4	B5	B6	B7	B0	B1	B2	B3	B4	B5	B6	B7
LSB							MSB	LSB							MSB	LSB							MSB	LSB							MSB

Time

Fig. 12.31: Example of ECC Calculation

Formula:

- ECC[7] = 0
- ECC[6] = 0
- ECC[5] = D10^D11^D12^D13^D14^D15^D16^D17^D18^D19^D21^D22^D23
- ECC[4] = D4^D5^D6^D7^D8^D9^D16^D17^D18^D19^D20^D22^D23
- ECC[3] = D1^D2^D3^D7^D8^D9^D13^D14^D15^D19^D20^D21^D23
- ECC[2] = D0^D2^D3^D5^D6^D9^D11^D12^D15^D18^D20^D21^D22
- ECC[1] = D0^D1^D3^D4^D6^D8^D10^D12^D14^D17^D20^D21^D22^D23
- ECC[0] = D0^D1^D2^D4^D5^D7^D10^D11^D13^D16^D20^D21^D22^D23

12.3.6.13 Packet Footer (PF)

PF, a part of a long packet, is located after the packet data. PF is a checksum calculated from the packet data of a long packet. The checksum is a 16-bit cyclic redundancy check (CRC) value generated by the polynomial X16+X12+X5+X0, as shown below:

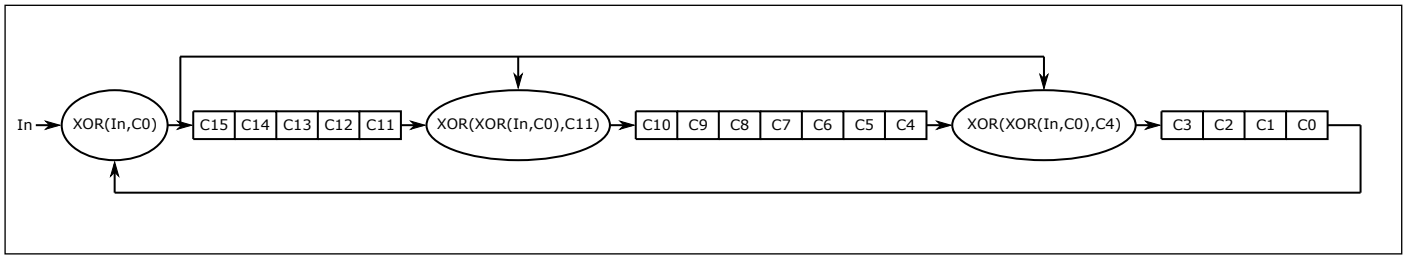


Fig. 12.32: CRC Calculation Method

The initial value of the 16-bit CRC generator before calculation is 0xFFFF, and the LSB of packet data is the first bit of the input CRC. The following figure illustrates the steps of CRC calculation by citing the data 0x01 as an example:

Step	In	XOR(In,C0)	C15	C14	C13	C12	C11	XOR(XOR(In,C0),C11(Step-1))	C10	C9	C8	C7	C6	C5	C4	XOR(XOR(In,C0),C4(Step-1))	C3	C2	C1	C0	C0
0	X	X	1	1	1	1	1	X	1	1	1	1	1	1	1	X	1	1	1	1	X
1	1(LSB)	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1
3	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1
4	0	1	1	1	1	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
5	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
6	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0
7	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0
8	0(MSB)	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0
	1 byte	CRC result	0	0	0	1	1		1	1	0	0	0	0	0		1	1	1	0	
			MSB																	LSB	

Fig. 12.33: Example of CRC Calculation

12.3.7 Application Layer

This layer describes the higher-level encoding and interpretation of data contained in the data stream. Depending on the display subsystem architecture, it may consist of pixels or coded bit streams having a prescribed format, or of commands that are interpreted by the display controller inside a Display Module. The DSI specification describes the mapping of pixel values, bit streams, commands and command parameters to bytes in the packet assembly. (For more details, see the Display Command Set (DCS) specification, which will not be presented here).

12.3.8 Command Mode

Command Mode refers to that the interaction between MCU and peripherals including the display controller (such as Display Module) by sending commands and data. The display controller may include local registers and compressed or uncompressed frame buffers. Systems using Command Mode write to, and read from, the registers and frame buffer memory. The MCU indirectly controls the activities at peripherals by sending commands, parameters and data to the display controller. The MCU can also read Display Module status information or the contents of the frame memory. Command Mode operation requires a bidirectional interface.

12.3.9 Video Mode

Video Mode refers to operation in which the transfer from the MCU to the peripheral takes the form of a real-time pixel stream. The Display Module relies on the MCU to provide image data at sufficient bandwidth to avoid flicker or other visible artifacts in the displayed image. Video information can only be transmitted in the HS Mode. Video Mode involves three modes: NonBurst Mode with SYNC Pulses, NonBurst Mode with SYNC Events, and Burst Mode. The three modes have different packet sequences, and the timing requirements of peripherals determine which mode is appropriate.

12.3.9.1 Non-Burst Mode with SYNC Pulses

It enables the peripheral to reconstruct the original video timing accurately, including sync pulse widths. In this mode, the goal is to convey DPI-type timing over the DSI serial Link. It includes matching DPI pixel-transmission rates, and widths of timing events like sync pulses. Accordingly, synchronization periods are defined using packets transmitting both start and end of sync pulses. The following figure shows an example of the mode. Periods shown as Horizontal Sync Active (HSA), Horizontal Back Porch (HBP) and Horizontal Front Porch (HFP) are filled by the timing intervals in LP Mode. DSI switches from HS to LP Mode at the beginning of HSA, HBP and HFP, stays in the LP11 state during the periods, and switches back to HS Mode at the end:

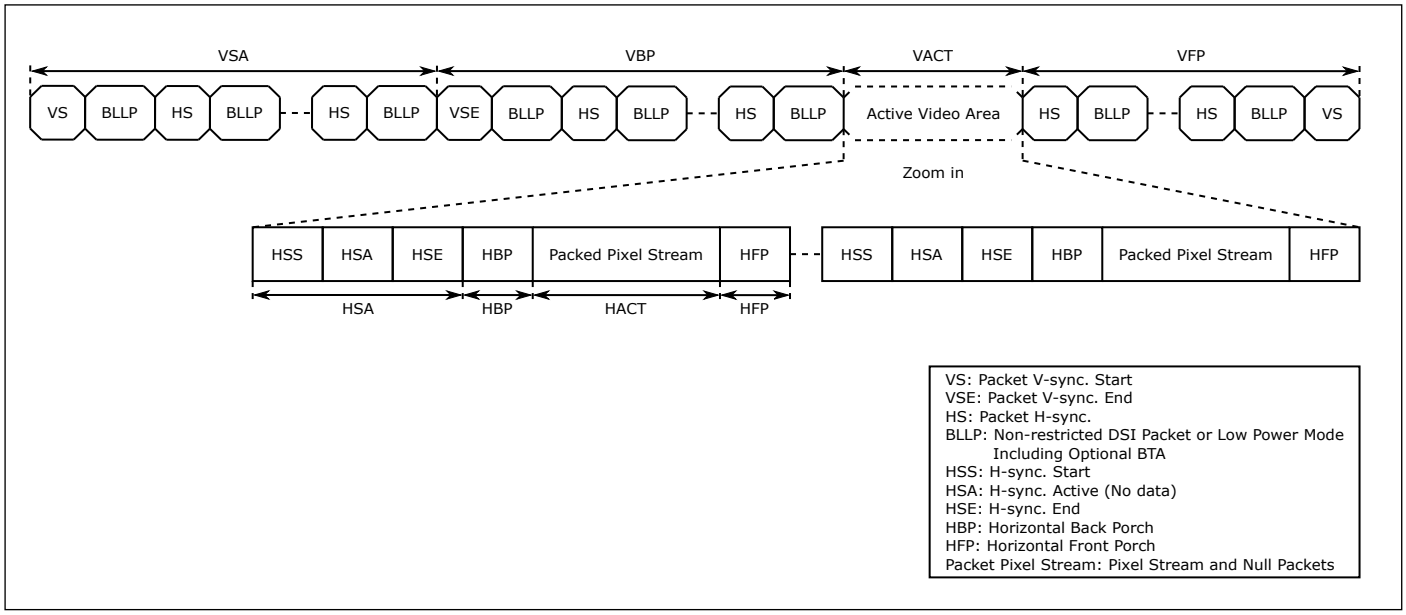


Fig. 12.34: Timing of NonBurst Mode with SYNC Pulses

12.3.9.2 Non-Burst Mode with SYNC Events

It is similar to the NonBurst Mode with SYNC Pulses. But an accurate reconstruction of sync pulse widths is not required, and hence the sync pulse is substituted by a single sync event, which is a simplification of the above NonBurst Mode with SYNC Pulses. In this mode, only the start of each sync pulse is transmitted. The peripheral may regenerate sync pulses as needed from each Sync Event packet received. Pixels are transmitted at the same rate as they would in a corresponding parallel display interface such as DPI-2. The following figure shows an example of the mode. Periods shown as Horizontal Sync Active (HSA), Horizontal Back Porch (HBP) and Horizontal Front Porch (HFP) are filled by the timing intervals in LP Mode. DSI switches from HS to LP Mode at the beginning of HSA, HBP and HFP, stays in the LP11 state during the periods, and switches back to HS Mode at the end:

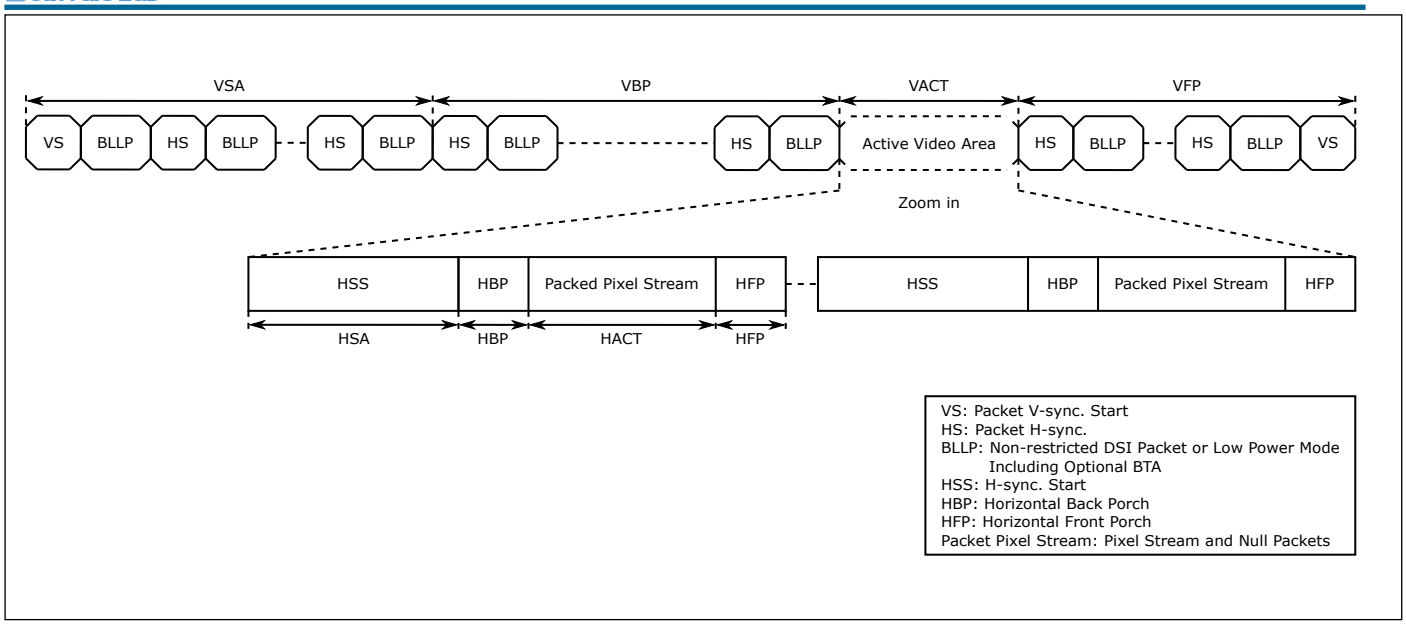


Fig. 12.35: Timing of NonBurst Mode with SYNC Events

12.3.9.3 Burst Mode

In this mode, the pixel packets are time-compressed, so that transmission can be completed in a short time. This strategy reduces overall DSI power consumption and enables larger blocks of time for other data transmissions over the Link in either direction. After the transmission of HS pixel data is completed, the bus switches to LP Mode. During this period, the bus may remain idle, i.e. staying in the LP11 state, or LP transmission can be performed in either direction. If the peripheral controls the bus that sends data to the MCU, its transmission time shall be limited to ensure that data will not flow from its internal buffer memory to the display device. The following figure shows an example of the mode. Periods shown as Horizontal Sync Active (HSA), Horizontal Back Porch (HBP) and Horizontal Front Porch (HFP) are filled by the timing intervals in LP Mode. DSI switches from HS to LP Mode at the beginning of HSA, HBP and HFP, stays in the LP11 state during the periods, and switches back to HS Mode at the end:

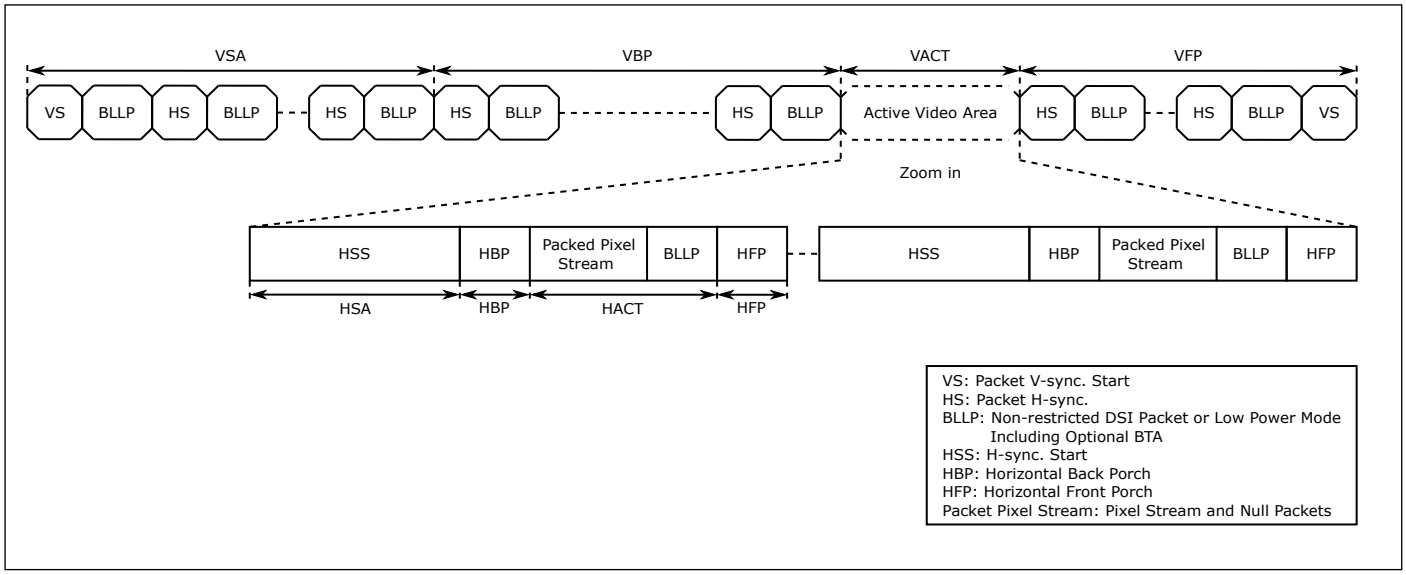


Fig. 12.36: Timing of Burst Mode

12.3.10 Line Buffer

Because the speed at which the frontend module inputs data to the DSI does not match the speed at which the DSI outputs data, it is necessary to buffer the data. Inside the DSI, there is a Line Buffer that can hold up to 1280*3 bytes and is used to cache the data input to the DSI in Video Mode. Users need to set a threshold through <CR_HSTX_-OUT_TH>. Then, for each row (H-Sync) of image data in Video Mode, the DSI will not send the data immediately after receiving the data input from its frontend module, but will cache it in Line Buffer first. When the pixels cached in Line Buffer reach the set threshold, the DSI starts to send data to the Display Module. In this way, when the DSI's input speed is greater than the output speed, the input data that cannot be sent out timely will be cached in Line Buffer first. When that is lower than the output speed, Line Buffer will cache some data first, and the DSI will start sending data after the pixel threshold is met. This can ensure that the time of sending the whole row of pixels by the DSI matches the time of inputting the remaining pixels. The formula for calculating the threshold is as follows:

Threshold = ceil(Width * (1 - Fdp * BPP / Fhs / LN))

- ceil() is rounded up.
- Width is the number of pixels in one row of the image.
- Fdp is the work clock of the previous-level module of DSI (data input module dp_dvp_tsrc of DSI).
- Byte Per Pixel (BPP) refers to the number of bytes in a pixel format, such as 3 for RGB888/RGB666 and 2 for RGB565/YUV422_8.
- Lane Number (LN) is the number of data lanes.
- Threshold is to be calculated and its minimum value is 6. When the calculated value is less than 6, it shall be set to 6.

12.3.11 Display Data Format

The display data formats support YUV422 (8-bit), RGB565, RGB666 (loosely packed), and RGB888, which are configured in <CR_DT>.

12.3.11.1 YUV422(8-bit)

In YUV422 (8-bit) format, the Y, U and V components of one pixel are 8 bits each, in which two pixels share a set of UV. The long packet in YUV422 (8-bit) format consists of 1-byte DI, 2-byte non-zero WC, 1-byte ECC, payload with length in WC bytes, and 2-byte CRC. With this format, pixel boundaries are aligned with some byte boundaries, and the value in WC shall be any non-zero value divisible by 4. The structure of a long packet in YUV422 (8-bit) format is shown as follows:

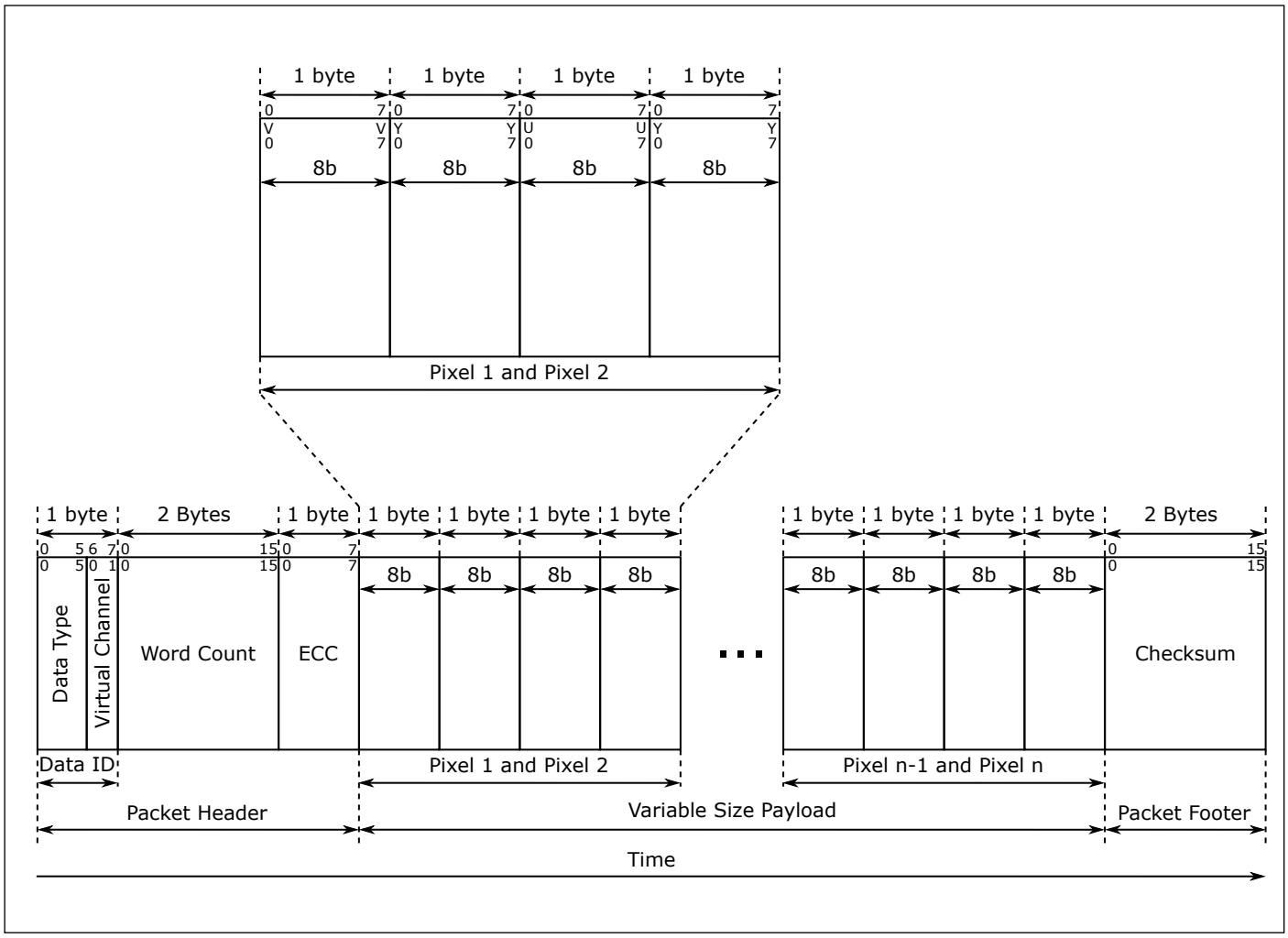


Fig. 12.37: YUV422 Pixel Format

12.3.11.2 RGB565

The R, G and B components of one pixel in RGB565 format are 5-bit, 6-bit and 5-bit respectively. The long packet in RGB565 format consists of 1-byte DI, 2-byte WC, 1-byte ECC, payload with length in WC bytes, and 2-byte CRC. With this format, the pixel boundary is aligned with the byte boundary every two bytes, and the value in WC shall be a multiple of 2. The structure of a long packet in RGB565 format is shown as follows:

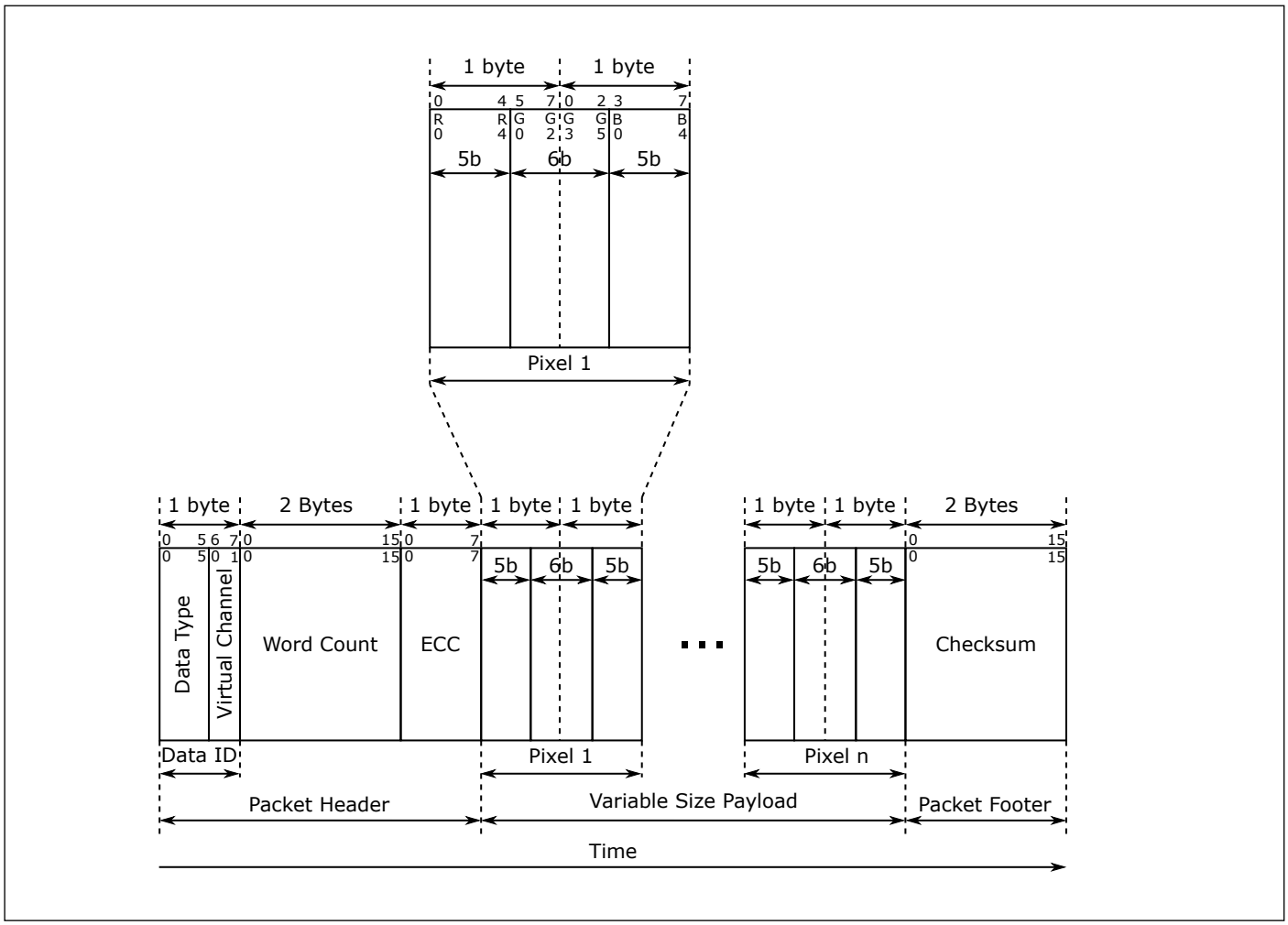


Fig. 12.38: RGB565 Pixel Format

12.3.11.3 RGB666(loosely packed)

The R, G and B components of one pixel in RGB666 (loosely packed) format are 6 bits each, but they will move to the high bit of the byte, so that the significant BPP occupies the bits [7:2] of each byte, while the bits [1:0] will be ignored. Accordingly, it takes three bytes for each pixel to be transmitted on the channel. The long packet in RGB666 format consists of 1-byte DI, 2-byte WC, 1-byte ECC, payload with length in WC bytes, and 2-byte CRC. With this format, the pixel boundary is aligned with the byte boundary every three bytes, and the value in WC shall be a multiple of 3. The structure of a long packet in RGB666 (loosely packed) format is shown as follows:

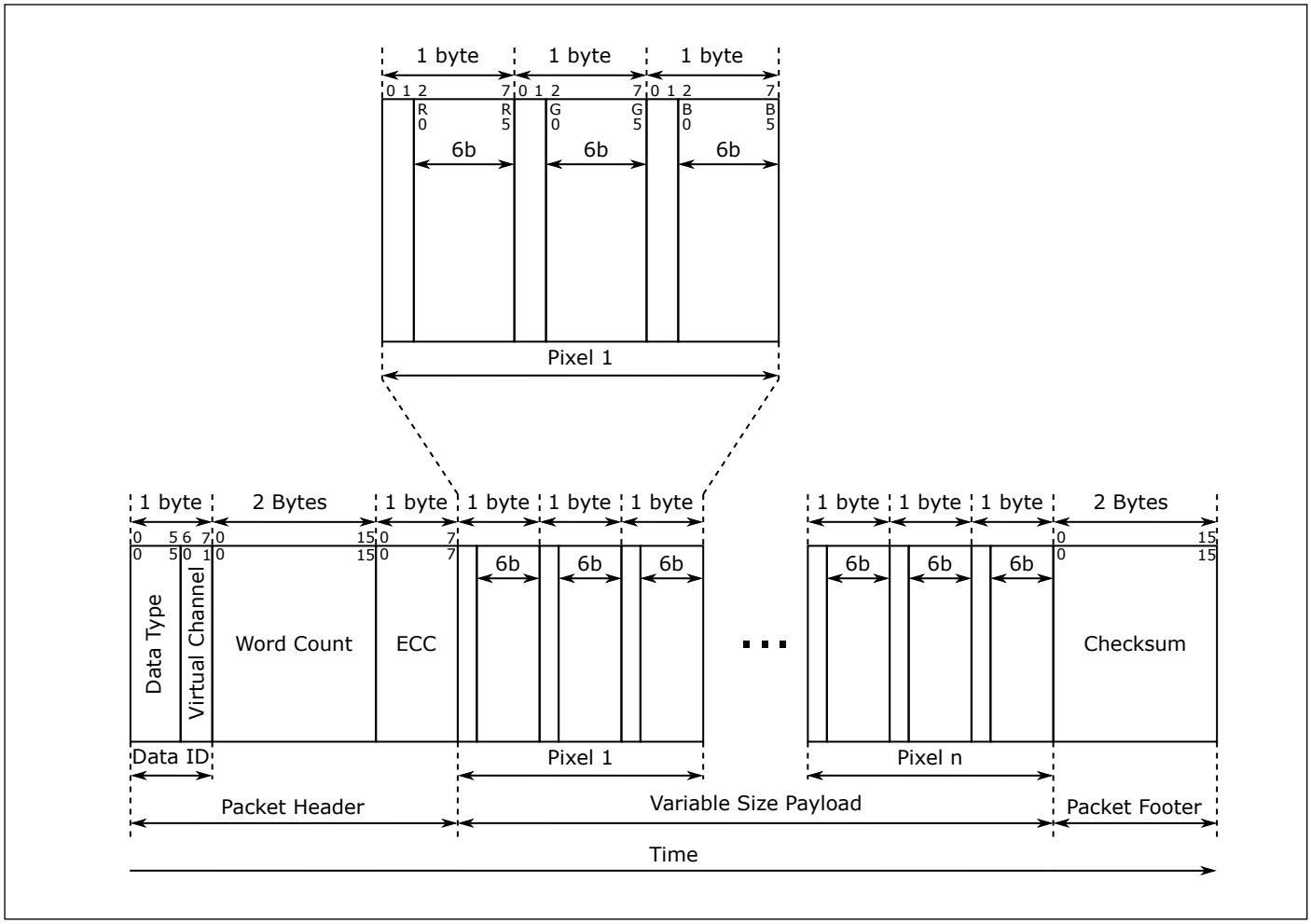


Fig. 12.39: RGB666 Pixel Format

12.3.11.4 RGB888

The R, G and B components of one pixel in RGB888 format are 8 bits each. The long packet in RGB888 format consists of 1-byte DI, 2-byte WC, 1-byte ECC, payload with length in WC bytes, and 2-byte CRC. With this format, the pixel boundary is aligned with the byte boundary every three bytes, and the value in WC shall be a multiple of 3. The structure of a long packet in RGB888 format is shown as follows:

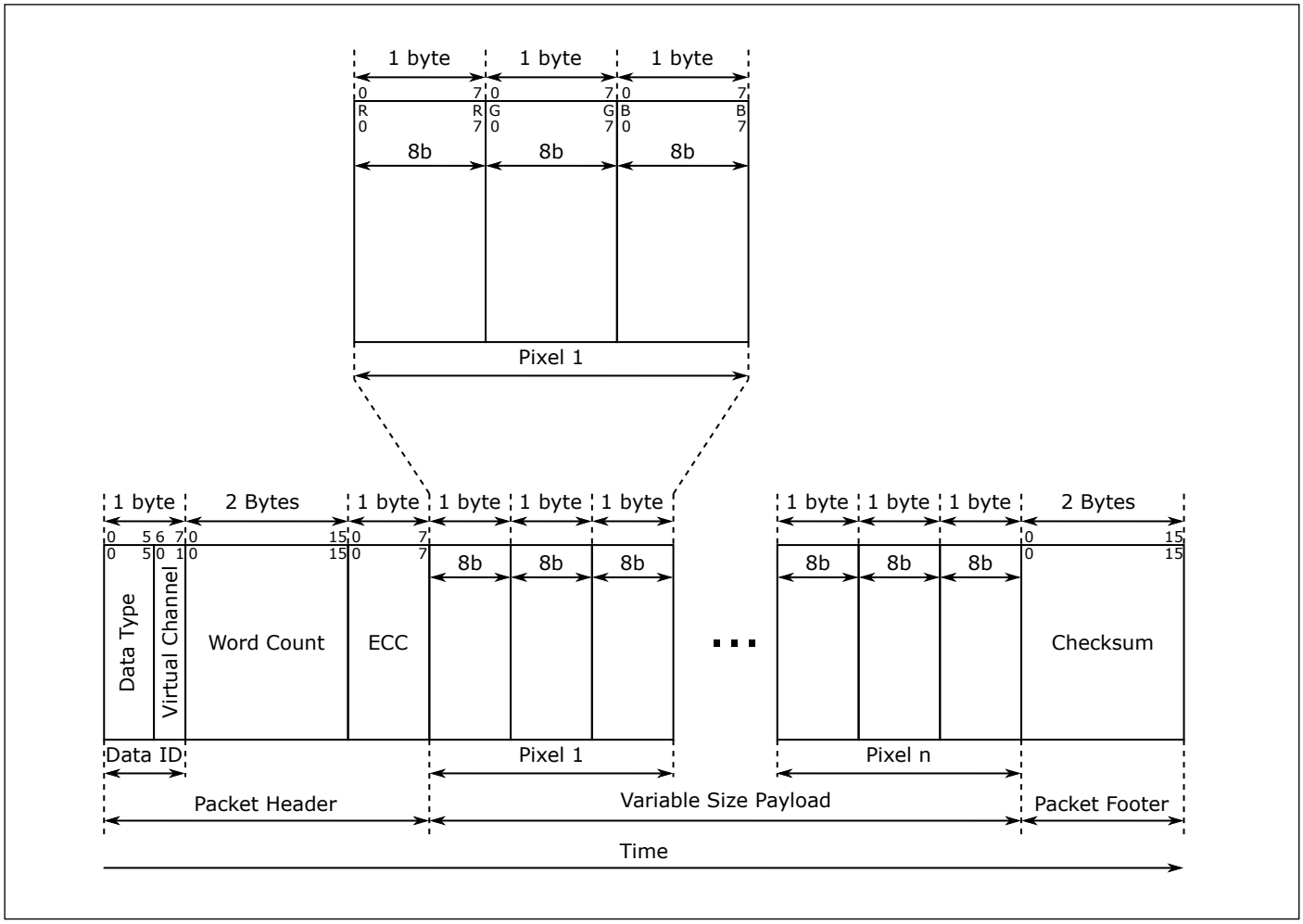


Fig. 12.40: RGB888 Pixel Format

12.3.12 Interrupt

DSI supports the following interrupts:

- TX Escape command end interrupt
- RX LPDT end interrupt
- RX ULPS command interrupt
- RX Trigger 0 command interrupt
- RX Trigger 1 command interrupt
- RX Trigger 2 command interrupt
- RX Trigger 3 command interrupt
- TX LPDT FIFO request interrupt
- RX LPDT FIFO request interrupt

- Buffer Overrun error interrupt
- Buffer Underrun error interrupt
- Too few pixel error interrupt
- Excessive pixel error interrupt
- FIFO overflow error interrupt

The TX Escape command end interrupt is triggered when any Escape command is sent.

RX LPDT end interrupt, RX ULPS command interrupt, RX Trigger 0 command interrupt, RX Trigger 1 command interrupt, RX Trigger 2 command interrupt, and RX Trigger 3 command interrupt will be triggered after the corresponding command is sent out and the ACK replied by the other side is received.

When TFICNT in DSI_FIFO_CONFIG_1 is greater than TFITH, the TX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not met.

When RFICNT in DSI_FIFO_CONFIG_1 is greater than RFITH, the RX FIFO request interrupt will be generated, and the interrupt flag will be automatically cleared when the condition is not met.

The Buffer Overrun error interrupt will be generated at Line Buffer Overrun, which means that the data cached in Line Buffer cannot be sent out timely and Line Buffer is full.

The Buffer Underrun error interrupt will be generated at Line Buffer Underrun, which means that the data in Line Buffer is completely read out and no new data is filled in timely.

The “too few pixel error interrupt” is be generated when the number of pixels sent during HSync is smaller than the value configured by <CR_HSTX_PC>.

The “excessive pixel error interrupt” is generated when the number of pixels sent during HSync is greater than the value configured by <CR_HSTX_PC>.

The FIFO overflow error interrupt is be generated in the case of Overflow or Underflow of TX or RX.

12.3.13 DMA

DSI LPDT supports the DMA transfer mode, which requires to set the thresholds of TX FIFO and RX FIFO respectively by configuring the <TFITH> and <RFITH> bits in the register DSI_FIFO_CONFIG_1. When this mode is enabled, if <TFICNT> is greater than <TFITH>, the DMA TX request is triggered. After DMA is configured, when receiving this request, DMA transfers data from memory to TX FIFO as configured. If <RFICNT> is greater than <RFITH>, the DMA RX request will be triggered. After DMA is configured, when receiving this request, DMA transfers the data in RX FIFO to the memory as configured.

12.3.14 Use with OSD

The input data of DSI can be configured to be processed by the OSD first. See the OSD Module section for functional description.

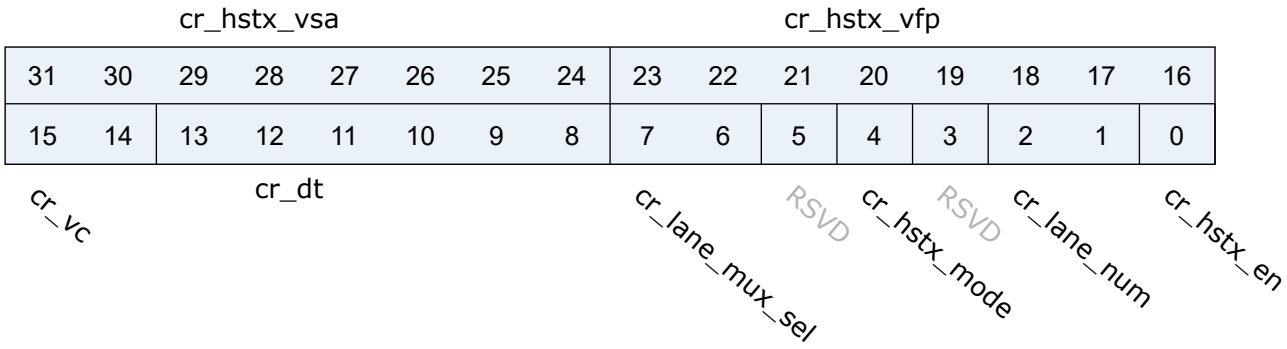
12.4 Register description

Name	Description
dsi_config	
dsi_esc_config	
dsi_lpdt_tx_config	
dsi_hstx_config	
dsi_int_status	
dsi_int_mask	
dsi_int_clear	
dsi_int_enable	
dsi_fifo_config_0	
dsi_fifo_config_1	
dsi_fifo_wdata	
dsi_fifo_rdata	
dphy_config_0	
dphy_config_1	
dphy_config_2	
dphy_config_3	
dphy_config_4	
dphy_config_5	
dphy_config_6	
dphy_config_7	
dphy_config_8	
dphy_config_9	
dphy_config_10	
dphy_config_11	

Name	Description
dphy_config_12	
dphy_config_13	
dphy_config_14	
dphy_config_15	
dphy_config_16	
dummy_reg	

12.4.1 dsi_config

Address：0x3001a100

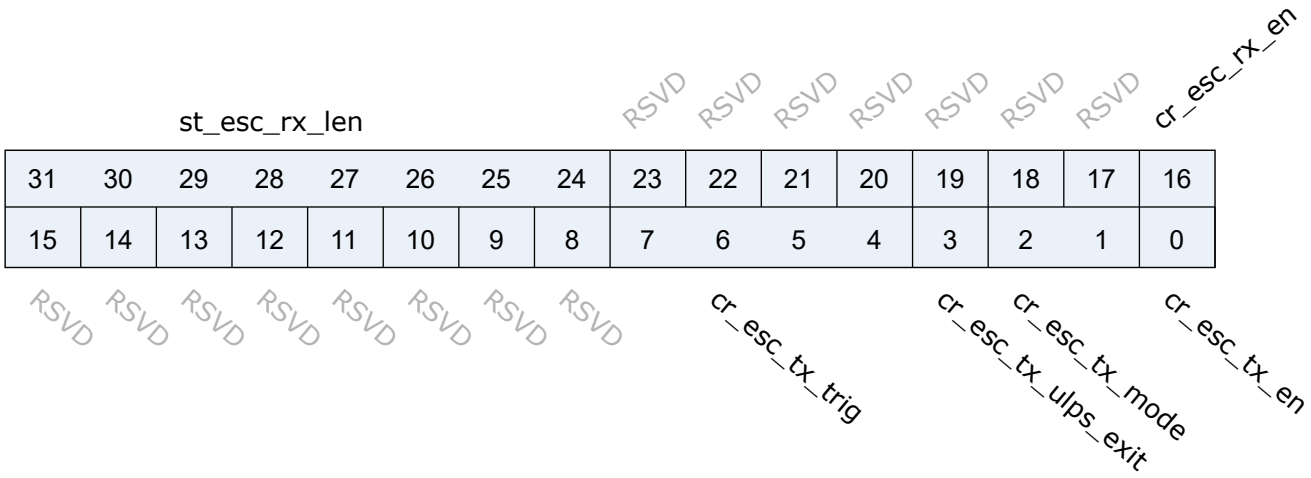


Bits	Name	Type	Reset	Description
31:24	cr_hstx_vsa	r/w	8'd2	HSTX Vertical Sync Active width
23:16	cr_hstx_vfp	r/w	8'd2	HSTX Vertical Front Porch width
15:14	cr_vc	r/w	2'd0	Virtual Channel number
13:8	cr_dt	r/w	6'h3E	Data Type 6'h2C: YUV422, 8-bit 6'h0E: RGB565 6'h2E: RGB666, loosely packed 6'h3E: RGB888 Others: Reserved
7:6	cr_lane_mux_sel	r/w	2'd0	Controls lane order 2'd0: lane3, lane2, lane1, lane0 (no lane is switched) 2'd1: lane2, lane1, lane3, lane0 2'd2: lane1, lane3, lane2, lane0 2'd3: lane3, lane1, lane2, lane0

Bits	Name	Type	Reset	Description
5	RSVD			
4	cr_hstx_mode	r/w	1'b1	HSTX mode select 1'b0: Sync Event mode 1'b1: Sync Pulse mode
3	RSVD			
2:1	cr_lane_num	r/w	2'd0	Lane number configuration 2'd0: 1-lane MIPI TX 2'd1: 2-lane MIPI TX 2'd2: 4-lane MIPI TX 2'd3: Reserved
0	cr_hstx_en	r/w	1'b0	HSTX function enable signal

12.4.2 dsi_esc_config

Address：0x3001a104



Bits	Name	Type	Reset	Description
31:24	st_esc_rx_len	r	8'd0	Escape RX length received (Include packet header/data/footer) (unit: byte)
23:17	RSVD			
16	cr_esc_rx_en	r/w	1'b0	Escape RX enable signal
15:8	RSVD			
7:4	cr_esc_tx_trig	r/w	4'h0	Escape TX trigger command
3	cr_esc_tx_ulps_exit	w1p	1'b0	Escape TX ULPS exit signal trigger

Bits	Name	Type	Reset	Description
2:1	cr_esc_tx_mode	r/w	2'd0	Escape TX mode select 2'd0: LPDT 2'd1: Trigger command 2'd2: ULPS Enable 2'd3: ULPS Disable
0	cr_esc_tx_en	w1p	1'b0	Escape TX enable signal

12.4.3 dsi_lpdt_tx_config

Address：0x3001a108

cr_lpdt_word1 cr_lpdt_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

cr_lpdt_di cr_lpdt_vc cr_lpdt_pdlen

Bits	Name	Type	Reset	Description
31:24	cr_lpdt_word1	r/w	8'h0	LPDT TX word 1
23:16	cr_lpdt_word0	r/w	8'h0	LPDT TX word 0
15:10	cr_lpdt_di	r/w	6'h0	LPDT TX data identifier
9:8	cr_lpdt_vc	r/w	2'h0	LPDT TX virtual channel
7:0	cr_lpdt_pdlen	r/w	8'd0	LPDT TX packet data length (exclude packet header & footer) (unit: byte)

12.4.4 dsi_hstx_config

Address：0x3001a10c

RSVD RSVD RSVD RSVD RSVD cr_hstx_out_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

RSVD RSVD RSVD RSVD RSVD cr_hstx_pc

Bits	Name	Type	Reset	Description
31:27	RSVD			
26:16	cr_hstx_out_th	r/w	11'd840	Line buffer threshold for controller to start transmitting each line (unit: pixel) Formula: th = ceil(W * (1 - Fdp*BPP/Fhs/Ln)) th: cr_hstx_out_th W: Frame width Fdp: Display (dp_dvp_tsrc) clock rate Fhs: DSI byte clock rate (dsi_bit_clk/8 or mipipll_clk/8) BPP: Byte-per-pixel (equals 3 for RGB888 & RGB666; equals 2 for RGB565 & YUV422_8) Ln: DSI lane number (controlled by cr_lane_num) Note: The minimum value is 6 for synchronization concern (Set to 6 if the formula result is negative or less than 6)
15:11	RSVD			
10:0	cr_hstx_pc	r/w	11'd1280	Pixel count of each line (frame width) (unit:pixel) Note: Pixel count should not exceed 1280 (720p) and should be a multiple of 4

12.4.5 dsi_int_status

Address：0x3001a110

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

dsi_int_status

RSVD RSVD

Bits	Name	Type	Reset	Description
31:14	RSVD			

Bits	Name	Type	Reset	Description
13:0	dsi_int_status	r	14'h0	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

12.4.6 dsi_int_mask

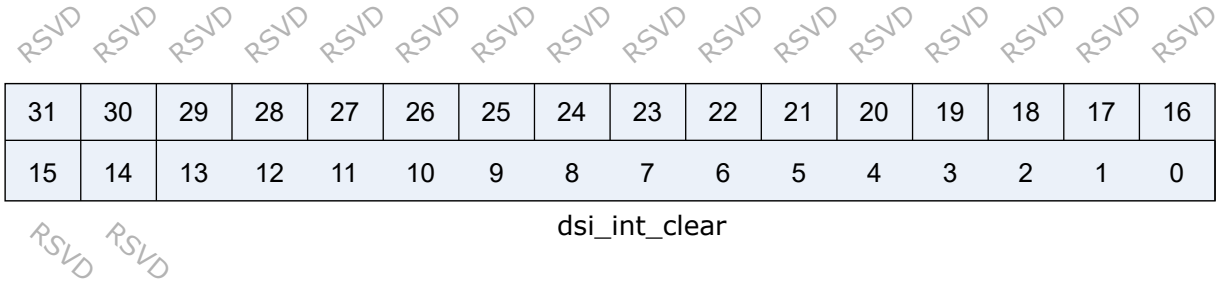
Address：0x3001a114

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	dsi_int_mask													

Bits	Name	Type	Reset	Description
31:14	RSVD			
13:0	dsi_int_mask	r/w	14'h3FFF	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

12.4.7 dsi_int_clear

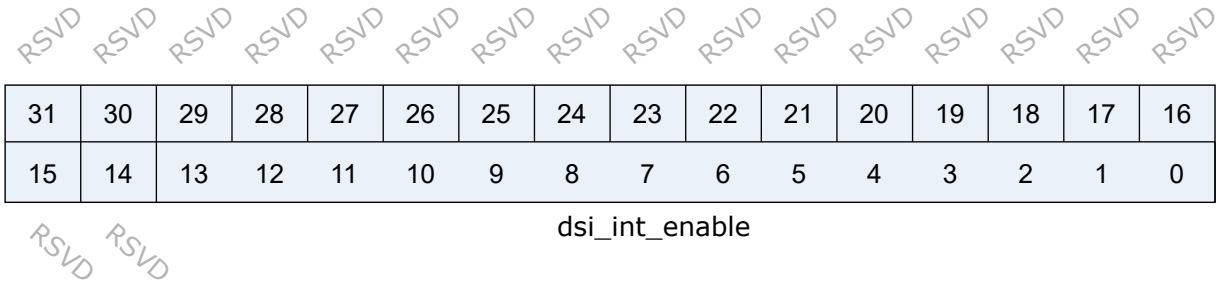
Address：0x3001a118



Bits	Name	Type	Reset	Description
31:14	RSVD			
13:0	dsi_int_clear	w1p	14'h0	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

12.4.8 dsi_int_enable

Address：0x3001a11c

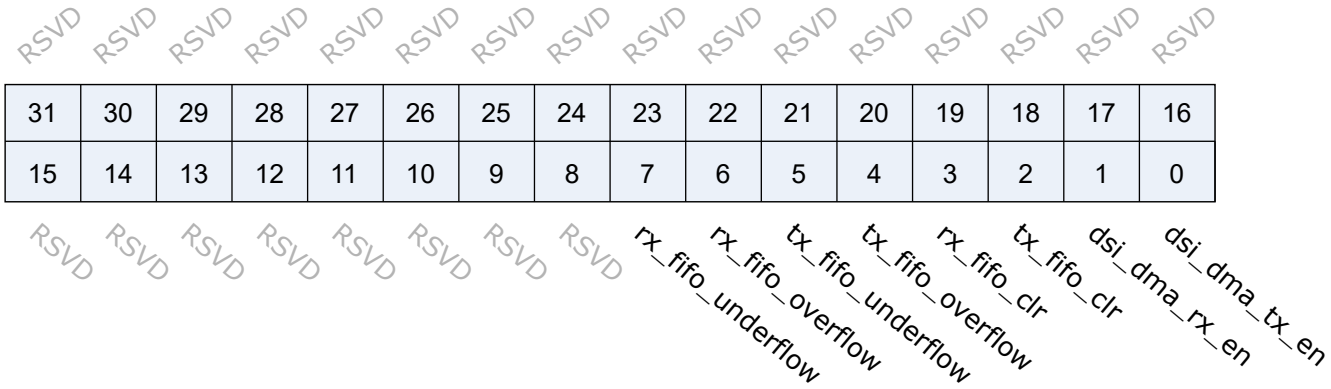


Bits	Name	Type	Reset	Description
31:14	RSVD			

Bits	Name	Type	Reset	Description
13:0	dsi_int_enable	r/w	14'h3FFF	[13]: FIFO Error (check 0x60[7:4] for detail) [12]: Pixel Count Too Large Error [11]: Pixel Count Too Small Error [10]: Buffer Underrun Error [9]: Buffer Overrun Error [8]: RX LPDT FIFO ready [7]: TX LPDT FIFO ready [6:3]: RX Trigger Command [2]: RX ULPS Command [1]: RX LPDT End [0]: TX Escape Command End

12.4.9 dsi_fifo_config_0

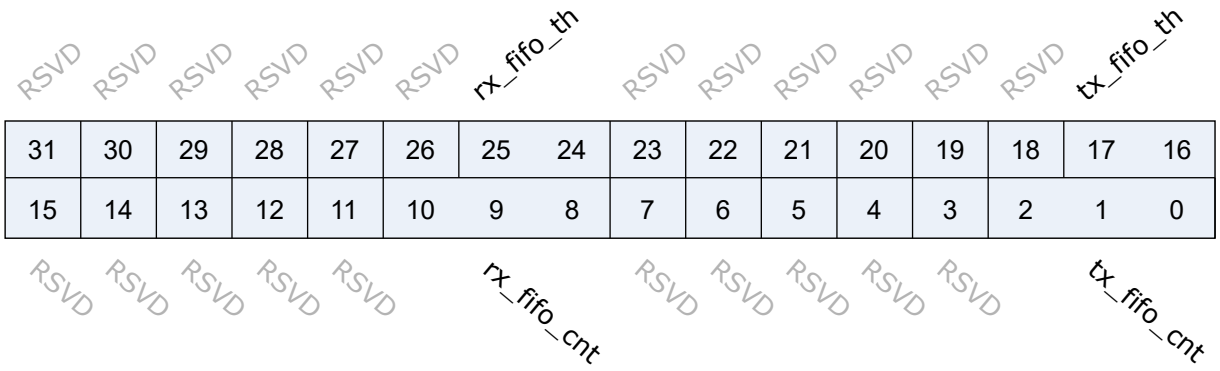
Address：0x3001a160



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	w1p	1'b0	Clear signal of RX FIFO
2	tx_fifo_clr	w1p	1'b0	Clear signal of TX FIFO
1	dsi_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface
0	dsi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

12.4.10 dsi_fifo_config_1

Address：0x3001a164



Bits	Name	Type	Reset	Description
31:26	RSVD			
25:24	rx_fifo_th	r/w	2'd0	RX FIFO threshold, dma_rx_req will not be asserted if tx_-fifo_cnt is less than this value
23:18	RSVD			
17:16	tx_fifo_th	r/w	2'd0	TX FIFO threshold, dma_tx_req will not be asserted if tx_-fifo_cnt is less than this value
15:11	RSVD			
10:8	rx_fifo_cnt	r	3'd0	RX FIFO available count
7:3	RSVD			
2:0	tx_fifo_cnt	r	3'd4	TX FIFO available count

12.4.11 dsi_fifo_wdata

Address：0x3001a168

dsi_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

dsi_fifo_wdata

Bits	Name	Type	Reset	Description
31:0	dsi_fifo_wdata	w	x	

12.4.12 dsi_fifo_rdata

Address：0x3001a16c

dsi_fifo_rdata

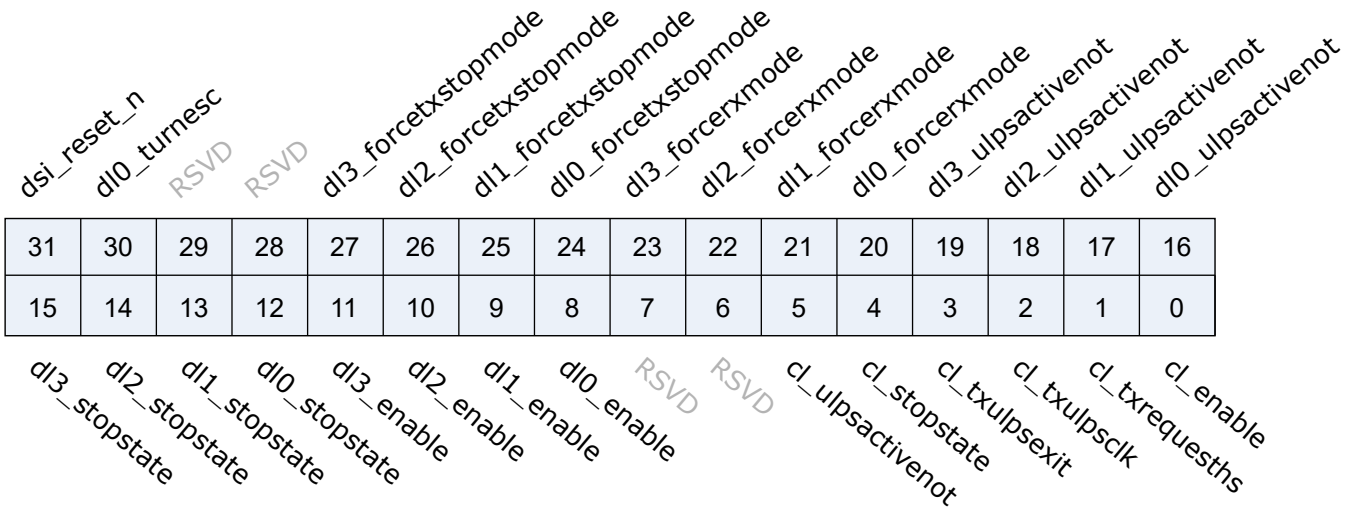
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

dsi_fifo_rdata

Bits	Name	Type	Reset	Description
31:0	dsi_fifo_rdata	r	32'h0	

12.4.13 dphy_config_0

Address：0x3001a180

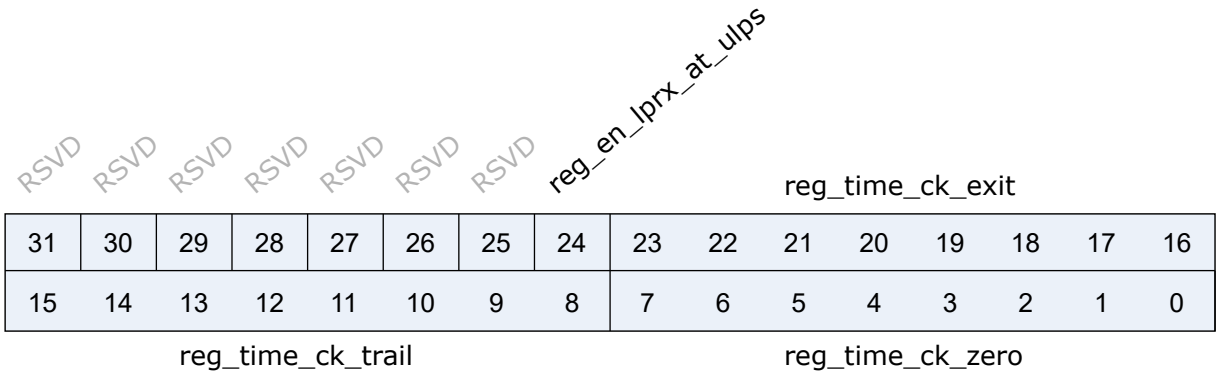


Bits	Name	Type	Reset	Description
31	dsi_reset_n	r/w	1'b0	MIPI DSI D-PHY reset pin
30	dl0_turnesc	w1p	1'b0	Data lane0 Bus Turnaround
29:28	RSVD			
27	dl3_forcetxstopmode	r/w	1'b0	Force Lane3 to Generate Stop State
26	dl2_forcetxstopmode	r/w	1'b0	Force Lane2 to Generate Stop State
25	dl1_forcetxstopmode	r/w	1'b0	Force Lane1 to Generate Stop State
24	dl0_forcetxstopmode	r/w	1'b0	Force Lane0 to Generate Stop State

Bits	Name	Type	Reset	Description
23	dl3_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane3 immediately transitions to receive mode.
22	dl2_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane2 immediately transitions to receive mode.
21	dl1_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane1 immediately transitions to receive mode.
20	dl0_forcerxmode	r/w	1'b0	Enables the reverse escape LP receiver. Lane0 immediately transitions to receive mode.
19	dl3_ulpsactivenot	r	1'b1	Data lane3 is NOT in the ULP state
18	dl2_ulpsactivenot	r	1'b1	Data lane2 is NOT in the ULP state
17	dl1_ulpsactivenot	r	1'b1	Data lane1 is NOT in the ULP state
16	dl0_ulpsactivenot	r	1'b1	Data lane0 is NOT in the ULP state
15	dl3_stopstate	r	1'b1	Data lane3 is in Stop state
14	dl2_stopstate	r	1'b1	Data lane2 is in Stop state
13	dl1_stopstate	r	1'b1	Data lane1 is in Stop state
12	dl0_stopstate	r	1'b1	Data lane0 is in Stop state
11	dl3_enable	r/w	1'b0	Data lane3 enable
10	dl2_enable	r/w	1'b0	Data lane2 enable
9	dl1_enable	r/w	1'b0	Data lane1 enable
8	dl0_enable	r/w	1'b0	Data lane0 enable
7:6	RSVD			
5	cl_ulpsactivenot	r	1'b1	Clock lane is NOT in the ULP state
4	cl_stopstate	r	1'b1	Clock lane is in Stop state
3	cl_txulpsexit	w1p	1'b0	Clock lane Transmit ULP Exit Sequence
2	cl_txulpsclk	r/w	1'b0	Clock lane Transmit Ultra-Low Power State
1	cl_txrequesths	r/w	1'b0	Clock lane High-Speed Transmit Request
0	cl_enable	r/w	1'b0	Clock lane enable

12.4.14 dphy_config_1

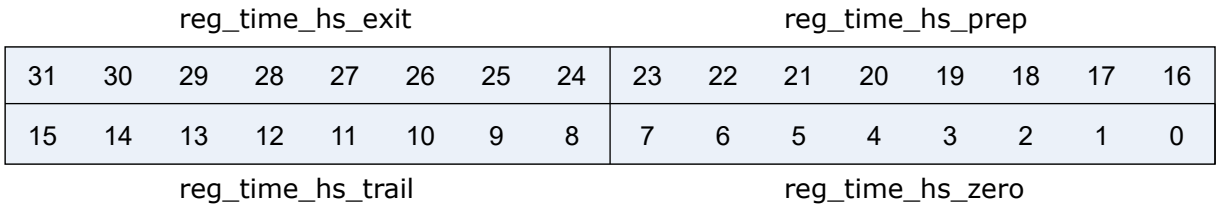
Address：0x3001a184



Bits	Name	Type	Reset	Description
31:25	RSVD			
24	reg_en_lprx_at_ulps	r/w	1'b0	MIPI DSI D-PHY control register - reg_en_lprx_at_ulps
23:16	reg_time_ck_exit	r/w	8'h5	MIPI DSI D-PHY control register - reg_time_ck_exit (tx_clk_esc) txclkesc: 40M, datarate: 800Mbps
15:8	reg_time_ck_trail	r/w	8'h3	MIPI DSI D-PHY control register - reg_time_ck_trail (tx_clk_esc)
7:0	reg_time_ck_zero	r/w	8'hF	MIPI DSI D-PHY control register - reg_time_ck_zero (tx_clk_esc)

12.4.15 dphy_config_2

Address：0x3001a188



Bits	Name	Type	Reset	Description
31:24	reg_time_hs_exit	r/w	8'h5	MIPI DSI D-PHY control register - reg_time_hs_exit (tx_clk_esc)
23:16	reg_time_hs_prep	r/w	8'h2	MIPI DSI D-PHY control register - reg_time_hs_prep (tx_clk_esc)

Bits	Name	Type	Reset	Description
15:8	reg_time_hs_trail	r/w	8'h3	MIPI DSI D-PHY control register - reg_time_hs_trail (tx_clk_esc)
7:0	reg_time_hs_zero	r/w	8'h5	MIPI DSI D-PHY control register - reg_time_hs_zero (tx_clk_esc)

12.4.16 dphy_config_3

Address：0x3001a18c

reg_time_lpx								reg_time_reqrdy							
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_time_ta_get								reg_time_ta_go							

Bits	Name	Type	Reset	Description
31:24	reg_time_lpx	r/w	8'h3	MIPI DSI D-PHY control register - reg_time_lpx (tx_clk_esc)
23:16	reg_time_reqrdy	r/w	8'h0	MIPI DSI D-PHY control register - reg_time_reqrdy
15:8	reg_time_ta_get	r/w	8'hF	MIPI DSI D-PHY control register - reg_time_ta_get (tx_clk_esc)
7:0	reg_time_ta_go	r/w	8'hC	MIPI DSI D-PHY control register - reg_time_ta_go (tx_clk_esc)

12.4.17 dphy_config_4

Address：0x3001a190

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
reg_time_wakeup															

Bits	Name	Type	Reset	Description
31:16	RSVD			

Bits	Name	Type	Reset	Description
15:0	reg_time_wakeup	r/w	16'h9C41	MIPI DSI D-PHY control register - reg_time_wakeup (tx_-clk_esc)

12.4.18 dphy_config_5

Address：0x3001a194

reg_trig0_code								reg_trig1_code							
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_trig2_code								reg_trig3_code							

Bits	Name	Type	Reset	Description
31:24	reg_trig0_code	r/w	8'b0100_-0110	MIPI DSI D-PHY control register - reg_trig0_code
23:16	reg_trig1_code	r/w	8'b1011_-1010	MIPI DSI D-PHY control register - reg_trig1_code
15:8	reg_trig2_code	r/w	8'b1000_-0100	MIPI DSI D-PHY control register - reg_trig2_code
7:0	reg_trig3_code	r/w	8'b0000_-0101	MIPI DSI D-PHY control register - reg_trig3_code

12.4.19 dphy_config_6

Address：0x3001a198

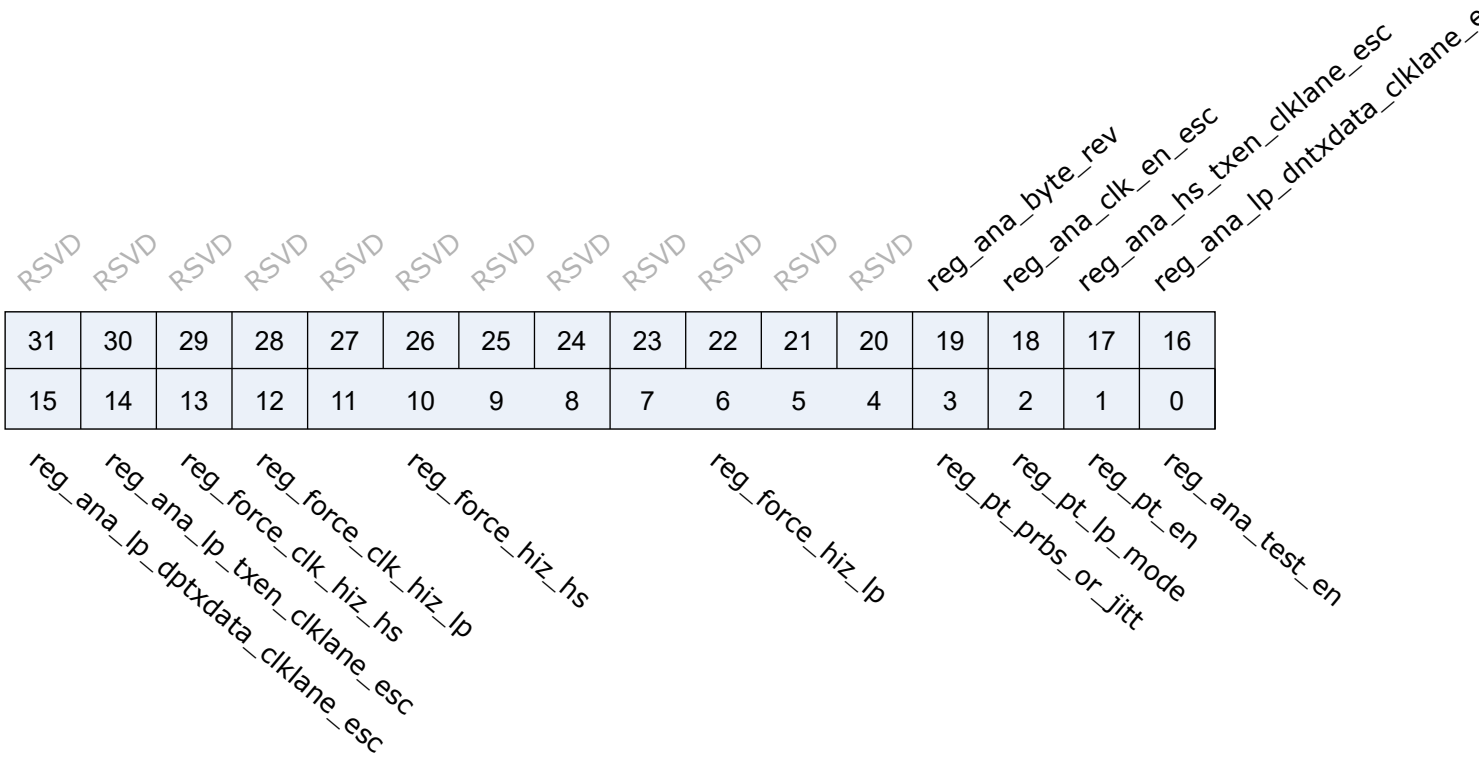
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reg_lpdt_code								reg_ulps_code							

Bits	Name	Type	Reset	Description
31:16	RSVD			

Bits	Name	Type	Reset	Description
15:8	reg_lpdt_code	r/w	8'b1000_-0111	MIPI DSI D-PHY control register - reg_lpdt_code
7:0	reg_ulps_code	r/w	8'b0111_-1000	MIPI DSI D-PHY control register - reg_ulps_code

12.4.20 dphy_config_7

Address：0x3001a19c

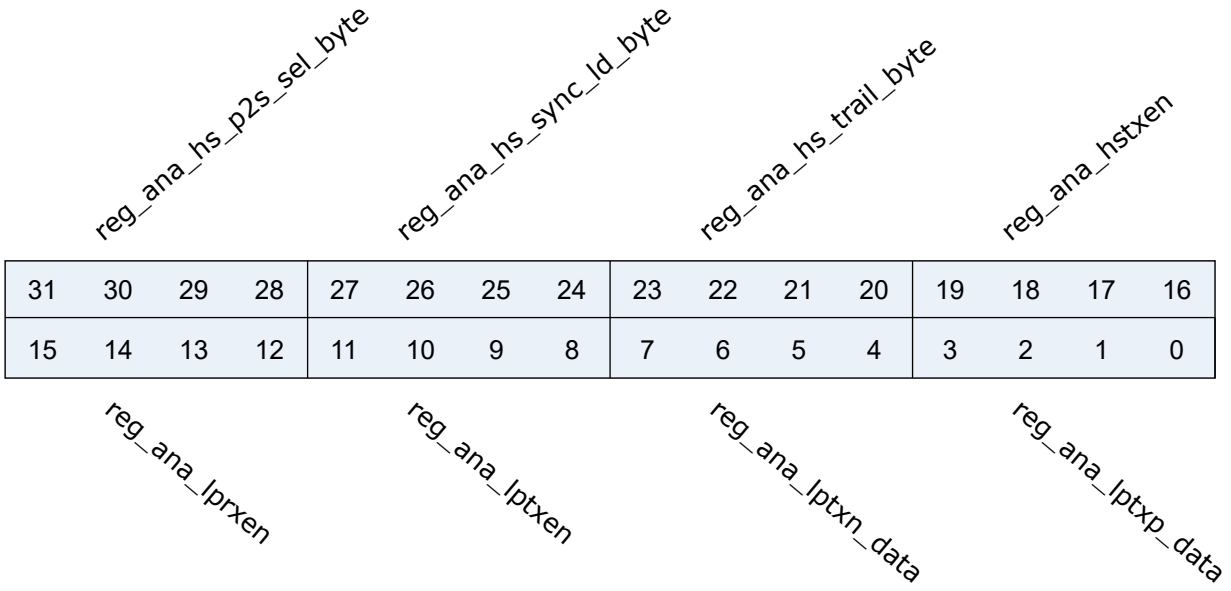


Bits	Name	Type	Reset	Description
31:20	RSVD			
19	reg_ana_byte_rev	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_byte_rev
18	reg_ana_clk_en_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_clk_en_esc
17	reg_ana_hs_txen_clklane_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_hs_txen_clk-lane_esc
16	reg_ana_lp_dntxdata_clklane_-esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_lp_dntxdata_-clklane_esc

Bits	Name	Type	Reset	Description
15	reg_ana_lp_dptxdata_clklane_-esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_lp_dptxdata_-clklane_esc
14	reg_ana_lp_txen_clklane_esc	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_lp_txen_clk-lane_esc
13	reg_force_clk_hiz_hs	r/w	1'b0	MIPI DSI D-PHY control register - reg_force_clk_hiz_hs
12	reg_force_clk_hiz_lp	r/w	1'b0	MIPI DSI D-PHY control register - reg_force_clk_hiz_lp
11:8	reg_force_hiz_hs	r/w	4'h0	MIPI DSI D-PHY control register - reg_force_hiz_hs
7:4	reg_force_hiz_lp	r/w	4'h0	MIPI DSI D-PHY control register - reg_force_hiz_lp
3	reg_pt_prbs_or_jitt	r/w	1'b0	MIPI DSI D-PHY control register - reg_pt_prbs_or_jitt
2	reg_pt_lp_mode	r/w	1'b0	MIPI DSI D-PHY control register - reg_pt_lp_mode
1	reg_pt_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_pt_en
0	reg_ana_test_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_test_en

12.4.21 dphy_config_8

Address：0x3001a1a0



Bits	Name	Type	Reset	Description
31:28	reg_ana_hs_p2s_sel_byte	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hs_p2s_sel_-byte
27:24	reg_ana_hs_sync_ld_byte	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hs_sync_ld_-byte

Bits	Name	Type	Reset	Description
23:20	reg_ana_hs_trail_byte	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hs_trail_byte
19:16	reg_ana_hstxen	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_hstxen
15:12	reg_ana_lprxen	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lprxen
11:8	reg_ana_lptxen	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lptxen
7:4	reg_ana_lptxn_data	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lptxn_data
3:0	reg_ana_lptxp_data	r/w	4'h0	MIPI DSI D-PHY control register - reg_ana_lptxp_data

12.4.22 dphy_config_9

Address：0x3001a1a4

reg_ana_hs_data_out_byte

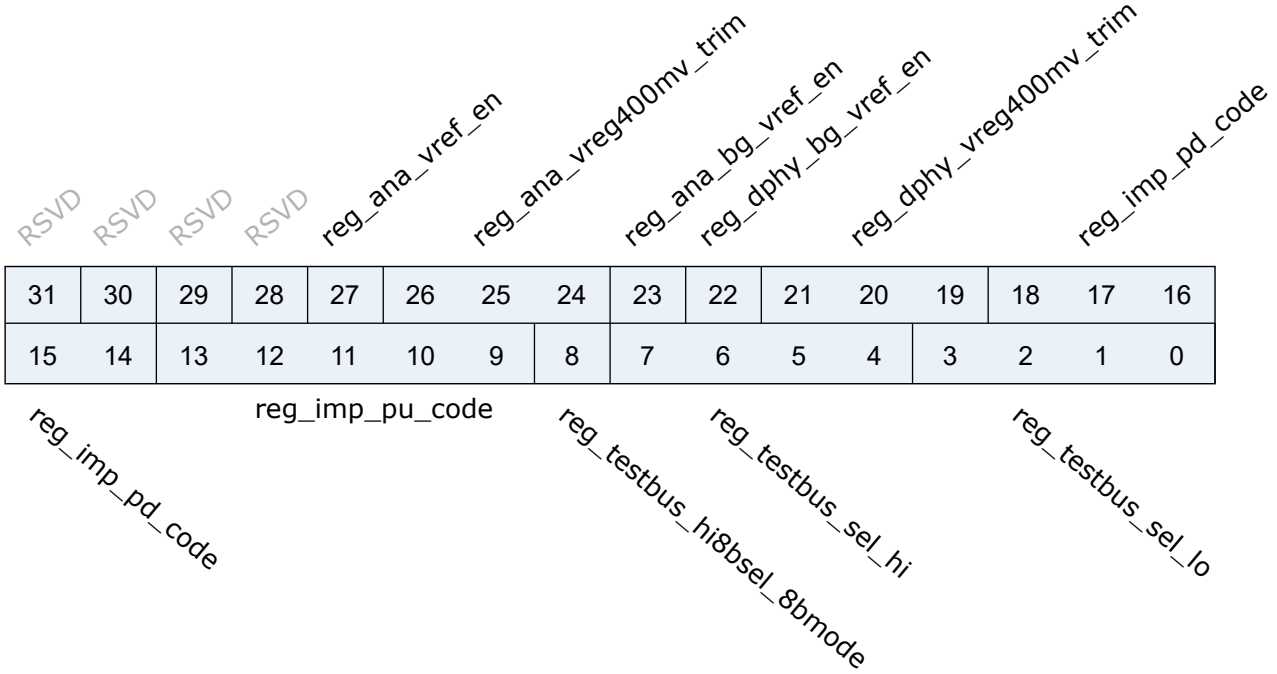
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

reg_ana_hs_data_out_byte

Bits	Name	Type	Reset	Description
31:0	reg_ana_hs_data_out_byte	r/w	32'h0	MIPI DSI D-PHY control register - reg_ana_hs_data_out_-byte

12.4.23 dphy_config_10

Address：0x3001a1a8



Bits	Name	Type	Reset	Description
31:28	RSVD			
27	reg_ana_vref_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_vref_en
26:24	reg_ana_vreg400mv_trim	r/w	3'h0	MIPI DSI D-PHY control register - reg_ana_vreg400mv_-trim
23	reg_ana_bg_vref_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_ana_bg_vref_en
22	reg_dphy_bg_vref_en	r/w	1'b0	MIPI DSI D-PHY control register - reg_dphy_bg_vref_en
21:19	reg_dphy_vreg400mv_trim	r/w	3'h0	MIPI DSI D-PHY control register - reg_dphy_vreg400mv_-trim
18:14	reg_imp_pd_code	r/w	5'h9	MIPI DSI D-PHY control register - reg_imp_pd_code
13:9	reg_imp_pu_code	r/w	5'h8	MIPI DSI D-PHY control register - reg_imp_pu_code
8	reg_testbus_hi8bsel_8bmode	r/w	1'b0	MIPI DSI D-PHY control register - reg_testbus_hi8bsel_-8bmode
7:4	reg_testbus_sel_hi	r/w	4'h0	MIPI DSI D-PHY control register - reg_testbus_sel_hi
3:0	reg_testbus_sel_lo	r/w	4'h0	MIPI DSI D-PHY control register - reg_testbus_sel_lo

12.4.24 dphy_config_11

Address：0x3001a1ac

reg_dsi_ana_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

reg_dsi_ana_0

Bits	Name	Type	Reset	Description
31:16	reg_dsi_ana_1	r/w	16'h0	MIPI DSI D-PHY control register - reg_dsi_ana_1
15:0	reg_dsi_ana_0	r/w	16'hc14	MIPI DSI D-PHY control register - reg_dsi_ana_0

12.4.25 dphy_config_12

Address：0x3001a1b0

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

reg_dsi_ana_2

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	reg_dsi_ana_2	r/w	16'h0	MIPI DSI D-PHY control register - reg_dsi_ana_2

12.4.26 dphy_config_13

Address：0x3001a1b4

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

reg_rd_dig_test_bus

Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	reg_rd_dig_test_bus	r	16'h0	MIPI DSI D-PHY control register - reg_rd_dig_test_bus

12.4.27 dphy_config_14

Address：0x3001a1b8

reg_pt_free_rep_pat

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_pt_free_rep_pat

Bits	Name	Type	Reset	Description
31:0	reg_pt_free_rep_pat	r/w	32'h87654321	MIPI DSI D-PHY control register - reg_pt_free_rep_pat

12.4.28 dphy_config_15

Address：0x3001a1bc

RSVD RSVD reg_csi_rst_n_pre reg_dsi_rst_n_pre RSVD RSVD RSVD reg_mipi_ldo_fast RSVD RSVD RSVD reg_ten_dsi_ldo RSVD reg_dphy_ldo11_rfb_sw

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

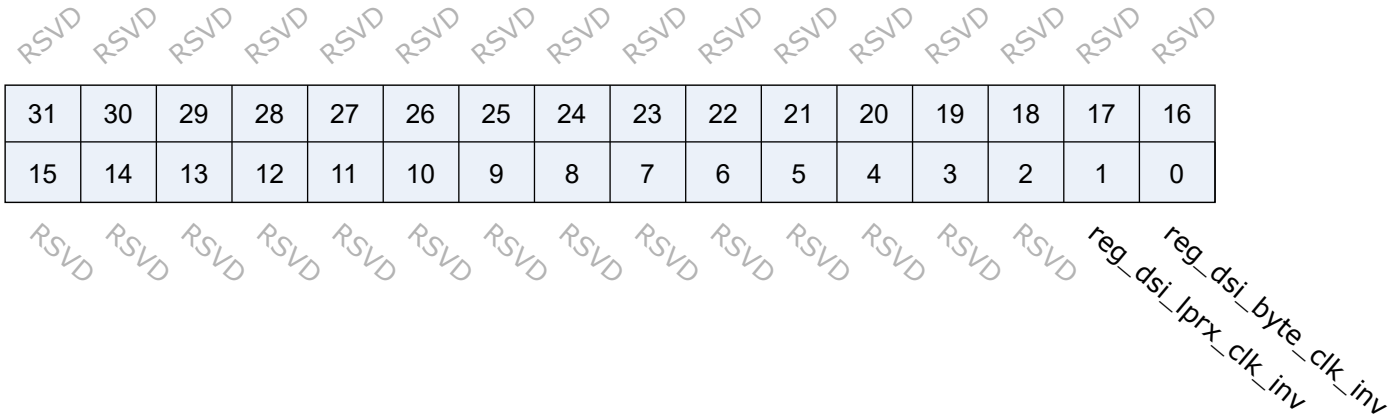
RSVD RSVD RSVD reg_dphy_short_ldo11 RSVD RSVD RSVD reg_dphy_pu_ldo11 RSVD RSVD RSVD reg_dsi_pw_avdd1815 RSVD RSVD RSVD reg_dsi_dc_tp_out_en

Bits	Name	Type	Reset	Description
31:30	RSVD			

Bits	Name	Type	Reset	Description
29	reg_csi_rst_n_pre	r/w	1'b0	Note: reg_csi_rst_n_pre should be released at least 2 us BEFORE releasing csi_reset_n
28	reg_dsi_rst_n_pre	r/w	1'b0	Note: reg_dsi_rst_n_pre should be released at least 2 us BEFORE releasing dsi_reset_n
27:25	RSVD			
24	reg_mipi_ldo_fast	r/w	1'b0	
23:21	RSVD			
20	reg_ten_dsi_ldo	r/w	1'b0	
19	RSVD			
18:16	reg_dphy_ldo11_rfb_sw	r/w	3'd4	
15:13	RSVD			
12	reg_dphy_short_ldo11	r/w	1'b0	
11:9	RSVD			
8	reg_dphy_pu_ldo11	r/w	1'b1	enable LDO11 for both dsi and csi
7:5	RSVD			
4	reg_dsi_pw_avdd1815	r/w	1'b0	0: power switch on
3:1	RSVD			
0	reg_dsi_dc_tp_out_en	r/w	1'b0	

12.4.29 dphy_config_16

Address：0x3001a1c0



Bits	Name	Type	Reset	Description
31:2	RSVD			
1	reg_dsi_lprx_clk_inv	r/w	1'b0	
0	reg_dsi_byte_clk_inv	r/w	1'b0	

12.4.30 dummy_reg

Address：0x3001a1fc

dummy_reg

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

dummy_reg

Bits	Name	Type	Reset	Description
31:0	dummy_reg	r/w	32'h0	Dummy registers

13 CAM

13.1 Overview

The Camera (CAM) module converts the parallel interface (DVP) into the general bus interface (AHB), and writes the pixel data generated by the image sensor into the system memory in a flat or packaged format for subsequent image transfer or compression. With flexible output format configuration, CAM can meet various image processing requirements.

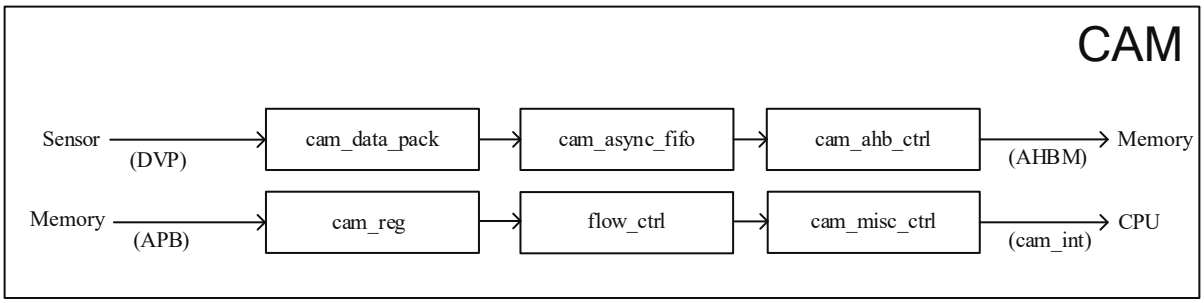


Fig. 13.1: Block Diagram of Cam

13.2 Features

- Parallel interface with 8-bit DVP signal, high-speed data transfer (80M), and configurable valid level and logical combination of DVP signal
- Supports 8-bit/16-bit/24-bit input pixel bit width
- Supports up to 36 data input sources
- Supports conversion of RGB888 input format to RGB565 or RGBA8888 output format
- Supports movie and photo modes
- Configurable discard modes:
  - Discard odd-bit data

  - Discard even-bit data
  - Discard odd-bit data of odd lines
  - Discard even-bit data of odd lines
- Configurable line and frame synchronization signal selection and polarity selection of image sensor
- Supports rectangle cropping of image
- Supports integrity test of line and frame synchronization signals
- AHB bus communication interface
- FIFO with 512-byte cache to assist occasional busy bus
- Caches up to 4 images continuously
- Multiple application interrupts are beneficial to flexible use and error prompt

13.3 Functional Description

13.3.1 DVP(Digital Video Port) signal and configuration

Digital Video Port (DVP), a parallel interface, provides clock, frame synchronization, line synchronization, and 8-bit data pins. The clock limit is 80 MHz, so it is generally used for sensors with resolution below 5 MP. The active levels of frame and line synchronization can be independently configured on the chip, and four modes are provided on the valid data:

1. Frame synchronization and line synchronization are valid at the same time (”&” logic)
2. Either frame synchronization or line synchronization is valid (”|” logic)
3. Frame synchronization is valid
4. Line synchronization is valid

13.3.2 YCbCr Format

The luminance signal is also called Y signal. The chrominance signal is composed of two independent signals. Depending on the color system and format, the two chrominance signals are often called U & V, Pb & Pr, or Cb & Cr. They use different coding formats, but their concepts are essentially the same.

As there are more retinal rod cells that recognize brightness than the retinal cone cells that recognize chrominance in the human retina, human eyes are more sensitive to brightness than to chrominance. Thus, a part of chrominance information can be discarded without being perceived by human eyes.

The chrominance signal with the highest resolution is denoted as 4:4:4. That is, every 4-point Y sample corresponds to 4-point Cb and 4-point Cr samples. Similarly, 4:2:2 means that every 4-point y sample corresponds to 2-point Cb and 2-point Cr samples. The number of scan lines of chrominance signal is as large as that of luminance signal, but

the chrominance sampling points on each scan line are only half of that of luminance signal. Different from the format described above, 4:2:0 does not mean that every 4-point Y sample corresponds to 2-point Cb and 0-point Cr samples, but means that every 4-point Y sample corresponds to 1-point Cb and 1-point Cr samples. The format 4:0:0 means discarding all the chrominance information, namely grayscale image.

13.3.3 Input Source

The data input source of CAM can be selected through configuration, and the relationship between configuration values and input sources is shown as follows:

Value	Input Type	Data Width	Value	Input Type	Data Width
1	Active DVP(TG)	8-bit	22	Adjust_1	16-bit
2	Defect Correct	8-bit	23	Adjust_2	16-bit
3	CCM_R	8-bit	24	Adjust_3	16-bit
4	CCM_G	8-bit	25	YUV420_0	16-bit
5	CCM_B	8-bit	26	YUV420_1	16-bit
6	Gamma_R	8-bit	27	YUV420_2	16-bit
7	Gamma_G	8-bit	28	YUV420_3	16-bit
8	-	-	29	Gamma_B	8-bit
9	BNR	8-bit	30	WDR_Y	8-bit
10	NR	16-bit	31	WDR_U	8-bit
11	EE	16-bit	32	WDR_V	8-bit
17	Scaler_0	16-bit	33	LSC	8-bit
18	Scaler_1	16-bit	34	AWB2	16-bit
19	Scaler_2	16-bit	35	YUV2RGB	24-bit
20	Scaler_3	16-bit	36	DVP_AS_2X	16-bit
21	Adjust_0	16-bit			

Fig. 13.2: Input Source Selection

13.3.4 Movie /Photo mode

In the photo mode, when the fixed-size memory provided by software is full, CAM will stop writing, and continue to write after the software performs POP operation for freeing up space.

In the movie mode, CAM will consecutively write in the fixed-size memory provided by software. That is, the memory will work as a ring buffer and no POP operation is required. In practice, it is necessary to ensure that images are taken out in real time or linked with the MJPEG module.

13.3.5 Rectangular Cropping of Image

You need to set the start and end positions for cropping line and frame synchronization signals by configuring the high 16 bits and low 16 bits of the registers HSYNC_CONTROL and VSYNC_CONTROL. Then, the image within the rectangular window marked based on the specified positions and size can be cropped and the remaining part beyond the window will be discarded. The start and end positions of the line synchronization signal are set by pixel numbers, while that of the frame synchronization signal are set by line numbers. The cropped image includes the start point but excludes the end point.

13.3.6 Integrity Test of Line and Frame Synchronization Signals

You can test the signal integrity by setting comparison values for line and frame synchronization signals respectively by configuring the low 16 bits and high 16 bits of the register FRAME_SIZE_CONTROL. The total number of pixels in each line is set for the line synchronization signal, and the total number of lines is set for the frame synchronization signal. When the count value of a line or frame synchronization signal of a frame of image is unequal to the comparison value, an interrupt will be generated.

13.3.7 Cached Image Information

The module has four FIFOs to record the address and size of images. Every time this module completely writes a frame into the memory, it will record the start address and image size of this frame of image in FIFO. However, when the memory is insufficient or four FIFOs are full, it will automatically discard the information of the incoming image. In the part where the image information has been taken out, the oldest image information will be removed by performing the pop operation through APB. Then, FIFO will automatically push data to ensure the timing of the image information in FIFO, as shown below:

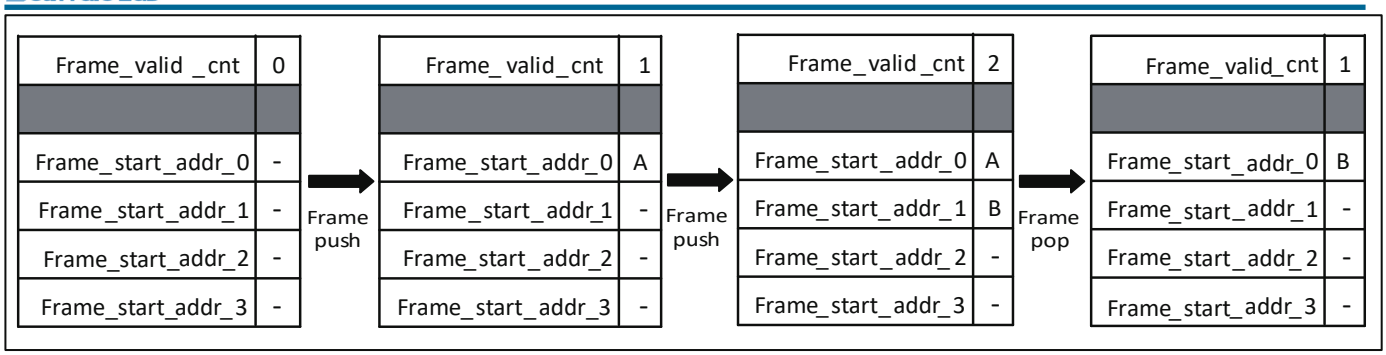


Fig. 13.3: FIFO Framework

13.3.8 Multiple Interrupts (Separately Configured)

A. Normal interrupt: You can set to generate an interrupt after a fixed number of images are written

B. Memory interrupt: When the memory is rewritten, an interrupt will be generated

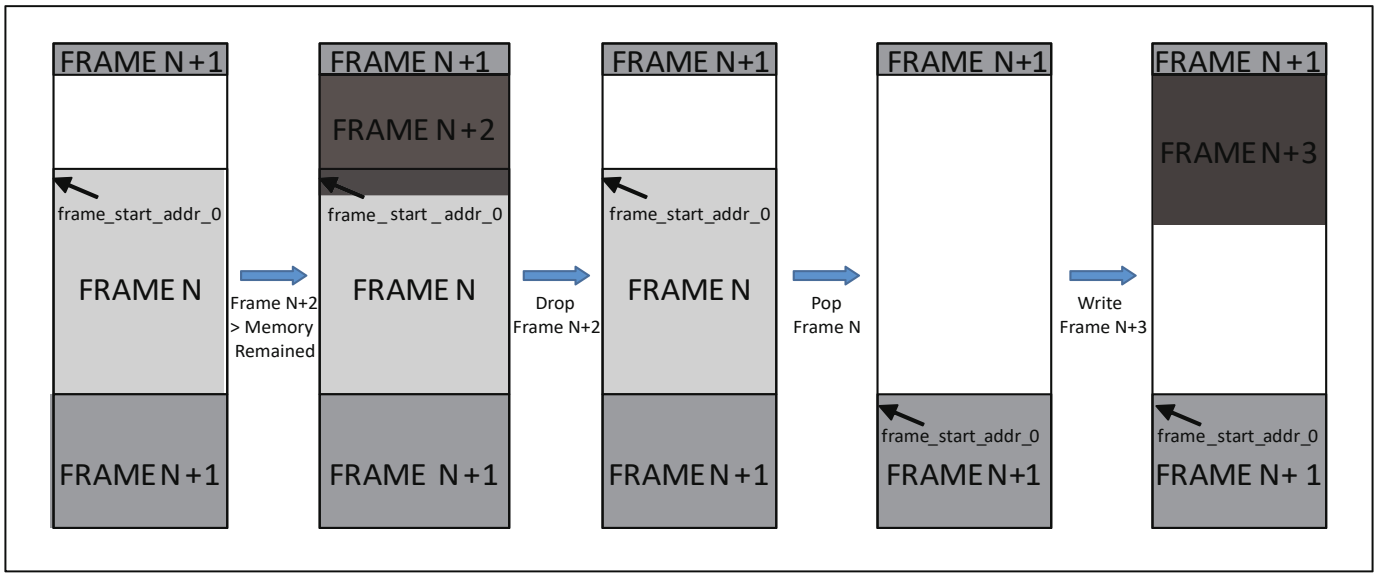


Fig. 13.4: Memory

C. Frame interrupt: When there are more than 4 unprocessed images, an interrupt will be generated

D. FIFO interrupt: When the bus is too late to be written into memory and FIFO overflows, an interrupt will be generated

E. Hsync interrupt: When the number of pixels in a line in a frame of image is unequal to the set value, an interrupt will be generated

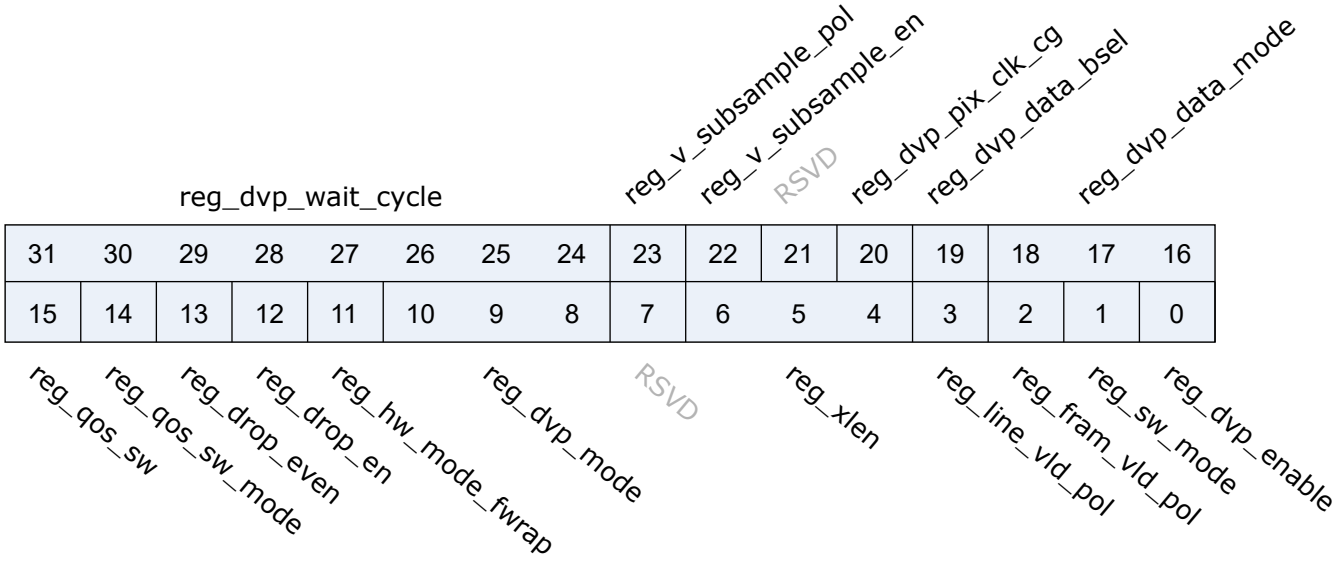
F. Vsync interrupt: When the total number of lines of a frame of image is unequal to the set value, an interrupt will be generated

13.4 Register description

Name	Description
dvp2axi_configue	
dvp2axi_addr_start	
dvp2axi_mem_bcnt	
dvp_status_and_error	
dvp2axi_frame_bcnt	
dvp_frame_fifo_pop	
dvp2axi_frame_vld	
dvp2axi_frame_period	
dvp2axi_misc	
dvp2axi_hsync_crop	
dvp2axi_vsync_crop	
dvp2axi_fram_exm	
frame_start_addr0	
frame_start_addr1	
frame_start_addr2	
frame_start_addr3	
frame_id_sts01	
frame_id_sts23	
dvp_debug	
dvp_dummy_reg	

13.4.1 dvp2axi_configue

Address：0x30012000



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

Bits	Name	Type	Reset	Description
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
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

13.4.2 dvp2axi_addr_start

Address：0x30012004

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

13.4.3 dvp2axi_mem_bcnt

Address：0x30012008

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"

13.4.4 dvp_status_and_error

Address：0x3001200c

RSVD RSVD st_dvp_idle axi_idle st_bus_flsh st_bus_wait st_bus_func st_bus_idle RSVD sts_vcnt_int sts_hcnt_int

frame_valid_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_cnt_trgr_int

sts_fifo_int sts_frame_int sts_mem_int sts_normal_int reg_int_fifo_en reg_int_frame_en reg_int_mem_en reg_int_normal_en reg_int_vcnt_en RSVD reg_int_hcnt_en

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

Bits	Name	Type	Reset	Description
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
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

13.4.5 dvp2axi_frame_bcnt

Address：0x30012010

reg_frame_byte_cnt

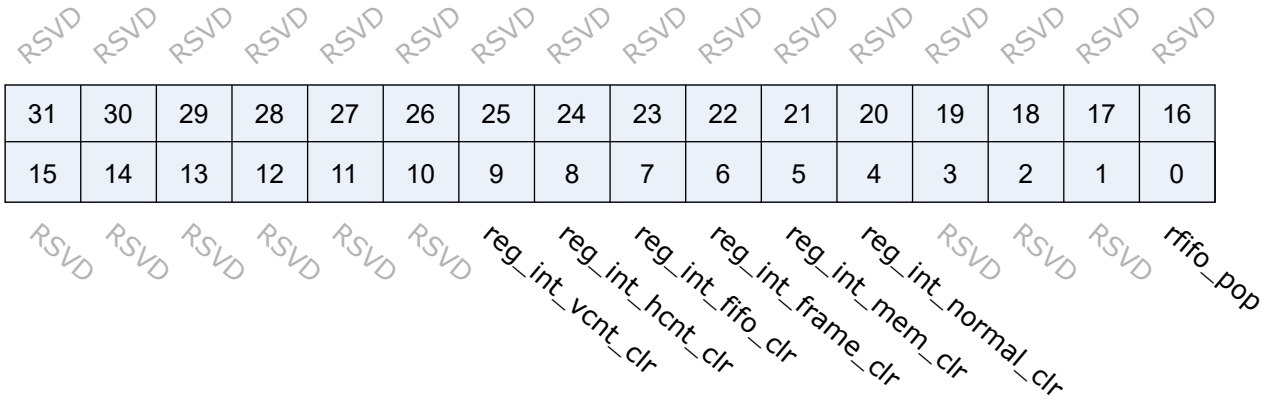
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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)

13.4.6 dvp_frame_fifo_pop

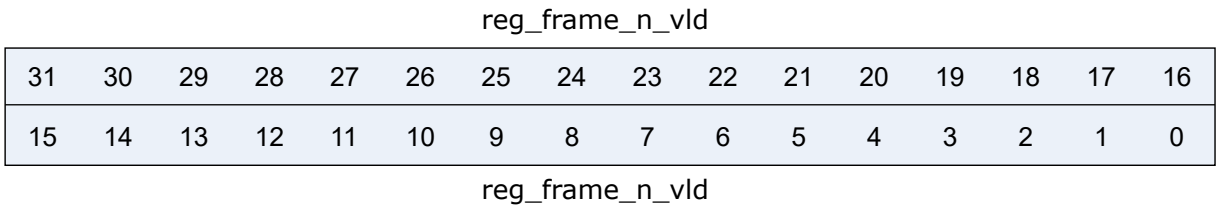
Address：0x30012014



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
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

13.4.7 dvp2axi_frame_vld

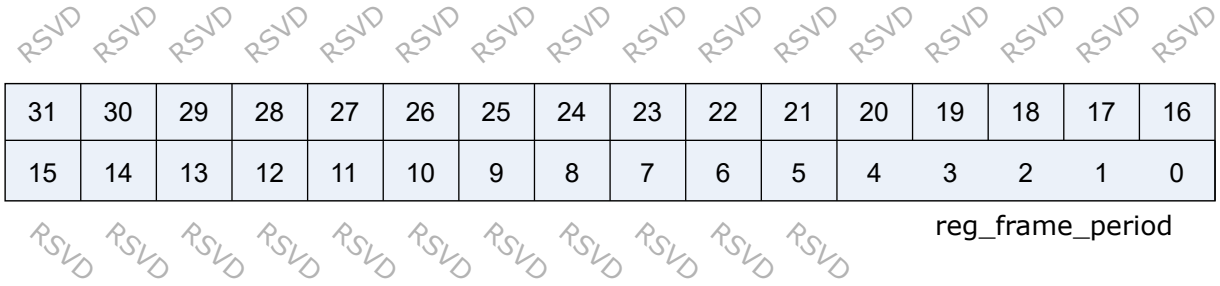
Address：0x30012018



Bits	Name	Type	Reset	Description
31:0	reg_frame_n_vld	r/w	32'hffff_ffff	Bitwise frame valid in period

13.4.8 dvp2axi_frame_period

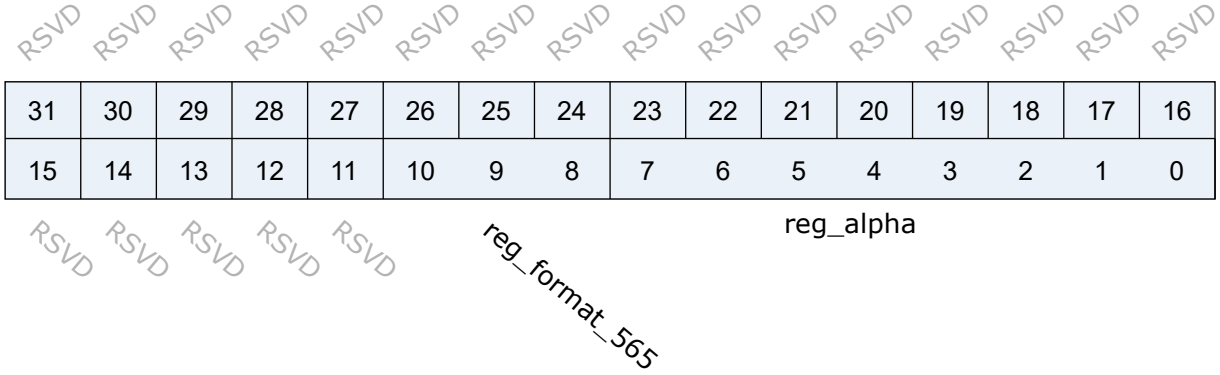
Address：0x3001201c



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)

13.4.9 dvp2axi_misc

Address：0x30012020



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)

Bits	Name	Type	Reset	Description
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]

13.4.10 dvp2axi_hsync_crop

Address：0x30012030

reg_hsync_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_hsync_act_end

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

13.4.11 dvp2axi_vsync_crop

Address：0x30012034

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

13.4.12 dvp2axi_fram_exm

Address：0x30012038

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

13.4.13 frame_start_addr0

Address：0x30012040

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

13.4.14 frame_start_addr1

Address：0x30012048

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

13.4.15 frame_start_addr2

Address：0x30012050

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

13.4.16 frame_start_addr3

Address：0x30012058

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

13.4.17 frame_id_sts01

Address：0x30012060

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

13.4.18 frame_id_sts23

Address：0x30012064

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

13.4.19 dvp_debug

Address：0x300120f0

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 reg_id_latch_line RSVD RSVD RSVD RSVD reg_dvp_dbg_sel reg_dvp_dbg_en

Bits	Name	Type	Reset	Description
31:12	RSVD			
11:8	reg_id_latch_line	r/w	4'd5	ID latch timing (line count)
7:4	RSVD			
3:1	reg_dvp_dbg_sel	r/w	3'd0	DVP2BUS debgu flag selection
0	reg_dvp_dbg_en	r/w	1'b0	DVP2BUS debgu flag enable

13.4.20 dvp_dummy_reg

Address：0x300120fc

RESERVED

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

RESERVED

Bits	Name	Type	Reset	Description
31:0	RESERVED	rsvd	32'hf0f0f0f0	RESERVED

14 IR

14.1 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.

14.2 Features

- Receives data through the fixed NEC and RC5 protocols
- Receives data in any format by pulse width counting
- With powerful infrared waveform editing capability: It can emit waveforms conforming to various protocols
- Up to 15-gear power settings to adapt to different power consumption requirements
- Receives up to 64-bit data
- Sends up to 128-bit data in the non-free mode and continuously sends the data of any length in the free mode
- 4*4 byte TX FIFO, 64*2 byte RX FIFO
- Adjustable width of the TX FIFO in the free mode (8/16/24/32-bit)
- Programmable carrier frequency and duty ratio
- Maximum operating frequency of 32 MHz
- Supports the DMA sending mode
- End of sending and receiving interrupts

14.3 Functional Description

14.3.1 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:

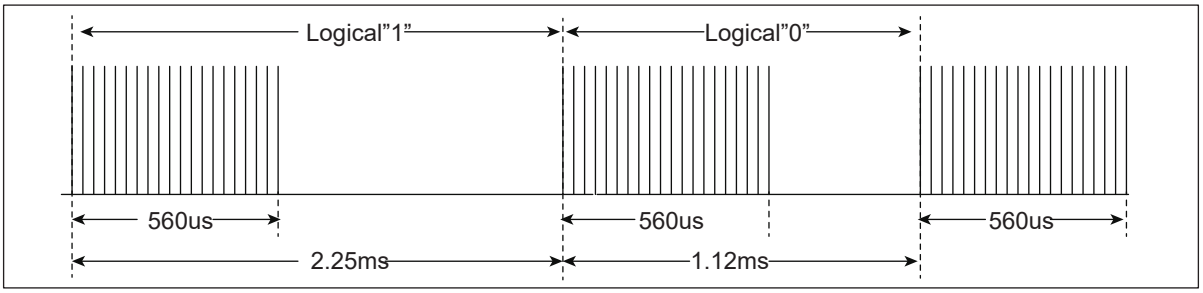


Fig. 14.1: NEC Logic Waveform

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:

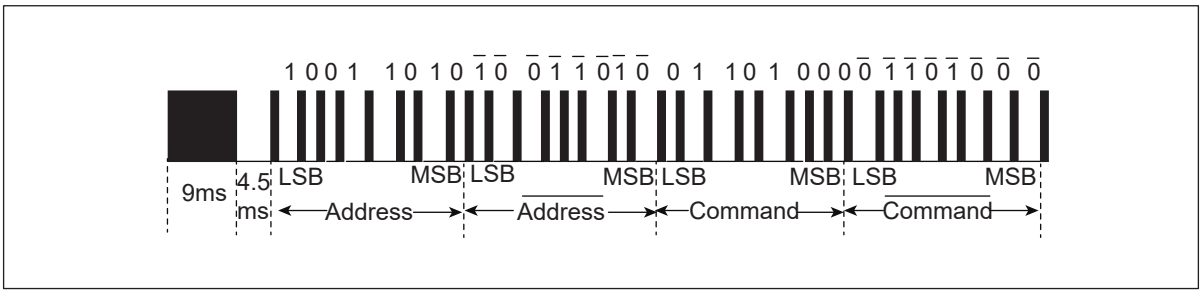


Fig. 14.2: NEC Protocol Waveform

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:

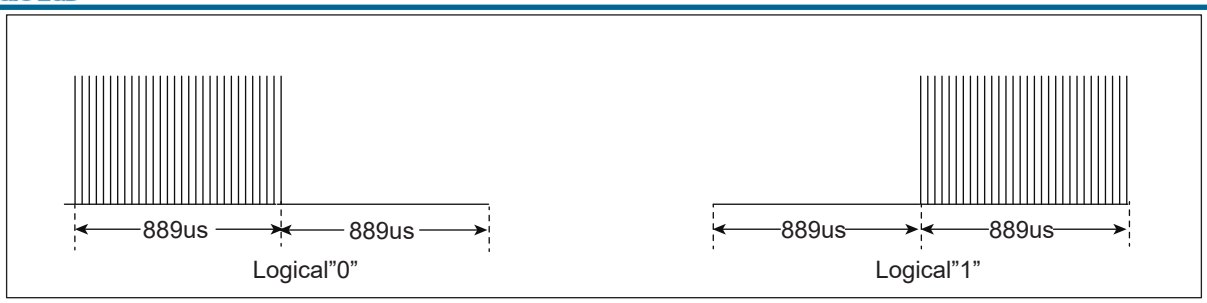


Fig. 14.3: RC5 Logic Waveform

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:

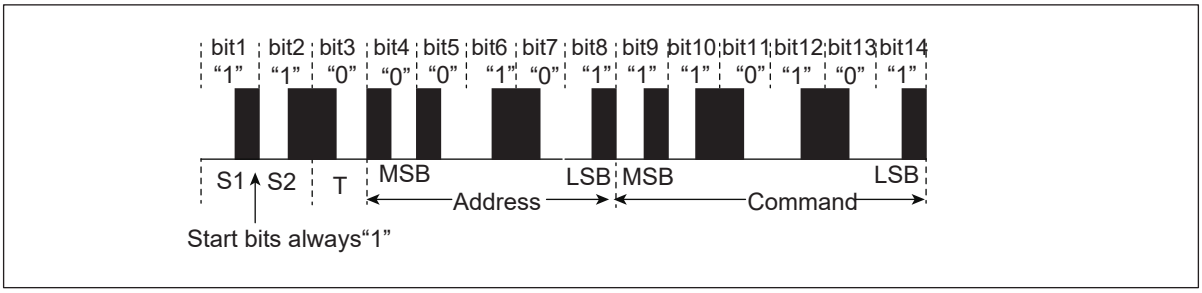


Fig. 14.4: RC5 Protocol Waveform

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.

14.3.2 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.

14.3.3 Normal sending mode

You can configure the head pulse, tail pulse, logical ‘0’ pulse, and logical ‘1’ pulse depending on protocols. During setting, you need to calculate the common pulse width unit—the greatest common divisor—of various pulses with different widths in the used protocol, fill it in the low 12 bits in the register IRTX_PULSE_WIDTH, and then fill the corresponding multiple of each pulse in the register IRTX_PW.

Then, you can fill the logic value to be sent into the TX FIFO with a depth of 4 and a width of 4 bytes.

14.3.4 Pulse width sending

IR provides a way of sending pulse width for the protocols that are unsuitable for the normal sending mode. You need to calculate the common pulse width unit—the greatest common divisor—of various pulses with different widths in the used protocol, and fill it in the low 12 bits in the register IRTX_PULSE_WIDTH. Then, for the multiples (8-bit per multiple) corresponding to each level width from the first high level to the last level, every four multiples are combined into a 32-bit number for padding in the TX FIFO.

14.3.5 Carrier modulation

Setting the high 16 bits of the register IRTX_PULSE_WIDTH can generate carriers with different frequencies and duty ratios. The <TXMPH1W> bit in the register is set to the width of carrier phase 1, while the <TXMPH0W> bit is set to the width of carrier phase 0.

14.3.6 Free mode

In the free mode, there is no limit to the total length of data to be sent. When there is data in TX FIFO, it will be sent, so it can be used with DMA. The width of TX FIFO in free mode can be configured to 8/16/24/32-bit.

14.3.7 DMA mode

IR TX supports the DMA mode, which requires to enable the free mode and set the threshold of TX FIFO by configuring <TFITH> in the register IRTX_FIFO_CONFIG_1.

When this mode is enabled, if <TFICNT> in the IRTX_FIFO_CONFIG_1 is greater than <TFITH>, the DMA TX request will be triggered. After DMA is configured, when receiving this request, DMA will transfer data from memory to TX FIFO as configured.

14.3.8 IR Interrupt

There are separate end of sending and receiving interrupts for IR, and when a sending action ends, a TX interrupt will be generated. When a piece of data is received, it will wait for the level duration to reach the preset end threshold before generating a receiving interrupt.

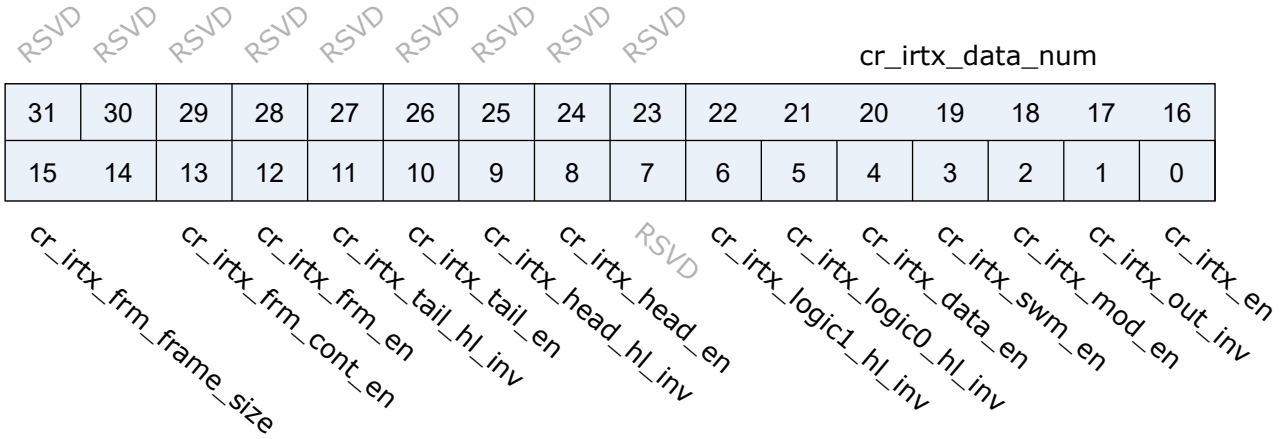
The register IRTX_INT_STS can query the TX interrupt status and clear the interrupt, and the register IRRX_INT_STS can query the receiving interrupt status and clear the interrupt.

14.4 Register description

Name	Description
irtx_config	
irtx_int_sts	
irtx_pulse_width	
irtx_pw_0	
irtx_pw_1	
irrx_config	
irrx_int_sts	
irrx_pw_config	
irrx_data_count	
irrx_data_word0	
irrx_data_word1	
irtx_fifo_config_0	
irtx_fifo_config_1	
ir_fifo_wdata	
ir_fifo_rdata	

14.4.1 irtx_config

Address：0x2000a600

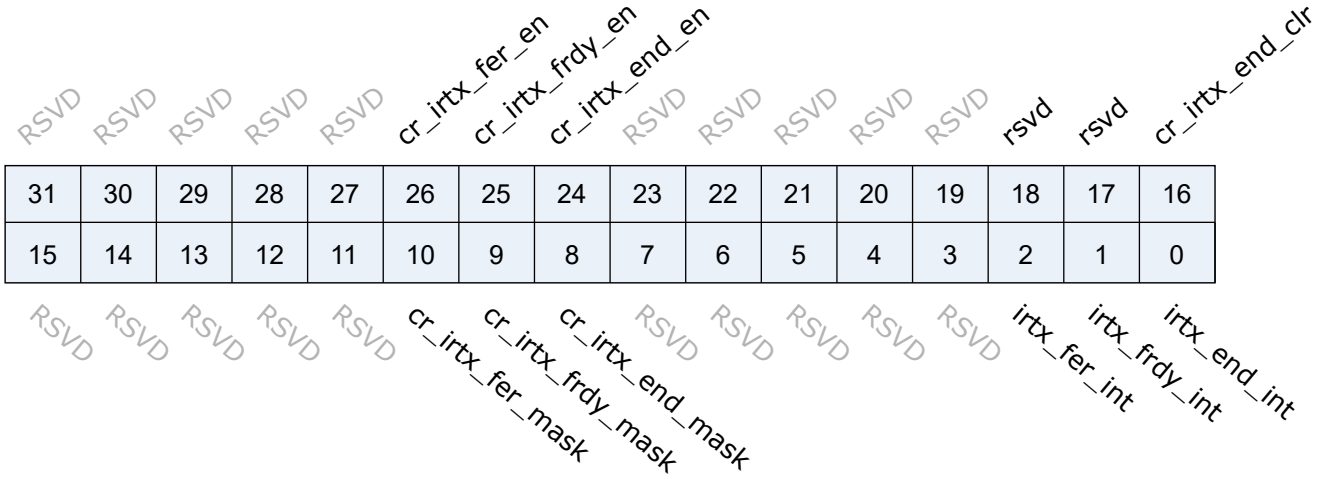


Bits	Name	Type	Reset	Description
31:23	RSVD			
22:16	cr_irtx_data_num	r/w	7'd31	Bit count of Data phase (unit: bit / PW for normal / SWM) (Don't-care if cr_irtx_frm_en is enabled)
15:14	cr_irtx_frm_frame_size	r/w	2'd0	IRTX freerun mode frame size (also the valid width for each FIFO entry) 2'd0: 8-bit 2'd1: 16-bit 2'd2: 24-bit 2'd3: 32-bit
13	cr_irtx_frm_cont_en	r/w	1'b0	Enable signal of IRTX freerun continuous mode 0: Disabled, each data frame is separated by an idle time (tail_ph0_w+1 + tail_ph1_w+1)*pw_unit 1: Enabled, data continuously sent without interval (if FIFO data is valid)
12	cr_irtx_frm_en	r/w	1'b0	Enable signal of IRTX freerun mode (Don't care if SWM is enabled) Note: HEAD/TAIL is disabled in freerun mode
11	cr_irtx_tail_hl_inv	r/w	1'b0	Tail pulse H/L inverse signal (Don't care if SWM or FRM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
10	cr_irtx_tail_en	r/w	1'b1	Enable signal of tail pulse (Don't care if SWM or FRM is enabled)
9	cr_irtx_head_hl_inv	r/w	1'b0	Tail pulse H/L inverse signal (Don't care if SWM or FRM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
8	cr_irtx_head_en	r/w	1'b1	Enable signal of head pulse (Don't care if SWM or FRM is enabled)
7	RSVD			
6	cr_irtx_logic1_hl_inv	r/w	1'b0	Logic 1 H/L inverse signal (Don't care if SWM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
5	cr_irtx_logic0_hl_inv	r/w	1'b0	Logic 0 H/L inverse signal (Don't care if SWM is enabled) 0: Phase 0 is High (Active), phase 1 is Low (Idle) (H -> L) 1: Phase 0 is Low (Idle), phase 1 is High (Active) (L -> H)
4	cr_irtx_data_en	r/w	1'b1	Enable signal of data phase (Don't care if SWM or FRM is enabled)

Bits	Name	Type	Reset	Description
3	cr_irtx_swm_en	r/w	1'b0	Enable signal of IRTX Software Mode (SWM)
2	cr_irtx_mod_en	r/w	1'b0	Enable signal of output modulation
1	cr_irtx_out_inv	r/w	1'b0	Output inverse signal 1'b0: Output stays at Low during idle state 1'b1: Output stays at High during idle state
0	cr_irtx_en	r/w	1'b0	Enable signal of IRTX function Asserting this bit will trigger the transaction, and should be de-asserted after finish

14.4.2 irtx_int_sts

Address：0x2000a604



Bits	Name	Type	Reset	Description
31:27	RSVD			
26	cr_irtx_fer_en	r/w	1'b1	Interrupt enable of irtx_fer_int
25	cr_irtx_frdy_en	r/w	1'b1	Interrupt enable of irtx_frdy_int
24	cr_irtx_end_en	r/w	1'b1	Interrupt enable of irtx_end_int
23:19	RSVD			
18	rsvd	rsvd	1'b0	
17	rsvd	rsvd	1'b0	
16	cr_irtx_end_clr	w1c	1'b0	Interrupt clear of irtx_end_int
15:11	RSVD			
10	cr_irtx_fer_mask	r/w	1'b1	Interrupt mask of irtx_fer_int

Bits	Name	Type	Reset	Description
9	cr_irtx_frdy_mask	r/w	1'b1	Interrupt mask of irtx_frdy_int
8	cr_irtx_end_mask	r/w	1'b1	Interrupt mask of irtx_end_int
7:3	RSVD			
2	irtx_fer_int	r	1'b0	IRTX FIFO error interrupt, auto-cleared when FIFO overflow/underflow error flag is cleared
1	irtx_frdy_int	r	1'b1	IRTX FIFO ready (tx_fifo_cnt > tx_fifo_th) interrupt, auto-cleared when data is pushed
0	irtx_end_int	r	1'b0	IRTX transfer end interrupt

14.4.3 irtx_pulse_width

Address：0x2000a610

cr_irtx_mod_ph1_w cr_irtx_mod_ph0_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	RSVD	cr_irtx_pw_unit											

Bits	Name	Type	Reset	Description
31:24	cr_irtx_mod_ph1_w	r/w	8'd34	Modulation phase 1 width
23:16	cr_irtx_mod_ph0_w	r/w	8'd17	Modulation phase 0 width
15:12	RSVD			
11:0	cr_irtx_pw_unit	r/w	12'd1124	Pulse width unit

14.4.4 irtx_pw_0

Address：0x2000a614

cr_irtx_logic1_ph1_w cr_irtx_logic1_ph0_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

cr_irtx_logic0_ph1_w cr_irtx_logic0_ph0_w

Bits	Name	Type	Reset	Description
31:24	cr_irtx_logic1_ph1_w	r/w	8'd2	Pulse width of logic1 phase 1 (Don't care if SWM is enabled)
23:16	cr_irtx_logic1_ph0_w	r/w	8'd0	Pulse width of logic1 phase 0 (Don't care if SWM is enabled)
15:8	cr_irtx_logic0_ph1_w	r/w	8'd0	Pulse width of logic0 phase 1 (Don't care if SWM is enabled)
7:0	cr_irtx_logic0_ph0_w	r/w	8'd0	Pulse width of logic0 phase 0 (Don't care if SWM is enabled)

14.4.5 irtx_pw_1

Address：0x2000a618

cr_irtx_tail_ph1_w								cr_irtx_tail_ph0_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
cr_irtx_head_ph1_w								cr_irtx_head_ph0_w							

Bits	Name	Type	Reset	Description
31:24	cr_irtx_tail_ph1_w	r/w	8'd0	Pulse width of tail pulse phase 1 (Don't care if SWM is enabled)
23:16	cr_irtx_tail_ph0_w	r/w	8'd0	Pulse width of tail pulse phase 0 (Don't care if SWM is enabled)
15:8	cr_irtx_head_ph1_w	r/w	8'd7	Pulse width of head pulse phase 1 (Don't care if SWM is enabled)
7:0	cr_irtx_head_ph0_w	r/w	8'd15	Pulse width of head pulse phase 0 (Don't care if SWM is enabled)

14.4.6 irrx_config

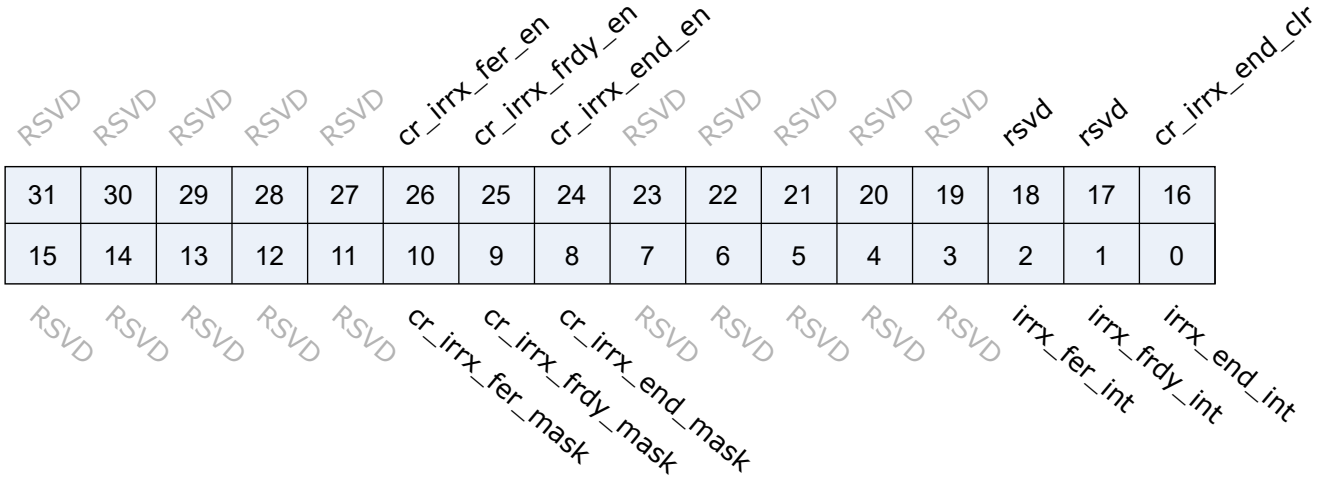
Address：0x2000a640

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	cr_irrx_deg_cnt				RSVD	RSVD	RSVD	cr_irrx_deg_en	cr_irrx_mode		cr_irrx_in_inv	cr_irrx_en

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

14.4.7 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			

Bits	Name	Type	Reset	Description
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

14.4.8 irrx_pw_config

Address：0x2000a648

cr_irrx_end_th

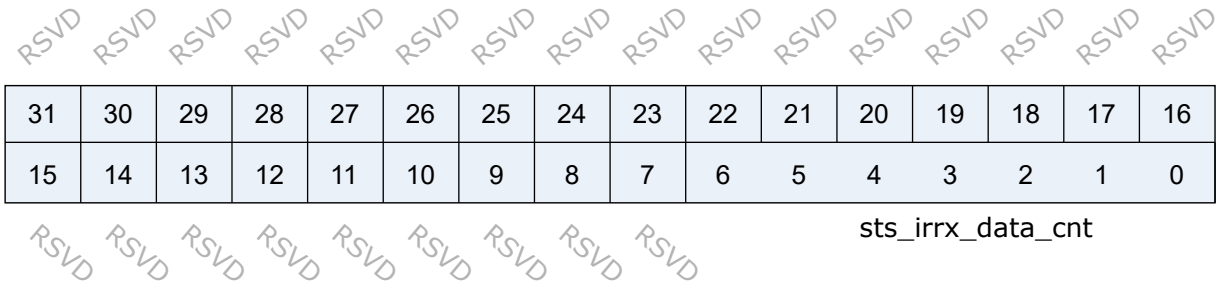
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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)

14.4.9 irrx_data_count

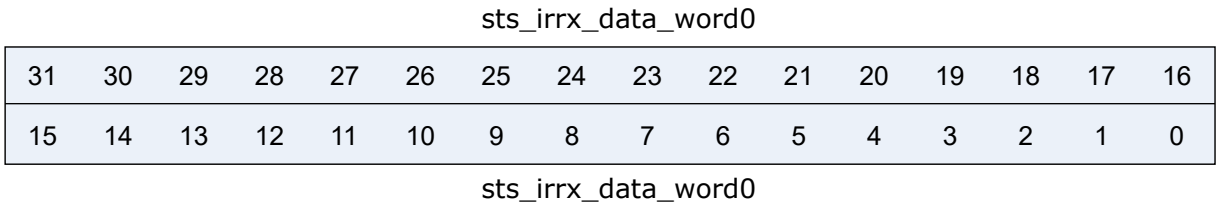
Address：0x2000a650



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)

14.4.10 irrx_data_word0

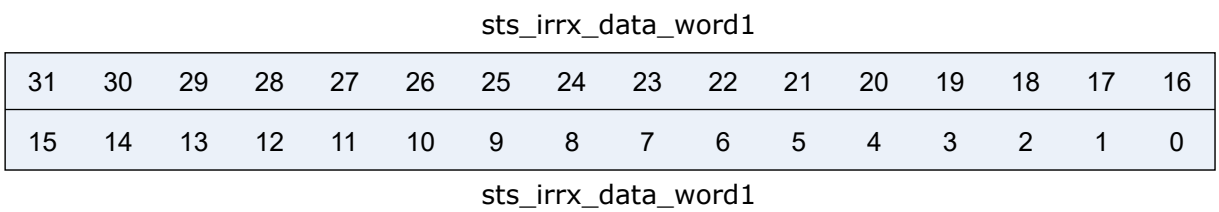
Address：0x2000a654



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

14.4.11 irrx_data_word1

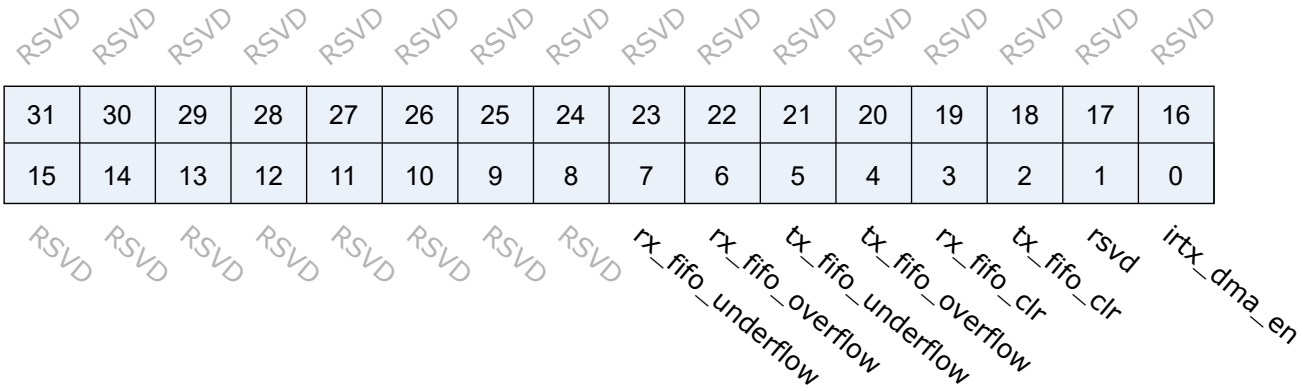
Address：0x2000a658



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

14.4.12 irtx_fifo_config_0

Address：0x2000a680



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
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	rsvd	rsvd	1'b0	
0	irtx_dma_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

14.4.13 irtx_fifo_config_1

Address：0x2000a684

RSVD RSVD rx_fifo_th RSVD RSVD RSVD RSVD RSVD RSVD tx_fifo_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

RSVD rx_fifo_cnt RSVD RSVD RSVD RSVD RSVD tx_fifo_cnt

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
23:18	RSVD			
17:16	tx_fifo_th	r/w	2'd0	TX FIFO threshold, irtx_frdy_int & dma_tx_req will not be asserted if tx_fifo_cnt is less than this value
15	RSVD			
14:8	rx_fifo_cnt	r	7'd0	RX FIFO available count
7:3	RSVD			
2:0	tx_fifo_cnt	r	3'd4	TX FIFO available count

14.4.14 ir_fifo_wdata

Address：0x2000a688

tx_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

tx_fifo_wdata

Bits	Name	Type	Reset	Description
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)

14.4.15 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

15 SPI

15.1 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).

15.2 Features

- Can be used as SPI master or slave
- Both master and slave support 4 clock formats (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
- Configurable MSB/LSB transfer
- 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

15.3 Functional Description

15.3.1 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, the idle level is low, and if CPOL=1, the idle level is high. CPHA determines the sampling time. If CPHA=0, sampling is made at the first lock edge of each cycle, and if CPHA=1, 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:

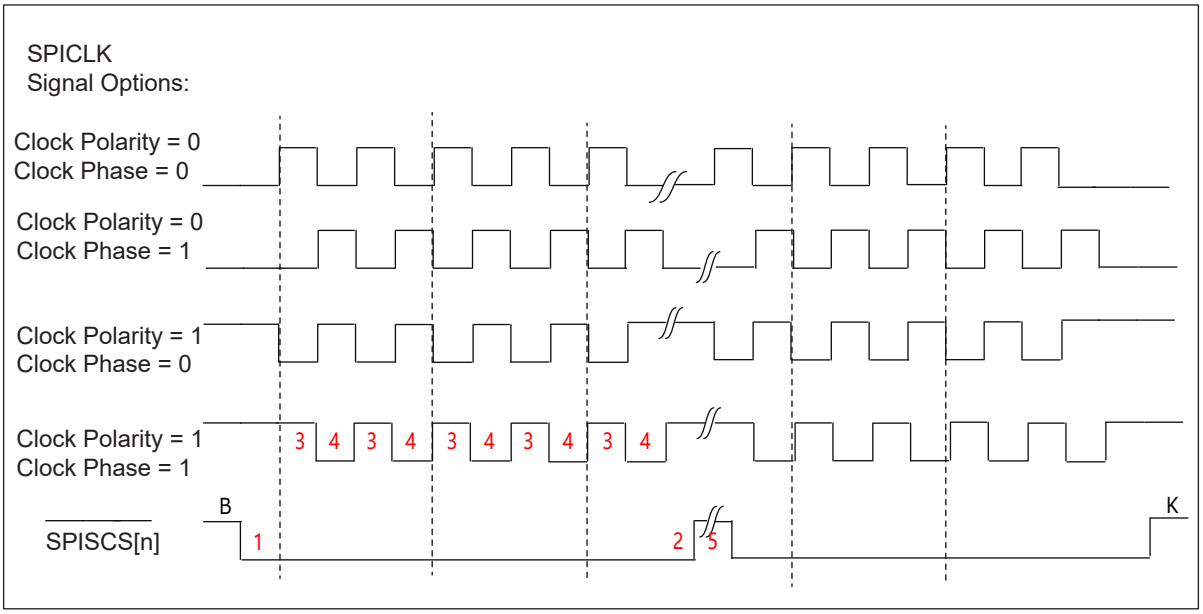


Fig. 15.1: SPI Timing

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.

15.3.2 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.

15.3.3 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.

15.3.4 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:

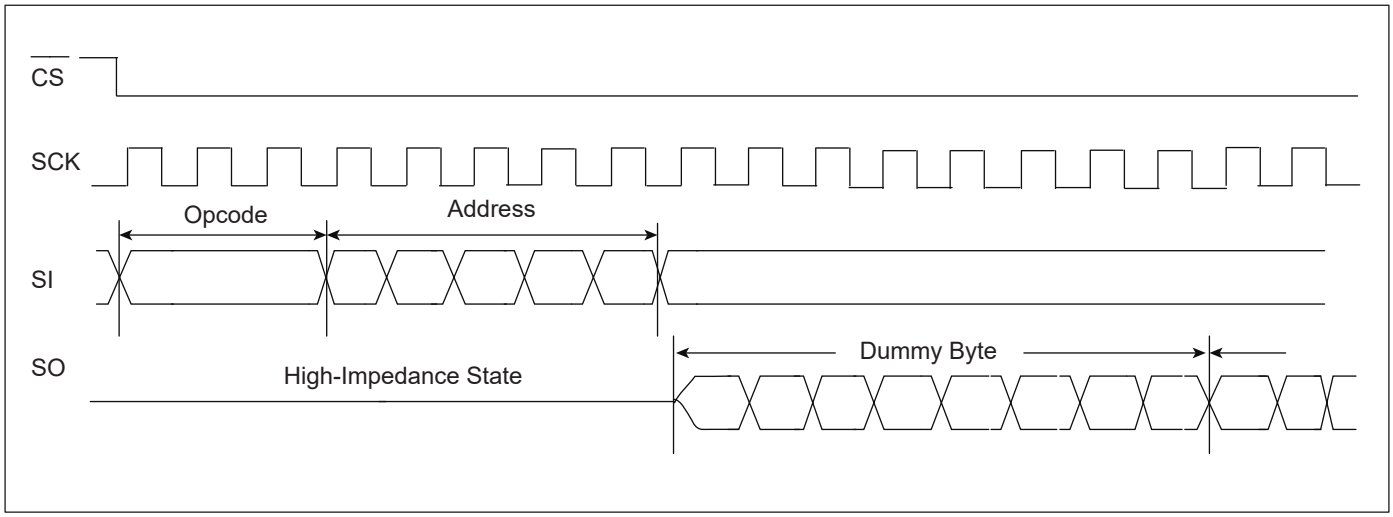


Fig. 15.2: SPI Ignore Waveform

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.

15.3.5 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.

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. The data width shall be less than cr_spi_deg_cnt+1: As shown in the figure below, when the data width is 4, setting the cr_spi_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.
- “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.

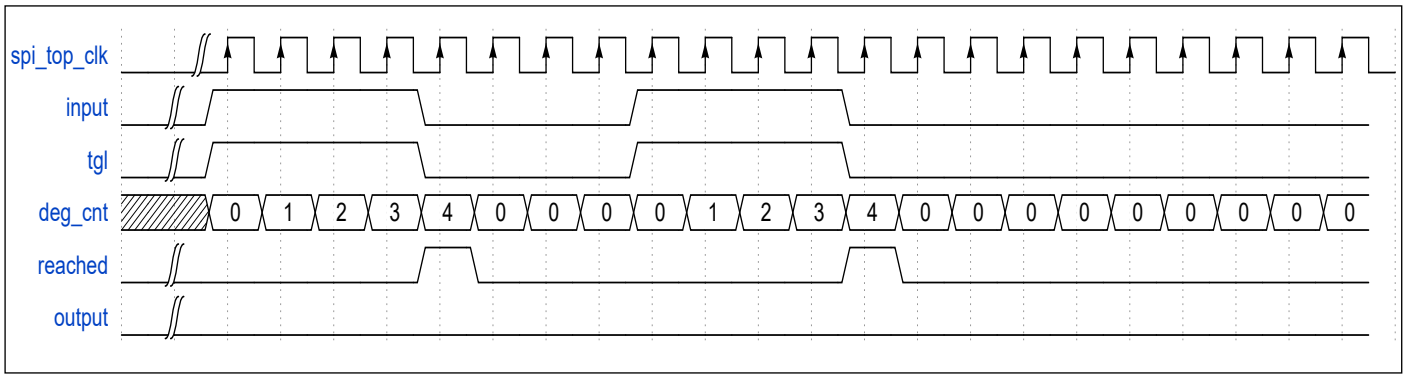


Fig. 15.3: SPI Filter Waveform

15.3.6 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).

15.3.7 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.

15.3.8 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.

15.3.9 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.

15.3.10 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.

15.3.11 SPI Interrupt

SPI supports the following interrupt control modes:

- SPI end of transfer interrupt
- TX FIFO request interrupt
- RX FIFO request interrupt
- Slave mode transfer timeout interrupt
- Slave mode TX overload interrupt
- TX/RX FIFO overflow 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. The TX/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. 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. 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.

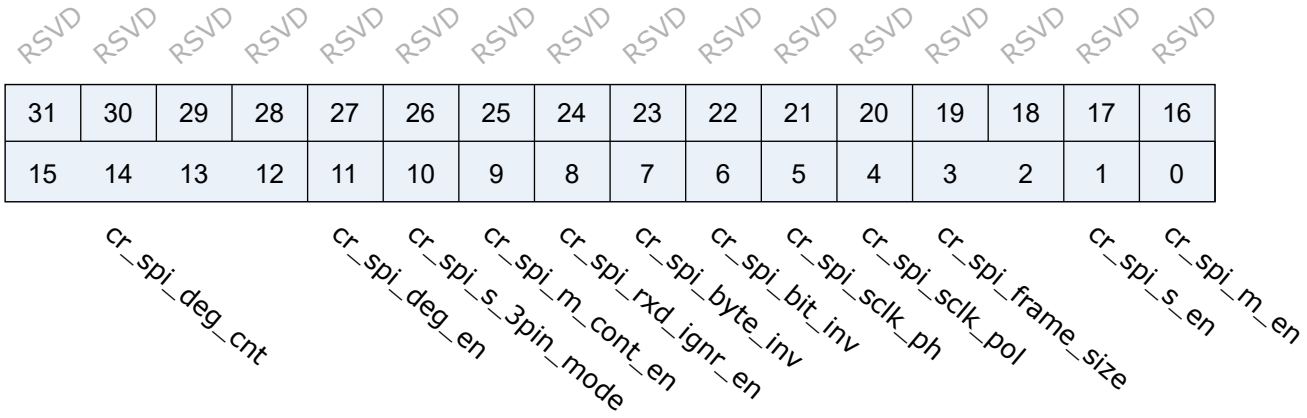
15.4 Register description

Name	Description
spi_config	
spi_int_sts	
spi_bus_busy	
spi_prd_0	
spi_prd_1	
spi_rxd_ignr	
spi_sto_value	
spi_fifo_config_0	
spi_fifo_config_1	
spi_fifo_wdata	
spi_fifo_rdata	

Name	Description
backup_io_en	

15.4.1 spi_config

Address：0x2000a200

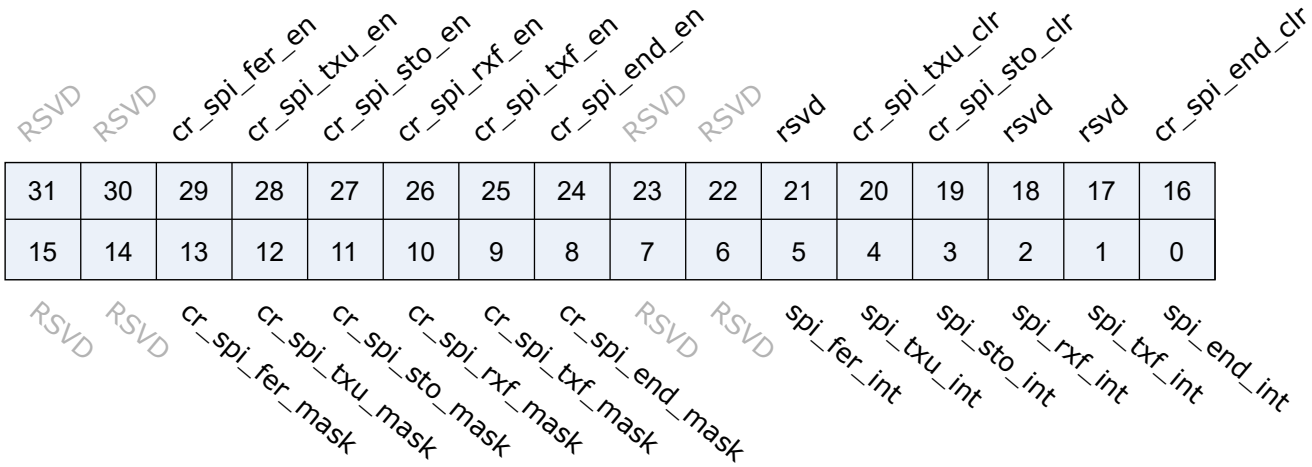


Bits	Name	Type	Reset	Description
31:16	RSVD			
15:12	cr_spi_deg_cnt	r/w	4'd0	De-glitch function cycle count
11	cr_spi_deg_en	r/w	1'b0	Enable signal of all input de-glitch function
10	cr_spi_s_3pin_mode	r/w	1'b0	SPI slave 3-pin mode 1'b0: 4-pin mode (SS_n is enabled) 1'b1: 3-pin mode (SS_n is disabled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

Bits	Name	Type	Reset	Description
5	cr_spi_sclk_ph	r/w	1'b0	SCLK clock phase inverse signal
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 2'd1: 16-bit 2'd2: 24-bit 2'd3: 32-bit
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

15.4.2 spi_int_sts

Address：0x2000a204



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

Bits	Name	Type	Reset	Description
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
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

15.4.3 spi_bus_busy

Address：0x2000a208

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 sts_spi_bus_busy

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

15.4.4 spi_prd_0

Address：0x2000a210

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 (please refer to "Timing" tab)
23:16	cr_spi_prd_d_ph_0	r/w	8'd15	Length of DATA phase 0 (please refer to "Timing" tab)
15:8	cr_spi_prd_p	r/w	8'd15	Length of STOP condition (please refer to "Timing" tab)
7:0	cr_spi_prd_s	r/w	8'd15	Length of START condition (please refer to "Timing" tab)

15.4.5 spi_prd_1

Address：0x2000a214

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 (please refer to "Timing" tab)

15.4.6 spi_rxd_ignr

Address：0x2000a218

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD cr_spi_rxd_ignr_s

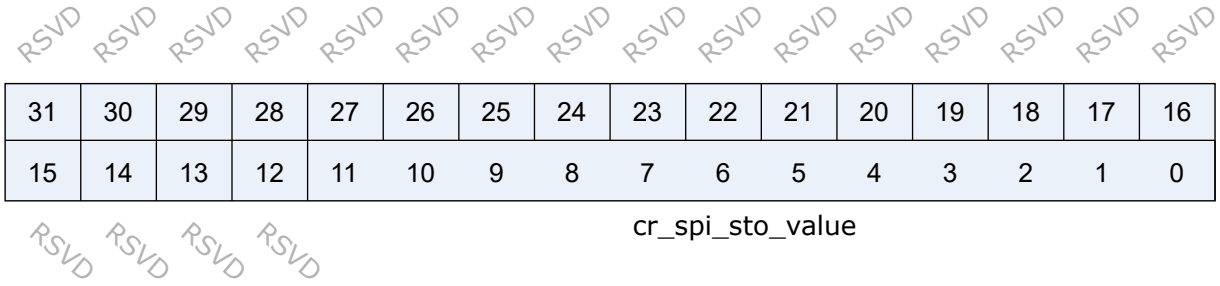
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD cr_spi_rxd_ignr_p

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
15:5	RSVD			
4:0	cr_spi_rxd_ignr_p	r/w	5'd0	Stopping point of RX data ignore function

15.4.7 spi_sto_value

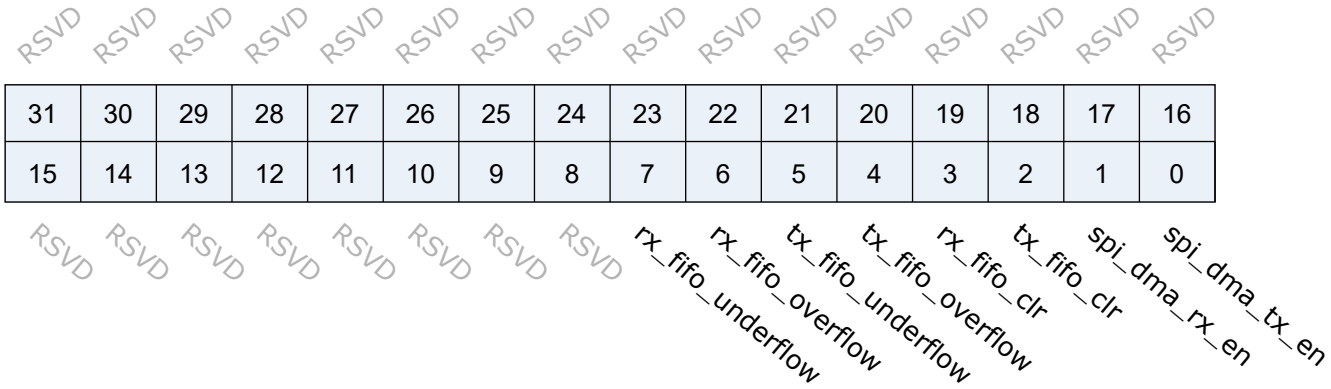
Address：0x2000a21c



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

15.4.8 spi_fifo_config_0

Address：0x2000a280



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
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
1	spi_dma_rx_en	r/w	1'b0	Enable signal of dma_rx_req/ack interface

Bits	Name	Type	Reset	Description
0	spi_dma_tx_en	r/w	1'b0	Enable signal of dma_tx_req/ack interface

15.4.9 spi_fifo_config_1

Address：0x2000a284

RSVD RSVD RSVD rx_fifo_th RSVD RSVD RSVD tx_fifo_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

RSVD RSVD rx_fifo_cnt RSVD RSVD tx_fifo_cnt

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 (unit: byte)
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count (unit: byte)

15.4.10 spi_fifo_wdata

Address：0x2000a288

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

15.4.11 spi_fifo_rdata

Address：0x2000a28c

spi_fifo_rdata

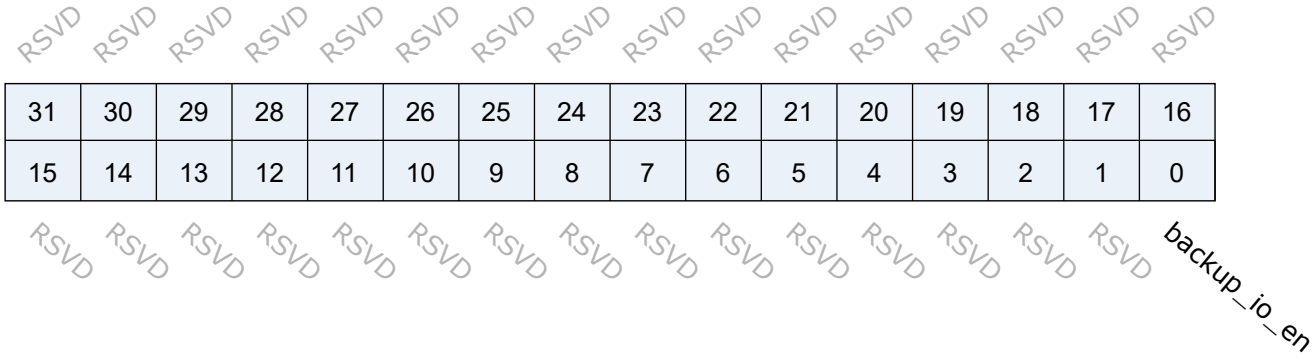
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

15.4.12 backup_io_en

Address：0x2000a2fc



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

16 UART

16.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) provides a flexible way to exchange full-duplex data with external devices. BL808 is provided with four UARTs, which can be used together with DMA to achieve efficient data communication.

16.2 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 above
- Supports the LIN bus protocol

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

16.3 Functional Description

16.3.1 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:

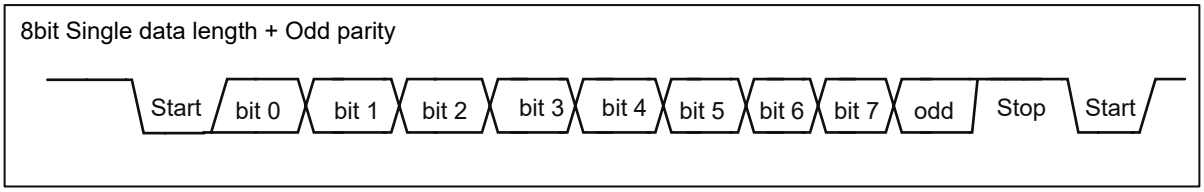


Fig. 16.1: UART Data Format

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.

16.3.2 Basic Architecture

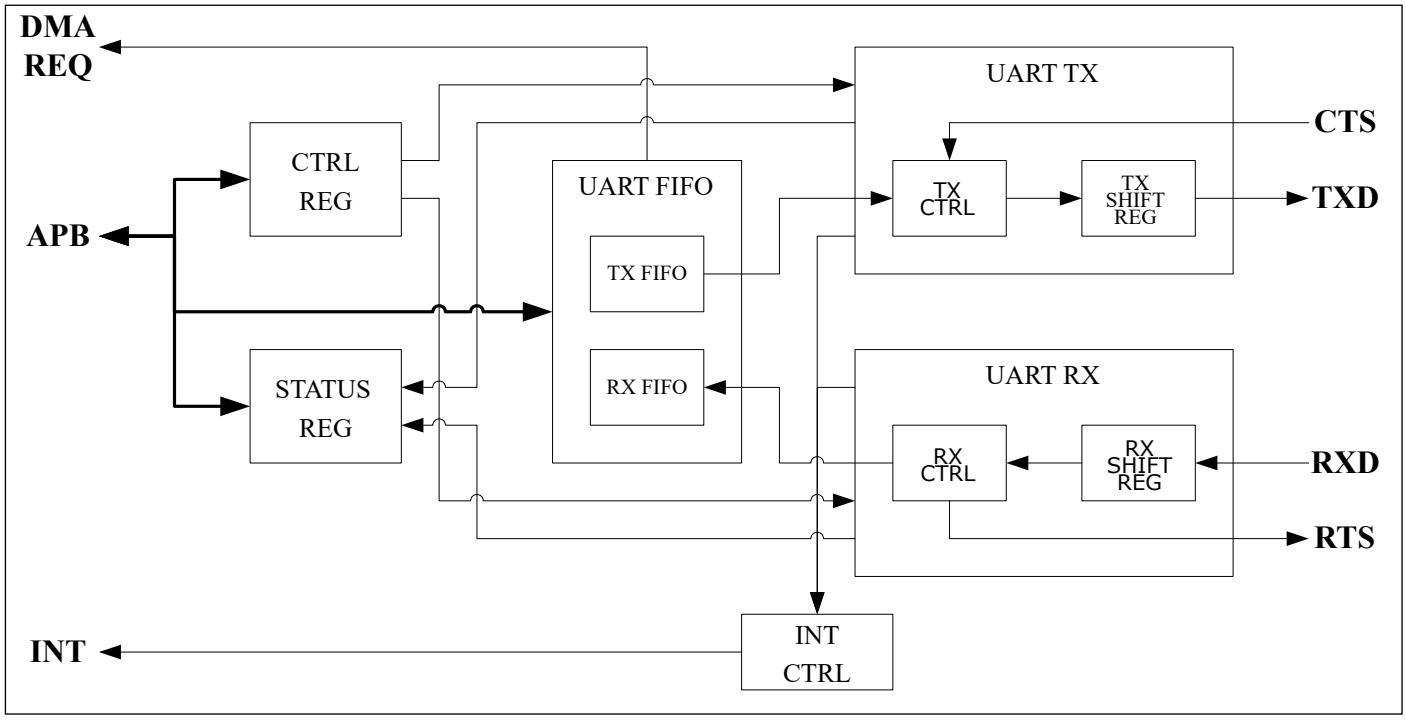


Fig. 16.2: UART Architecture

16.3.3 Clock Source

UART’s clock sources include XCLK, 160 Mhz CLK, and BCLK. The frequency divider in the clock is used to divide the clock source and then generate the clock signal to drive UART, as shown below:

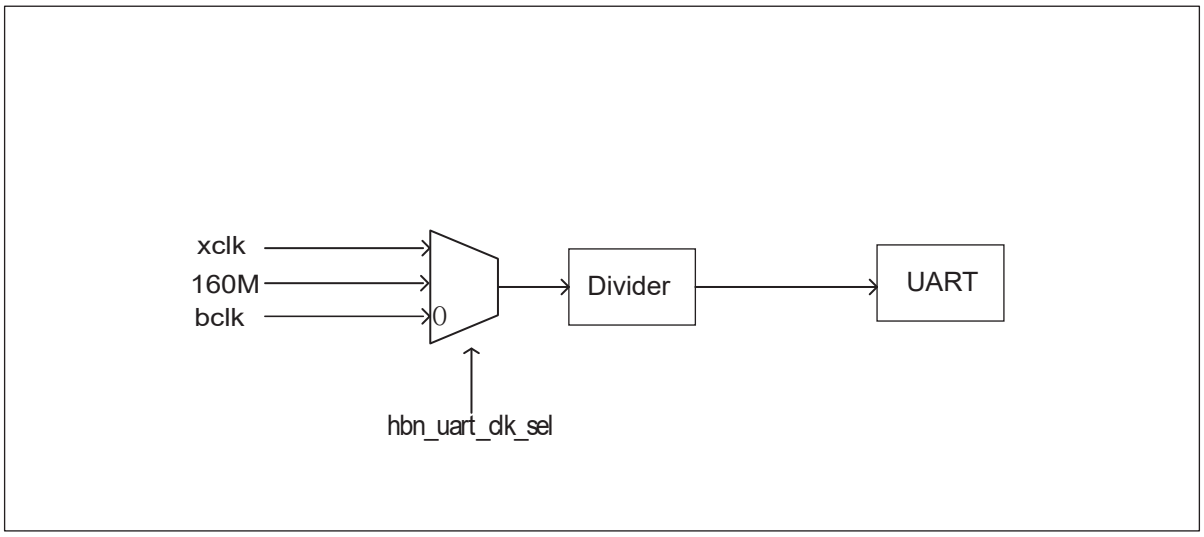


Fig. 16.3: UART clock

16.3.4 Baud Rate Setting

The user can set the register UART_BIT_PRD to generate the required baud rate. The high 16 bits cr_urx_bit_prd and low 16 bits cr_utx_bit_prd of this register correspond to RX and TX respectively. That is, the baud rates of RX and TX can be set separately. The 16 bits value is calculated by the following formula: baud rate = UART clock/(16-bit width factor + 1), that is, 16-bit width factor = UART clock/baud rate-1. This factor means the count value obtained by counting the bit width of the current baud rate with UART clock. Since the maximum 16-bit width factor is 65535, the minimum baud rate supported by UART is UART clock/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:

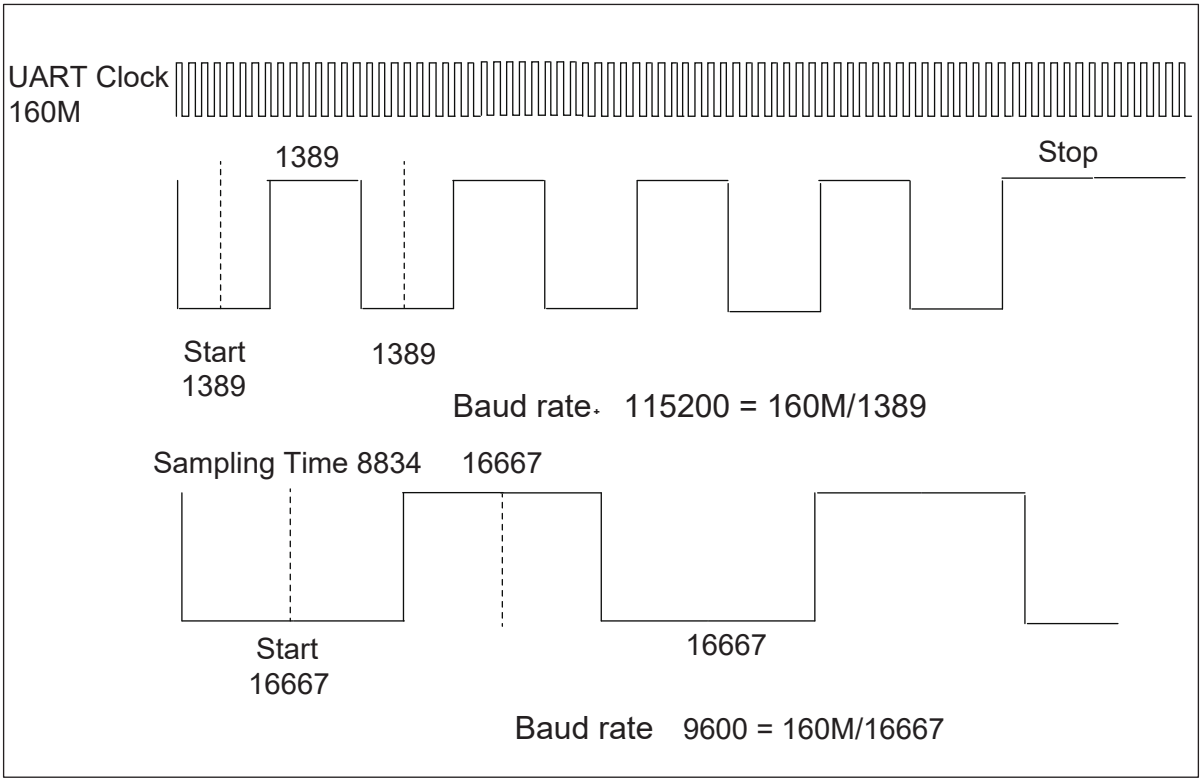


Fig. 16.4: UART Sampling Waveform

16.3.5 Filtering

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.

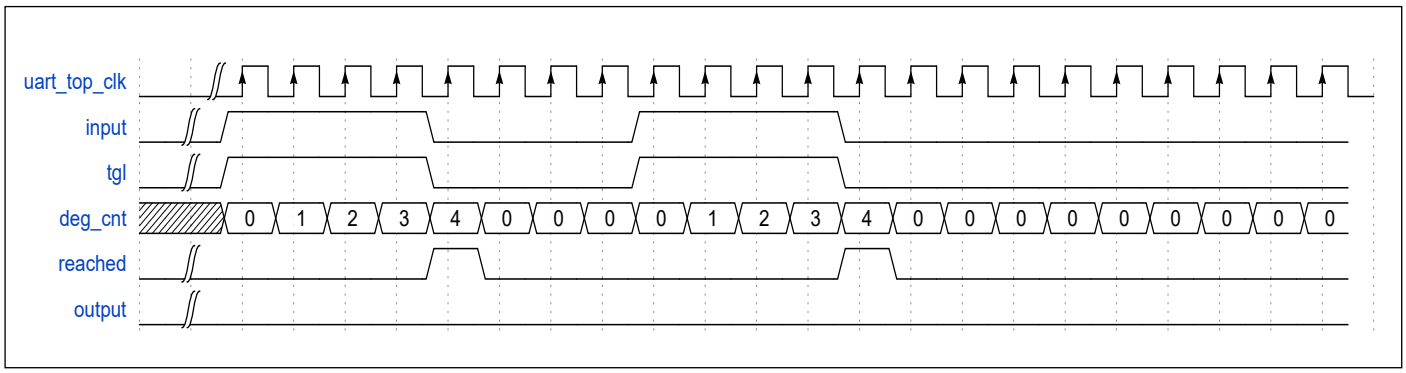


Fig. 16.5: UART Filter Waveform

16.3.6 Transmitter

The transmitter contains a 32-byte TX FIFO to store the data to be sent. When the transmission enable (TxE) 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.

16.3.7 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.

16.3.8 Automatic Baud Rate Detection

UART supports automatic baud rate detection, which includes the general mode and fixed character mode. The two modes will be enabled every time cr_urx_abr_en in the register urx_config is set.

Common 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 mode

In this mode, after UART counts the clocks in the start bit width, it will continue to count the clocks of the subsequent data bits and compare them with that of the start bit. If the fluctuation is within the allowable error range, the detection is passed, and otherwise, count values will be discarded. The allowable error can be set by configuring the cr_urx_abr_pw_tol bit in the register urx_abr_pw_tol, and the unit is the clock source of UART.

Thus, only when the fixed characters ‘0x55’/’0xD5’ under LSBFIRST or the ‘0xAA’/’0xAB’ under MSBFIRST are received, UART will write the clock count value in the start bit width into the high 16 bits sts_urx_abr_prd_0x55 in the register STS_URX_ABR_PRD, as shown below:

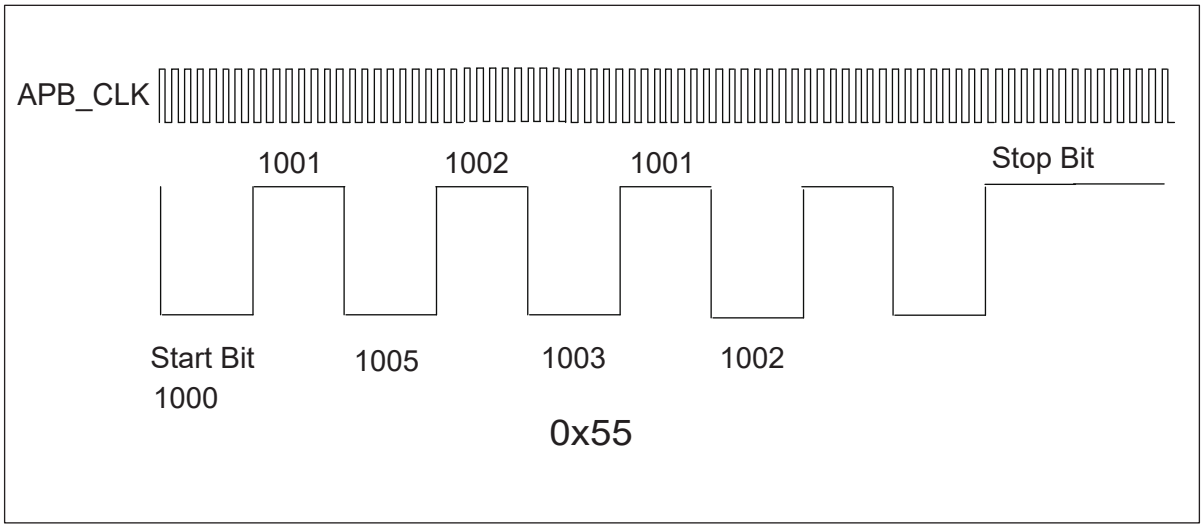


Fig. 16.6: Waveform of UART in fixed character mode

For an unknown baud rate, UART uses UART_CLK to count the width of the start bit and gets a count value of 1000, and the width of the second bit is 1001. If the width fluctuates for not more than four UART_CLK from the previous bit, UART will continue to count the third bit and get a count value of 1005. If the difference between the start bit and the third bit exceeds 4, the detection fails and the data will be discarded. UART compares the width of the first six data bits with that of the start bit in turn.

Formula for calculating the detected baud rate: baud rate = source clock/(16-bit detection value + 1)

16.3.9 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:

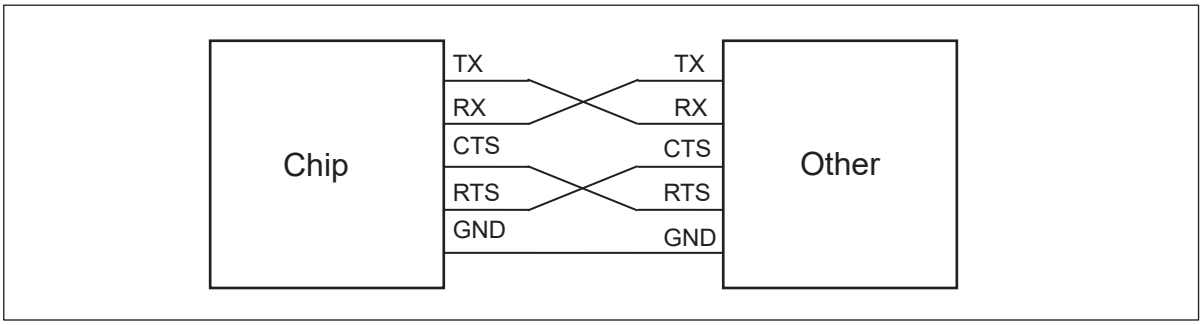


Fig. 16.7: UART hardware flow control

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:

- 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).
- 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 URX_SW_MODE.

16.3.10 DMA Transfer Mode

UART supports the DMA transfer mode. To enable this mode, you must set the thresholds of TX FIFO and RX FIFO through tx_fifo_th and rx_fifo_th in the register uart_fifo_config_1. When this mode is enabled, if tx_fifo_cnt in the uart_fifo_config_1 is greater than tx_fifo_th, the DMA TX request will be triggered. After DMA is configured, when receiving this request, DMA will transfer data from memory to TX FIFO as configured. If the rx_fifo_cnt in uart_fifo_config_1 is larger than rx_fifo_th, the DMA RX request will be triggered. After DMA is configured, when receiving this request, DMA will transfer the data in RX FIFO to the memory as configured. In the transmission mode, to ensure

correct data transfer by the DMA TX Channel, the Channel configuration must meet the condition that transferWidth multiplied by burstSize is less than or equal to the preset tx_fifo_th plus 1.

In the transmission mode, to ensure complete data transfer by the DMA RX Channel, the Channel configuration must meet the condition that transferWidth multiplied by burstSize is equal to the preset tx_fifo_th plus 1.

16.3.11 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, and a typical LIN data transfer is shown as follows.

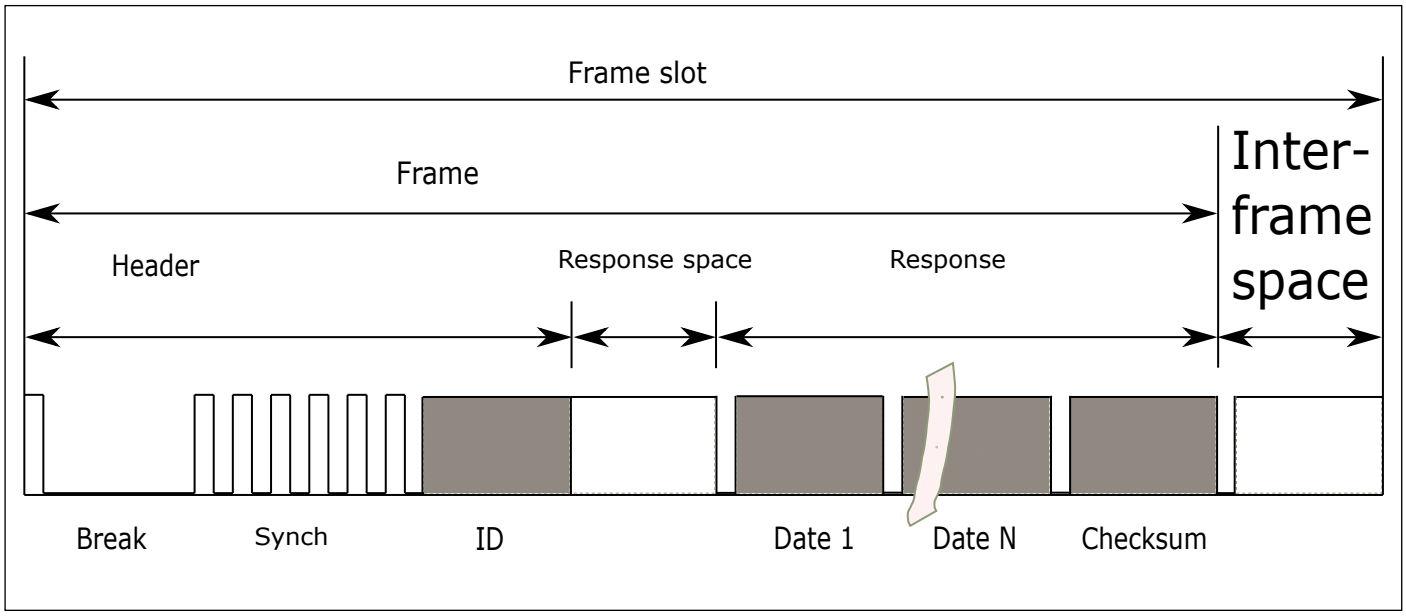


Fig. 16.8: A typical LIN frame

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.

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.

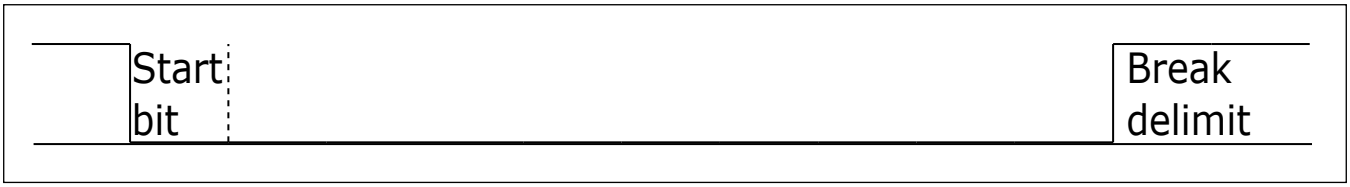


Fig. 16.9: Break Field of LIN

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

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).

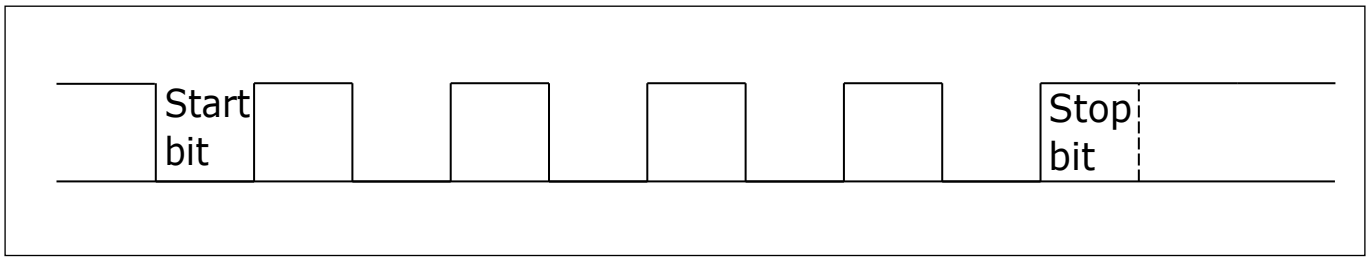


Fig. 16.10: Sync Field of LIN

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.

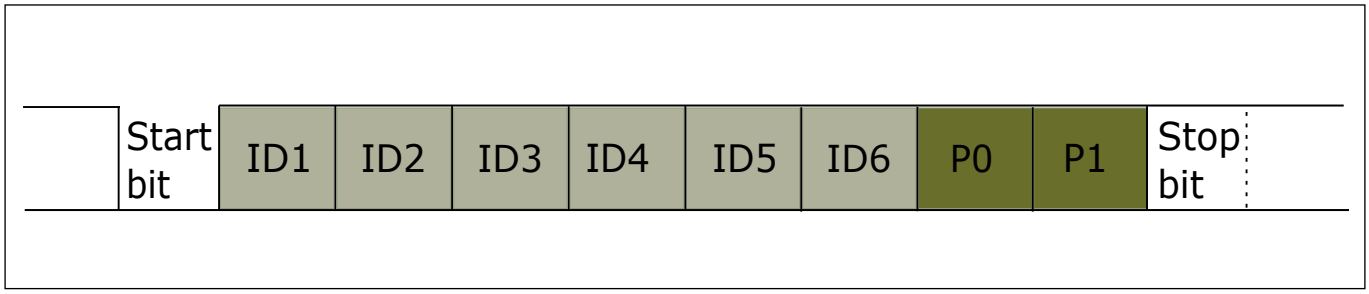


Fig. 16.11: ID Field of LIN

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.

16.3.12 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.

16.3.13 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

16.3.14 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 example, if the TX/RX transfer length is 64 bytes, when the chip sends 63 bytes, no interrupt will be generated;

when it sends 1 byte again, a TX end interrupt will be generated as the number of transferred bytes reaches the preset transfer length of TX. After that, no data can be sent out again and all the data is stored in TX FIFO. When the other side sends 63 bytes to the chip, no interrupt will be generated; but when it sends 1 byte again, a RX end interrupt will be generated as the number of bytes received by the chip reaches the preset transfer length of RX. Then, if it sends 64 bytes to the chip again, UART can receive data and the FIFO data is not empty. When transferlen=15 is set, if TX sends 16-byte data, RX will receive 15-byte data only. The extra one byte is stored in TX FIFO, and the value of TX FIFO COUNT is 1 by default. After TX/RX is disabled and enabled again, the remaining data in TX FIFO will be sent out.

16.3.15 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.

For example, you can set tx_fifo_th/rx_fifo_th to 16 by configuring tx_fifo_th/rx_fifo_th in the register uart_fifo_config_1; enable the free run mode by setting cr_utx_frm_en in the register utx_config; enable the TX/RX FIFO interrupts by setting cr_utx_frdy_mask/cr_urx_frdy_mask in the register uart_int_mask to 0; enable TX/RX by setting cr_utx_en/cr_urx_en in the register utx_config/urx_config. TX FIFO interrupt: TX will always generate FIFO interrupts. After the chip sends 128 bytes, cr_utx_frdy_mask is set to 1 to mask this interrupt. To enable the TX FIFO interrupt again, you need to set cr_utx_frdy_mask to 0. RX FIFO interrupt: When the other side sends 15 bytes, no interrupt is generated and the value of rx_fifo_cnt is 15. When one byte is sent again, an interrupt will be generated as the preset value of rx_fifo_th is met. After a TX interrupt is generated, if the other side sends data again, the chip can receive the data.

16.3.16 RX timeout interrupt

The RX timeout interrupt will be triggered when the receiver fails to receive data beyond the timeout threshold. The timeout threshold is measured by Bit.

When the other side sends data to the chip, a timeout interrupt will be generated after the preset timeout threshold is reached. For example, you can enable the timeout interrupt by setting the cr_urx_rto_mask bit in the register uart_int_mask to 0, and data is received during the timeout interrupt. Through the printed RX buffer and the print log, you can confirm the timeout interrupt state and that correct data is received.

16.3.17 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.

16.3.18 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.

16.3.19 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”.

16.3.20 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.

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

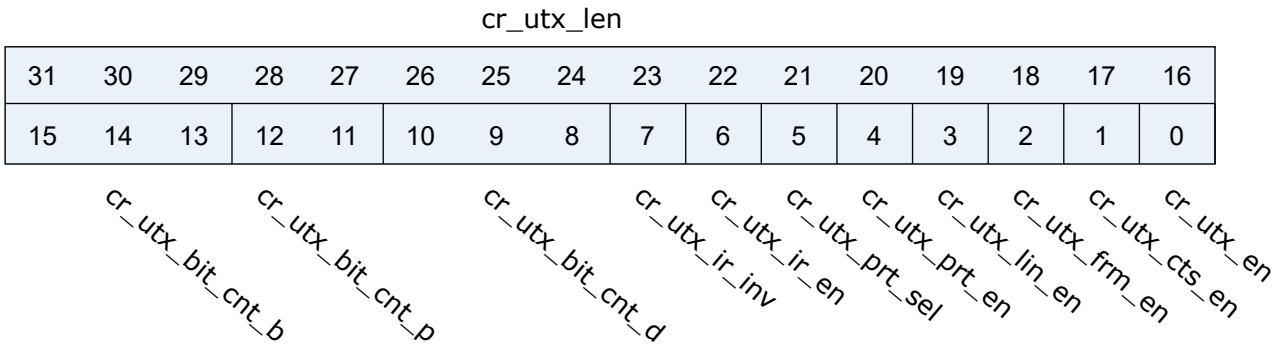
In the auto baud rate detection mode, when a baud rate is detected, the auto baud rate detection (universal/fixed characters mode) interrupt will be generated as configured.

16.4 Register description

Name	Description
utx_config	
urx_config	
uart_bit_prd	
data_config	
utx_ir_position	
urx_ir_position	
urx_rto_timer	
uart_sw_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	
sts_urx_abr_prd	
urx_abr_prd_b01	
urx_abr_prd_b23	
urx_abr_prd_b45	
urx_abr_prd_b67	
urx_abr_pw_tol	
urx_bcr_int_cfg	
utx_rs485_cfg	
uart_fifo_config_0	
uart_fifo_config_1	
uart_fifo_wdata	
uart_fifo_rdata	

16.4.1 utx_config

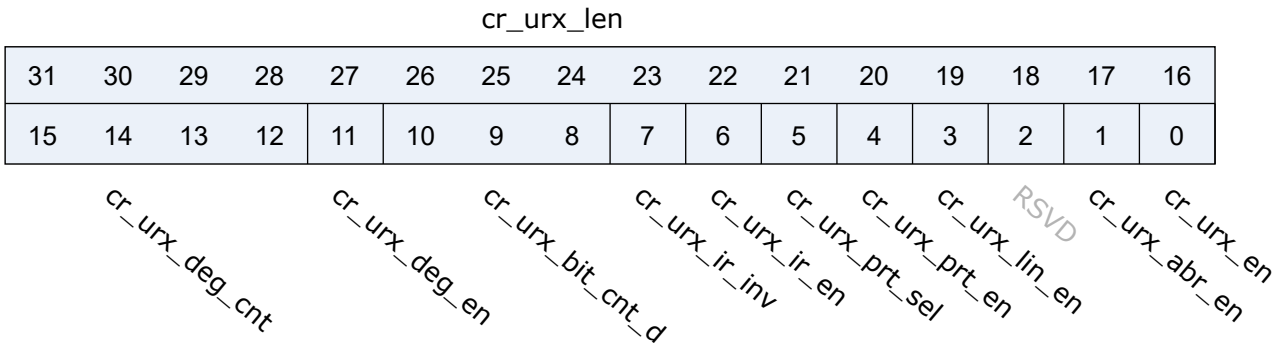
Address：0x2000a000



Bits	Name	Type	Reset	Description
31:16	cr_utx_len	r/w	16'd0	Length of UART TX data transfer (Unit: character/byte) (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
12:11	cr_utx_bit_cnt_p	r/w	2'd1	UART TX STOP bit count (unit: 0.5 bit)
10:8	cr_utx_bit_cnt_d	r/w	3'd7	UART TX DATA bit count for each character
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)
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

16.4.2 urx_config

Address：0x2000a004



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
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			
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

16.4.3 uart_bit_prd

Address：0x2000a008

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, related to baud rate
15:0	cr_utx_bit_prd	r/w	16'd255	Period of each UART TX bit, related to baud rate

16.4.4 data_config

Address：0x2000a00c

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			
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

16.4.5 utx_ir_position

Address：0x2000a010

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

16.4.6 urx_ir_position

Address：0x2000a014

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

16.4.7 urx_rto_timer

Address：0x2000a018

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

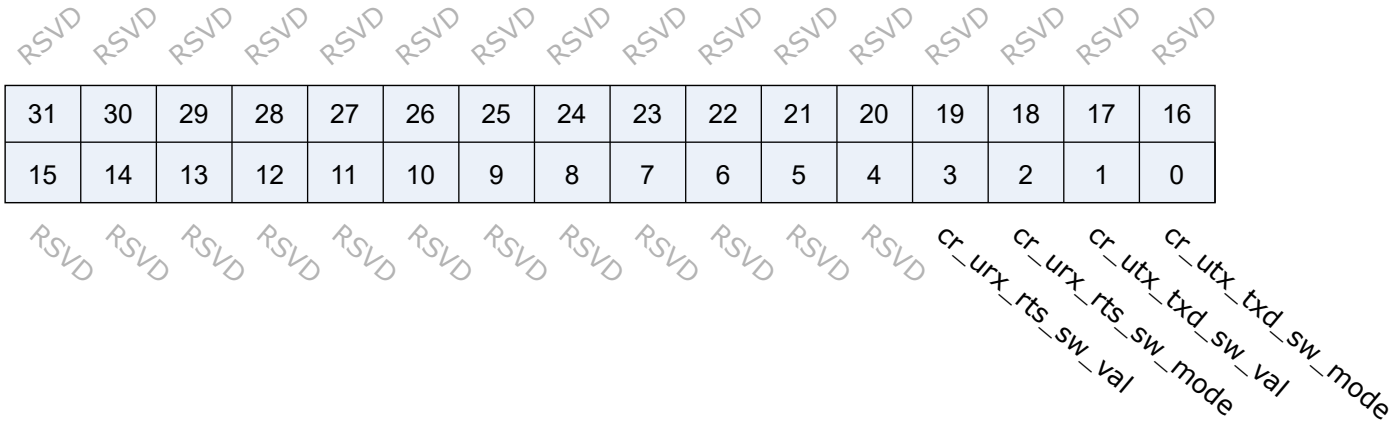
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD cr_urx_rto_value

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)

16.4.8 uart_sw_mode

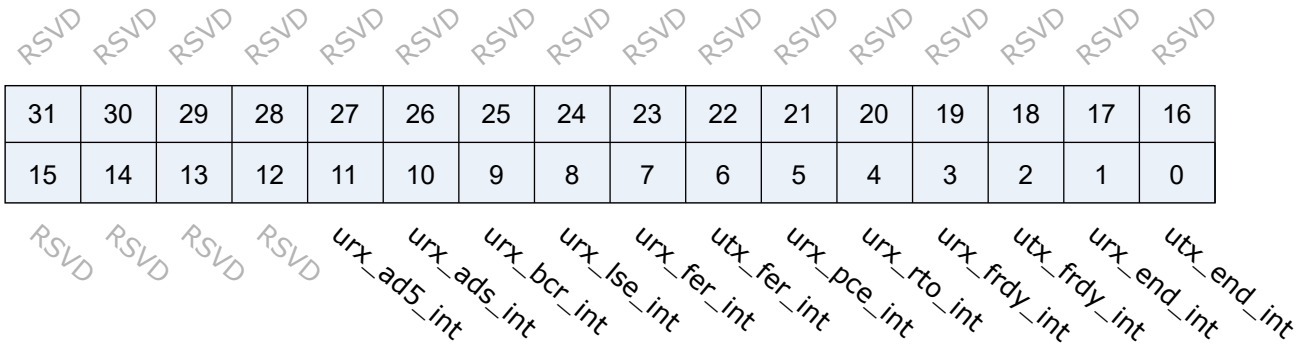
Address：0x2000a01c



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
2	cr_urx_rts_sw_mode	r/w	1'b0	UART RX RTS output SW control mode
1	cr_utx_txd_sw_val	r/w	1'b0	UART TX TXD output SW control value
0	cr_utx_txd_sw_mode	r/w	1'b0	UART TX TXD output SW control mode

16.4.9 uart_int_sts

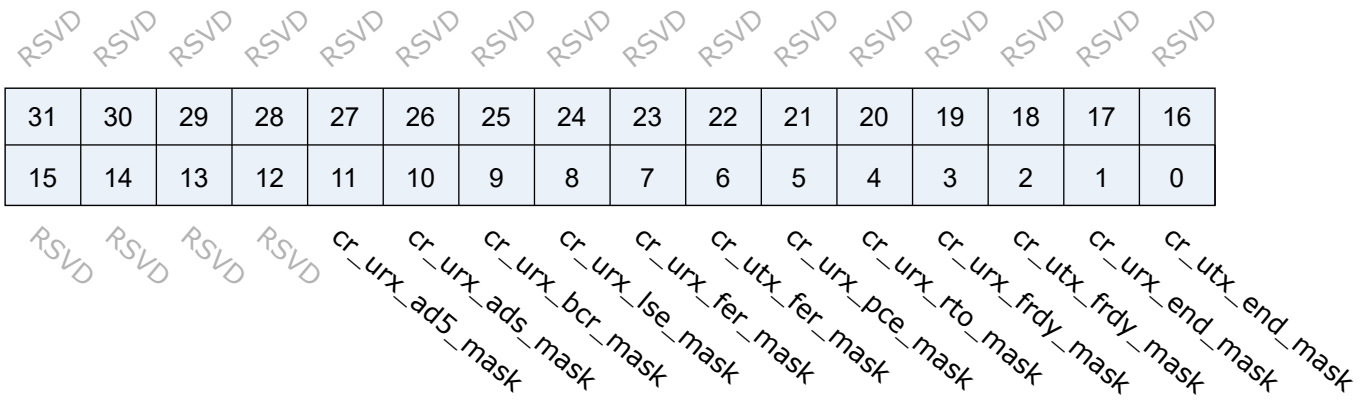
Address：0x2000a020



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)

16.4.10 uart_int_mask

Address：0x2000a024



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

16.4.11 uart_int_clear

Address：0x2000a028

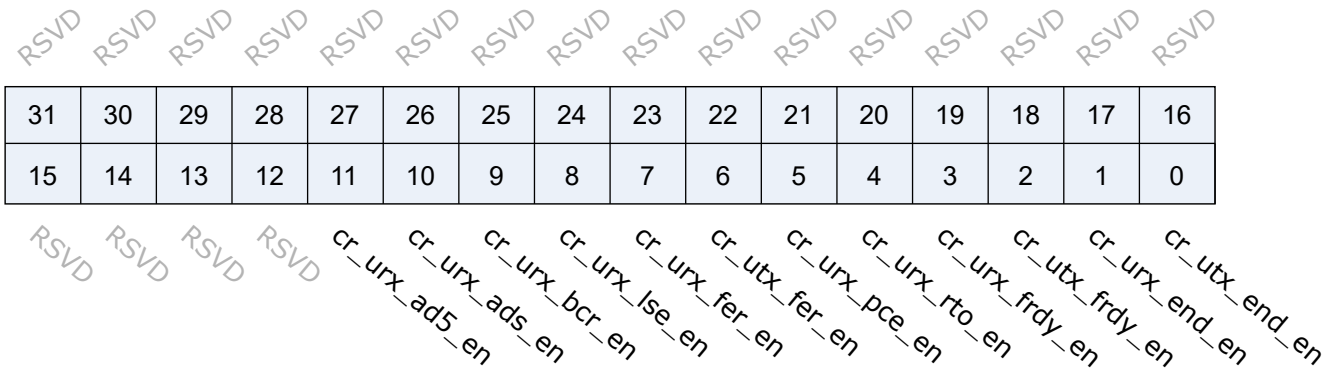
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	cr_urx_ad5_clr	cr_urx_ads_clr	cr_urx_bcr_clr	cr_urx_lse_clr	rsvd	rsvd	cr_urx_pce_clr	cr_urx_rto_clr	rsvd	rsvd	cr_urx_end_clr	cr_utx_end_clr

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

Bits	Name	Type	Reset	Description
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

16.4.12 uart_int_en

Address：0x2000a02c

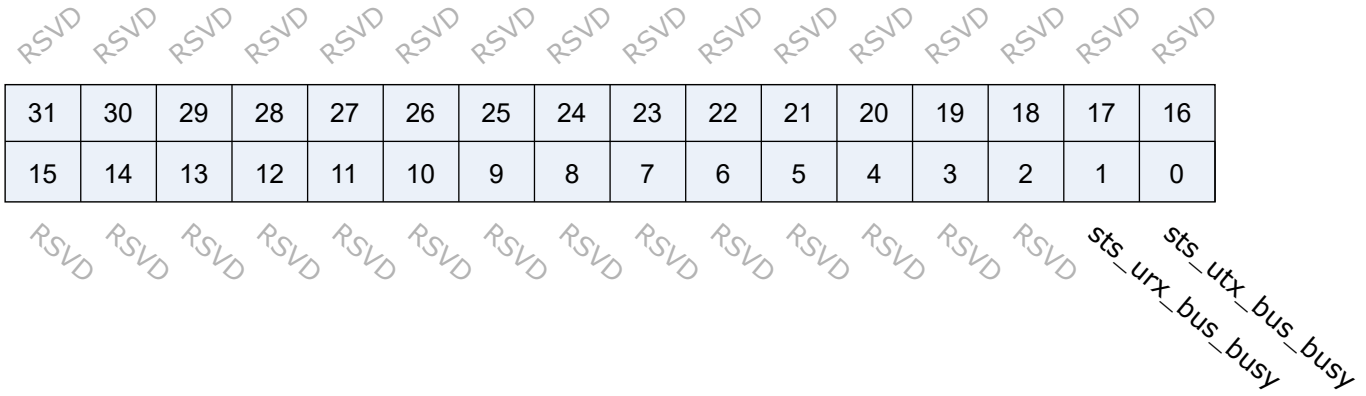


Bits	Name	Type	Reset	Description
31:12	RSVD			
11	cr_urx_ad5_en	r/w	1'b1	Interrupt enable of urx_ad5_int
10	cr_urx_ads_en	r/w	1'b1	Interrupt enable of urx_ads_int
9	cr_urx_bcr_en	r/w	1'b1	Interrupt enable of urx_bcr_int
8	cr_urx_lse_en	r/w	1'b1	Interrupt enable of urx_lse_int
7	cr_urx_fer_en	r/w	1'b1	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

Bits	Name	Type	Reset	Description
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

16.4.13 uart_status

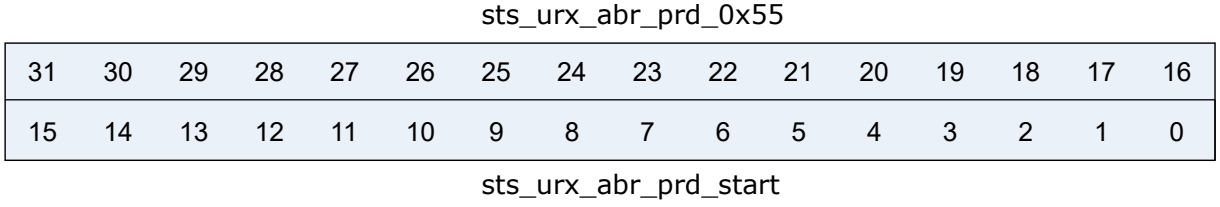
Address：0x2000a030



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

16.4.14 sts_urx_abr_prd

Address：0x2000a034



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

Bits	Name	Type	Reset	Description
15:0	sts_urx_abr_prd_start	r	16'd0	Bit period of Auto Baud Rate detection using START bit

16.4.15 urx_abr_prd_b01

Address：0x2000a038

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]

16.4.16 urx_abr_prd_b23

Address：0x2000a03c

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]

16.4.17 urx_abr_prd_b45

Address：0x2000a040

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]

16.4.18 urx_abr_prd_b67

Address：0x2000a044

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]

16.4.19 urx_abr_pw_tol

Address：0x2000a048

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

16.4.20 urx_bcr_int_cfg

Address：0x2000a050

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

16.4.21 utx_rs485_cfg

Address：0x2000a054

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

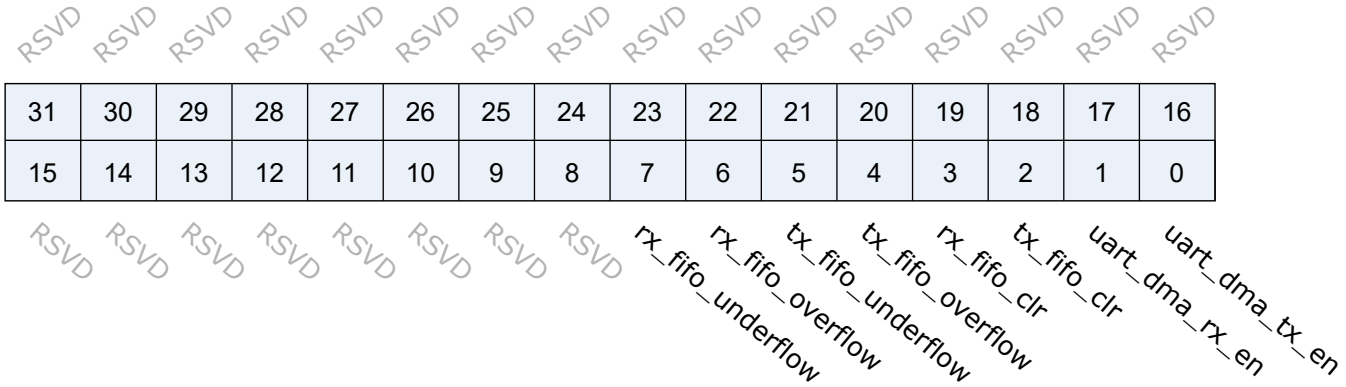
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

16.4.22 uart_fifo_config_0

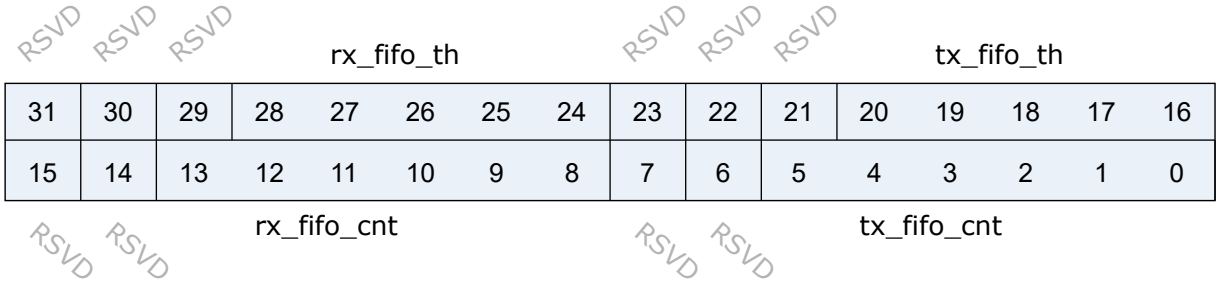
Address：0x2000a080



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
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
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

16.4.23 uart_fifo_config_1

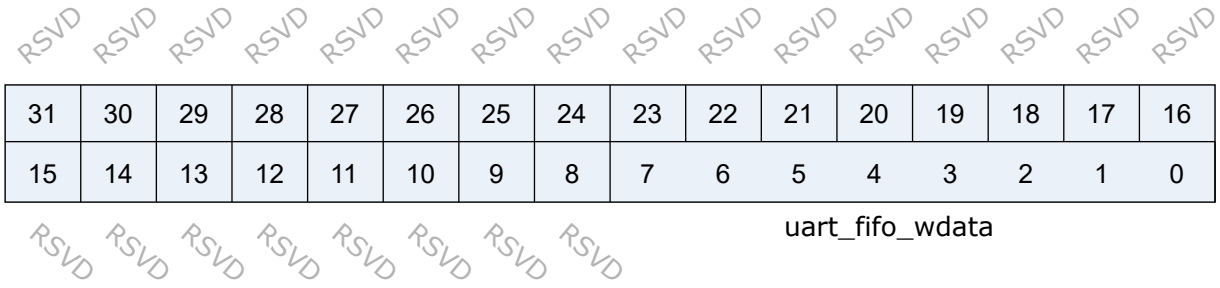
Address：0x2000a084



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
7:6	RSVD			
5:0	tx_fifo_cnt	r	6'd32	TX FIFO available count

16.4.24 uart_fifo_wdata

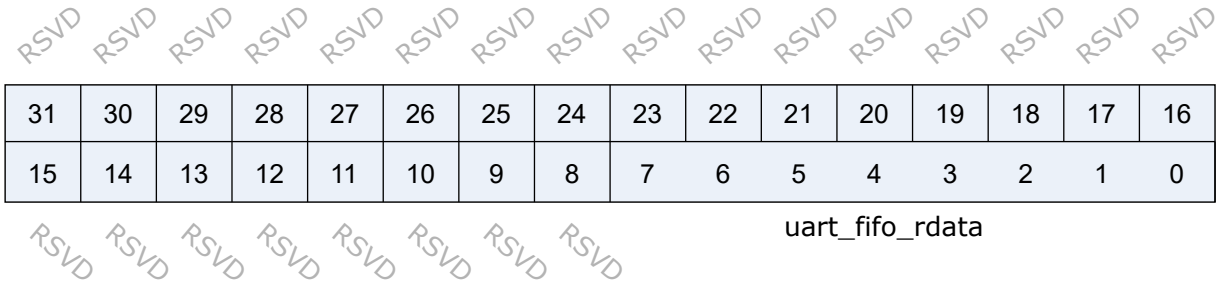
Address：0x2000a088



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	uart_fifo_wdata	w	x	

16.4.25 uart_fifo_rdata

Address：0x2000a08c



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	uart_fifo_rdata	r	8'h0	

17 I2C

17.1 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.

BL808 has four 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.

17.2 Features

- Master mode
- Multi-master mode and arbitration function
- Flexible adjustment of clock frequency
- Supports 10-bit address mode
- Supports DMA transfer mode

17.3 Functional Description

Table 17.1: I2C pin list

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

17.3.1 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:

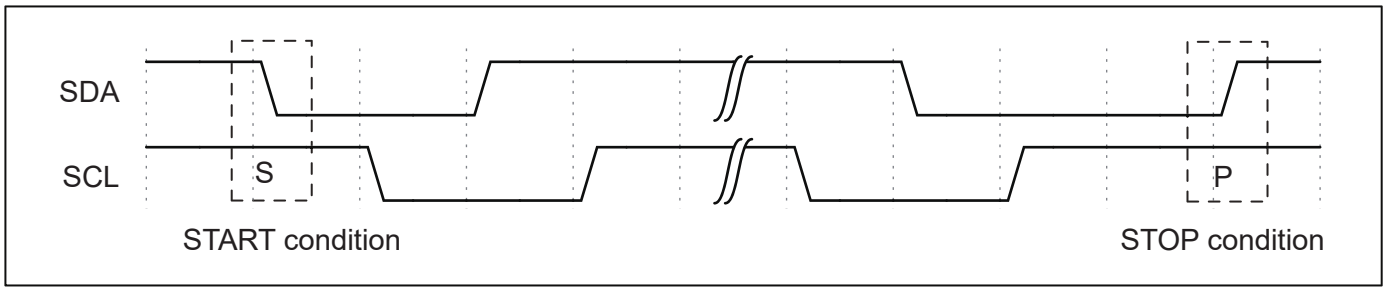


Fig. 17.1: Start and stop conditions of I2C

17.3.2 Data Transfer Format

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, followed by the slave sending out an acknowledgement (ACK) bit. Upon data transfer completed, the master sends out a STOP signal, with waveform shown below:

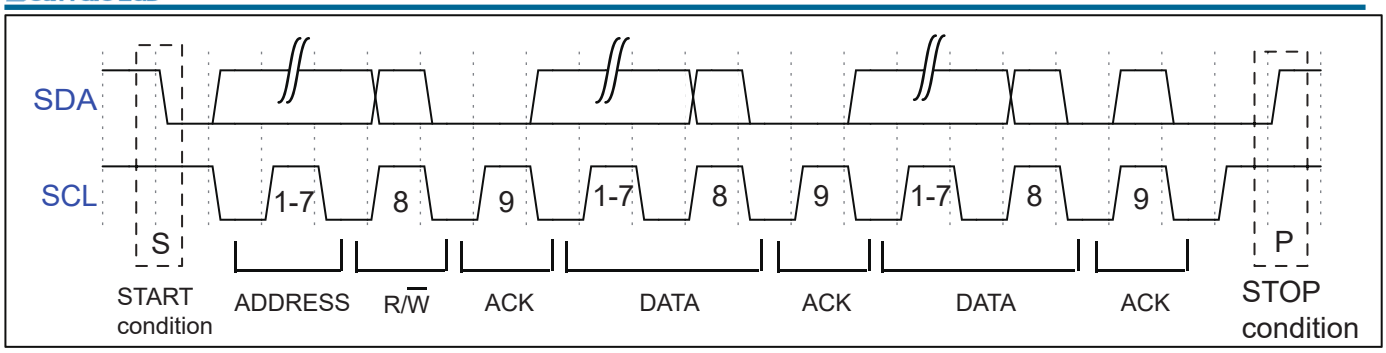


Fig. 17.2: I2C data transfer format

Master Transmit and Slave Receive Timing

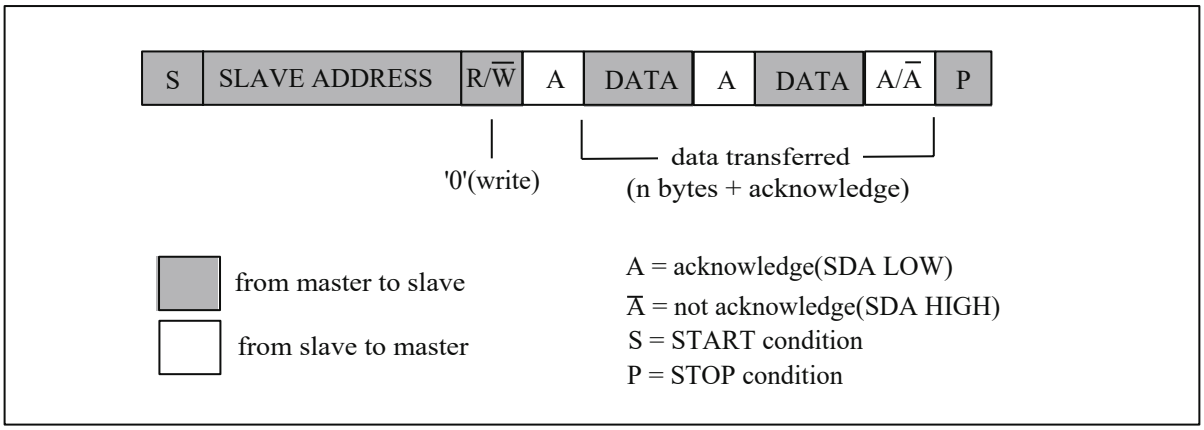


Fig. 17.3: Timing of master transmitter and slave receiver

Master Receive and Slave Transmit Timing

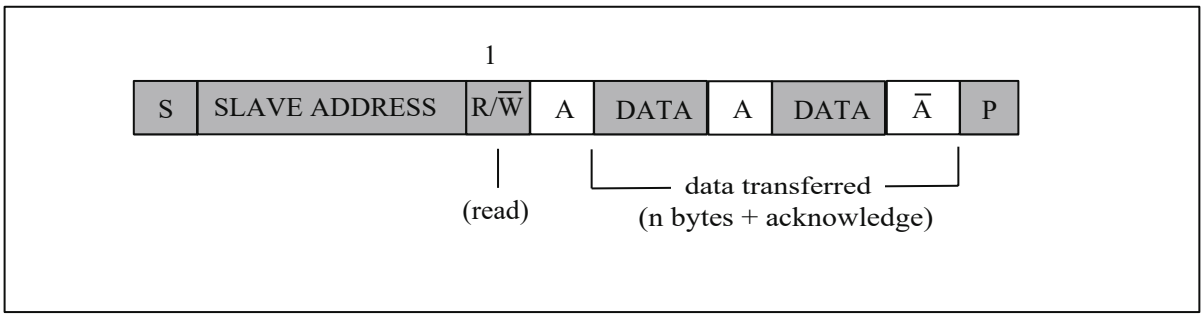


Fig. 17.4: Timing of master receiver and slave transmitter

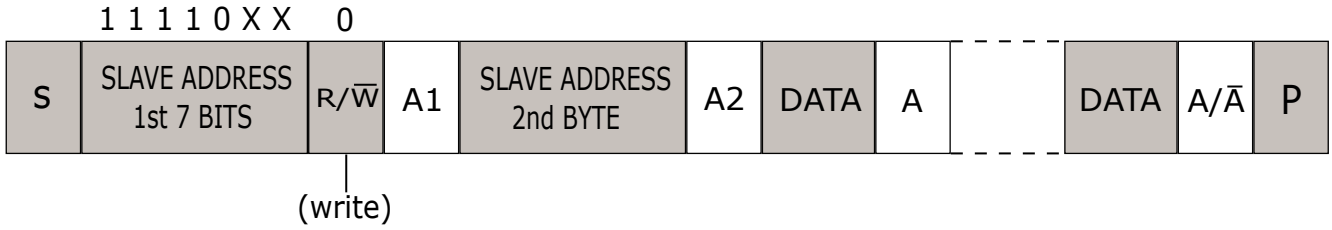
10-bit address mode: The cr_i2c_10b_addr_en in the register i2c_config must be set to 1 before use.

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. Although there are 8 possible combinations

of 1111 XXX, only the four types of 1111 0XX can be used for 10-bit addressing, and the remaining four types of 1111 1XX are used for future I2C expansion.

The aforementioned read-write formats for 7-bit addressing all suit 10-bit addressing, as follows:

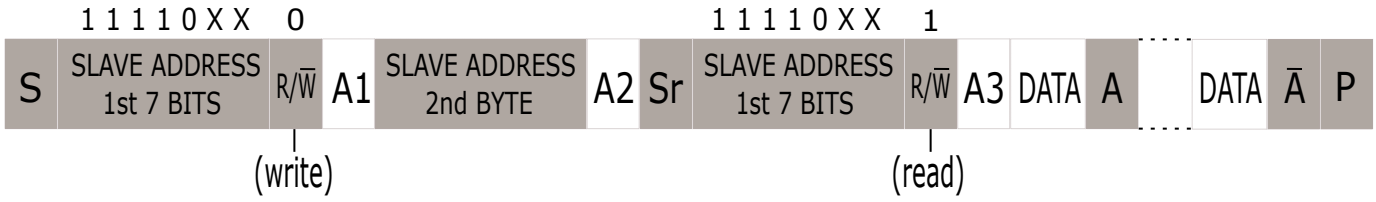
1. A master-transmitter sends data to a slave-receiver with a 10-bit slave address



A master-transmitter addresses a slave-receiver with a 10-bits address

The figure shows that the transfer direction is unchanged. 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. It is possible that multiple devices all match and generate an acknowledgement (A1). Next, all the slaves start to match their own addresses with the 8 bits of the second byte (XXXX XXXX). At this time, only one slave matches and generates an acknowledgement (A2). The slave that is addressed by the master will remain in the addressed state until it receives a STOP condition or a repeated START condition, followed by a different slave address.

2. A master-receiver receives data from a slave-transmitter (10-bit slave address)



A master-receiver addresses a slave-transmitter with a 10-bits address

The transfer direction will change after the second read/write bit. Before the second acknowledgement A2, 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 correct, 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 (A3). 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.

17.3.3 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:

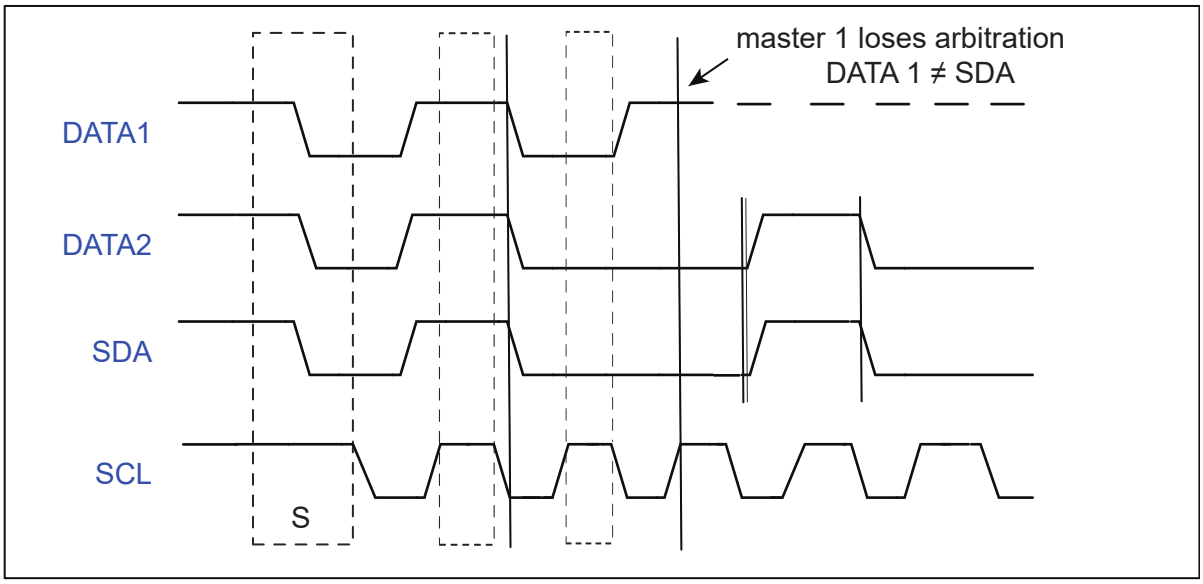


Fig. 17.5: Waveform of simultaneous data transfer

17.4 I2C Clock Setting

I2C clock can be derived from bclk (bus clock) and xclk, and frequency division can be done on this basis. The register i2c_prd_data can divide the clock of the data segment. The I2C module divides data transfer into 4 stages. Each stage is controlled by a single byte in the register, and the number of samples of each stage can be set. The 4 numbers jointly determine the division factor of i2c clock.

For example, bclk is 32M now, and the default value of the register i2c_prd_data is 0x0f0f0f0f without configuration, so I2C’s clock frequency is 32M/((15 + 1) * 4) = 500K. Similarly, registers i2c_prd_start and i2c_prd_stop will divide the clock of the start and stop bits, respectively.

17.5 I2C Configuration Flow

17.5.1 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

17.5.2 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.

17.5.3 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.

17.5.4 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.

17.5.5 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.

17.5.6 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.

17.5.7 Data Length

The data length is subtracted by 1 and then written to cr_i2c_pkt_len in the register i2c_config.

17.5.8 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 and 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 and 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

17.6 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.

17.7 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.

17.7.1 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 desti-

nation 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

17.7.2 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

17.8 Interrupt

I2C includes the following interrupts:

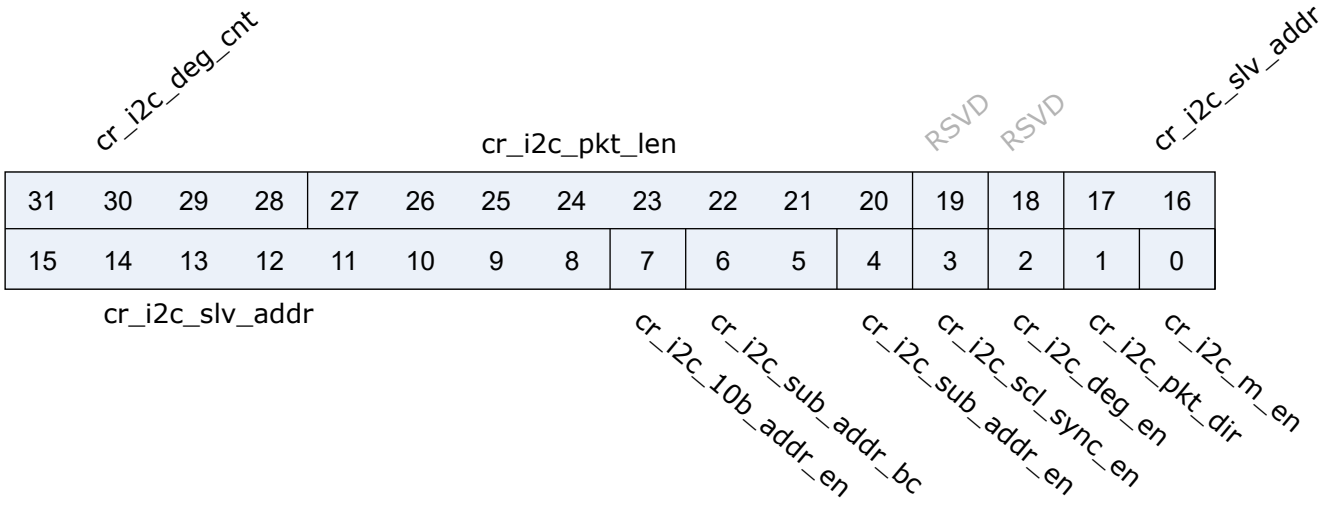
- I2C_TRANS_END_INT: I2C end of transfer interrupt
- I2C_TX_FIFO_READY_INT: The interrupt is triggered when I2C TX FIFO has free space for padding
- I2C_RX_FIFO_READY_INT: The interrupt is triggered when I2C RX FIFO receives data
- I2C_NACK_RECV_INT: The interrupt is triggered when I2C detects the NACK state
- I2C_ARB_LOST_INT: I2C arbitration lost interrupt
- I2C_FIFO_ERR_INT: I2C FIFO ERROR interrupt

17.9 Register description

Name	Description
i2c_config	
i2c_int_sts	
i2c_sub_addr	
i2c_bus_busy	
i2c_prd_start	
i2c_prd_stop	
i2c_prd_data	
i2c_fifo_config_0	
i2c_fifo_config_1	
i2c_fifo_wdata	
i2c_fifo_rdata	

17.9.1 i2c_config

Address：0x2000a300

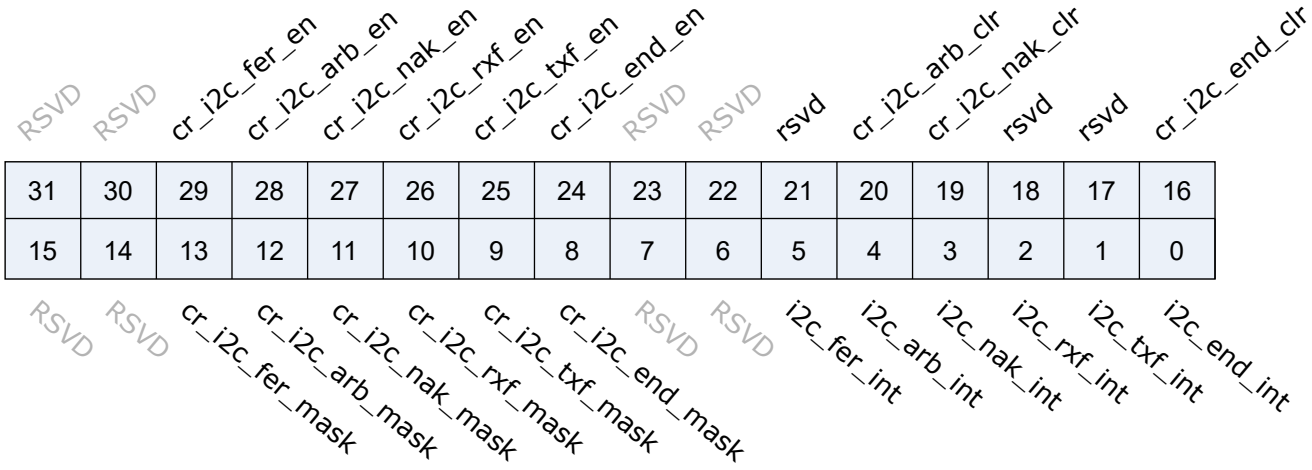


Bits	Name	Type	Reset	Description
31:28	cr_i2c_deg_cnt	r/w	4'd0	De-glitch function cycle count
27:20	cr_i2c_pkt_len	r/w	8'd0	Packet length (unit: byte)

Bits	Name	Type	Reset	Description
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
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

17.9.2 i2c_int_sts

Address：0x2000a304



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

Bits	Name	Type	Reset	Description
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
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

17.9.3 i2c_sub_addr

Address：0x2000a308

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)

17.9.4 i2c_bus_busy

Address：0x2000a30c

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_i2c_bus_busy_clr	sts_i2c_bus_busy

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

17.9.5 i2c_prd_start

Address：0x2000a310

cr_i2c_prd_s_ph_3								cr_i2c_prd_s_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_s_ph_1								cr_i2c_prd_s_ph_0							

Bits	Name	Type	Reset	Description
31:24	cr_i2c_prd_s_ph_3	r/w	8'd15	Length of START condition phase 3
23:16	cr_i2c_prd_s_ph_2	r/w	8'd15	Length of START condition phase 2
15:8	cr_i2c_prd_s_ph_1	r/w	8'd15	Length of START condition phase 1
7:0	cr_i2c_prd_s_ph_0	r/w	8'd15	Length of START condition phase 0

17.9.6 i2c_prd_stop

Address：0x2000a314

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
23:16	cr_i2c_prd_p_ph_2	r/w	8'd15	Length of STOP condition phase 2
15:8	cr_i2c_prd_p_ph_1	r/w	8'd15	Length of STOP condition phase 1
7:0	cr_i2c_prd_p_ph_0	r/w	8'd15	Length of STOP condition phase 0

17.9.7 i2c_prd_data

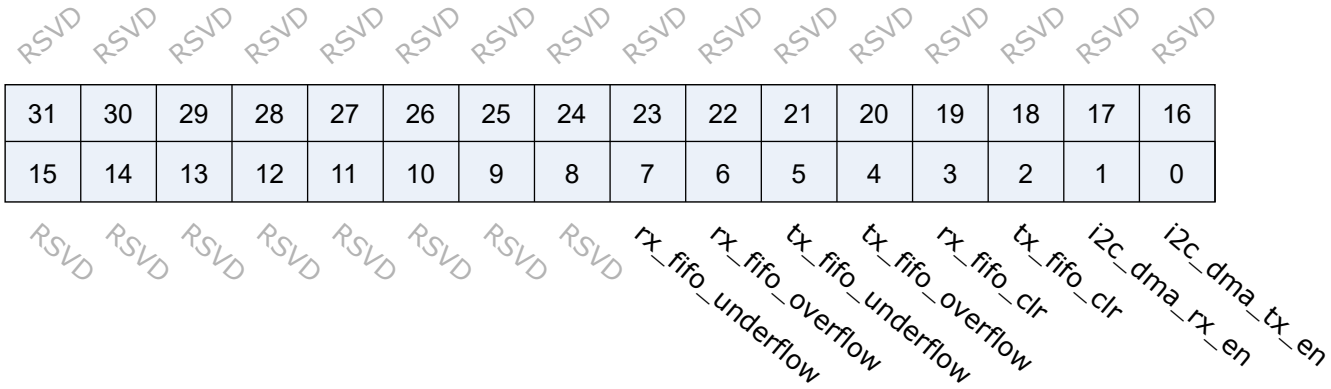
Address：0x2000a318

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
23:16	cr_i2c_prd_d_ph_2	r/w	8'd15	Length of DATA phase 2
15:8	cr_i2c_prd_d_ph_1	r/w	8'd15	Length of DATA phase 1 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

17.9.8 i2c_fifo_config_0

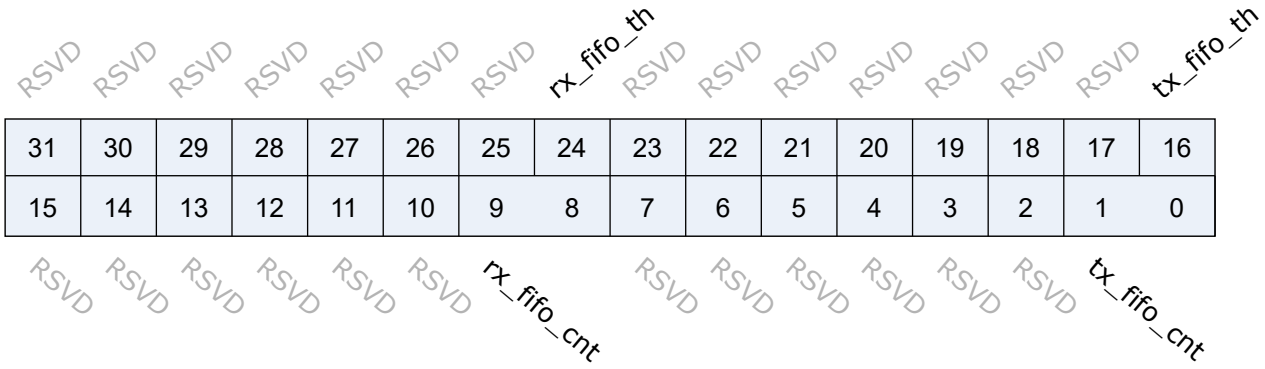
Address：0x2000a380



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
2	tx_fifo_clr	w1c	1'b0	Clear signal of TX FIFO
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

17.9.9 i2c_fifo_config_1

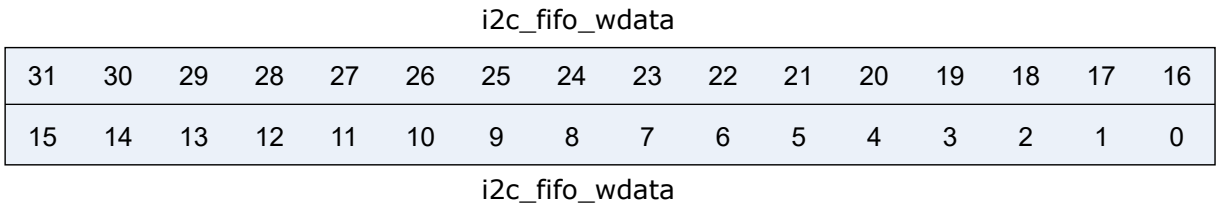
Address：0x2000a384



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
7:2	RSVD			
1:0	tx_fifo_cnt	r	2'd2	TX FIFO available count

17.9.10 i2c_fifo_wdata

Address：0x2000a388



Bits	Name	Type	Reset	Description
31:0	i2c_fifo_wdata	w	x	

17.9.11 i2c_fifo_rdata

Address：0x2000a38c

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	

18 PWM

18.1 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.

18.2 Features

- Two 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 (bus clock<bclk>, crystal oscillator clock<xtal_ck>, and slow clock<32k>), with 16-bit clock divider
- Each PWM can be independently set to different cycles
- 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
- Interrupts will be generated when these events occur: counter overflow, threshold comparison and matching, brake signal generation, and PWM cycles reaching the set value.

18.3 Functional Description

18.3.1 Clock and Frequency Divider

Each PWM counter has three optional clock sources:

1. bclk-bus clock of chip
2. XTAL-external crystal oscillator clock
3. f32k-RTC clock of system

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.

18.3.2 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 pwmx (x = 0, 1) 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.

18.3.3 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 pwmx (x = 0, 1) 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 pwmx (x = 0, 1) 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.

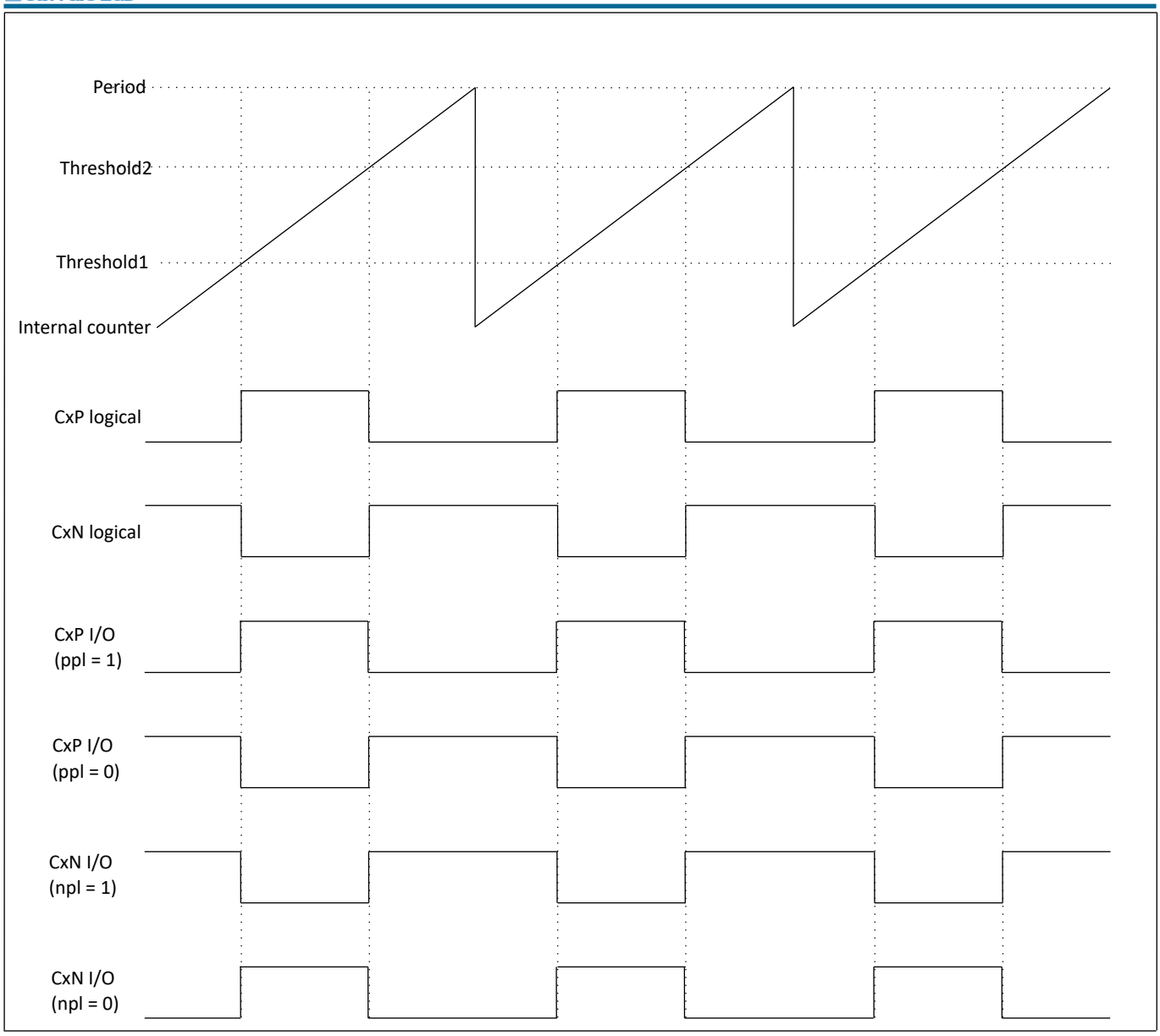


Fig. 18.1: Waveform of PWM in different configurations

18.3.4 Brake

When the brake function is used and the brake signal is generated, the level of the PWM output will be changed to a preset state, which is determined by the <pwm_chy_pbs> and <pwm_chy_nbs> (y = 0, 1, 2, 3) bits in the register pwm_mcx_config1 (x = 0, 1). When the bits are set to 1, PWM outputs logical ‘1’ after the brake signal is generated. When the same is set to 0, PWM outputs logical ‘0’ after the brake signal is generated.

Brake signals include external brake and software brake. When the <pwm_sw_break_en> in pwm_mcx_config0 (x = 0, 1) is 1, the internal brake signal will be generated. When this bit is 0, the brake state will be exited and PWM will resume to its previous operation mode. When the <pwm_ext_break_en> bit is 1, the external brake function is enabled. If the external brake pin state matches the value set by <pwm_ext_break_pl>, a brake signal will be generated. If that state is opposite to the value set by <pwm_ext_break_pl>, PWM will exit the brake state. It should

be noted that only PWM0 supports external brake, and PWM1 cannot do that.

If the brake interrupt is enabled, the interrupt will be triggered when the external brake signal is generated. Software brake will not trigger an interrupt.

18.3.5 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 threshold 1, and immediately change the level state when it matches the threshold 2. The complementary channel of PWM will change the level state immediately when it matches the threshold 1, and delay the generation of jump level when it matches the threshold 2. The length of dead zone is set by the register pwm_mcx_dead_time.

18.3.6 Cycle and Duty Ratio Calculation

The PWM cycle is determined the clock division factor and the clock duration cycle. The clock division factor is set by the register PWMx_CLK_DIV[15:0](x: 0~1) and is used to divide the source clock of PWM. The clock duration cycle is set by the register PWMx_PERIOD[15:0](x: 0~1) and is used to set how many divided clock cycles a PWM cycle consists of. That is, PWM cycle = PWM source clock/PWMx_CLK_DIV[15:0]/PWMx_PERIOD[15:0].

18.3.7 PWM Interrupt

For each PWM, the cycle count value can be set by the high 16 bits of the register pwm_mcx_period. When the number of PWM cycles reaches this value, the PWM interrupt will be generated.

When the PWM count value reaches the number of cycles or it matches the threshold, the PWM interrupt will be generated.

A PWM interrupt will be generated when an external brake signal is generated.

18.3.8 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.

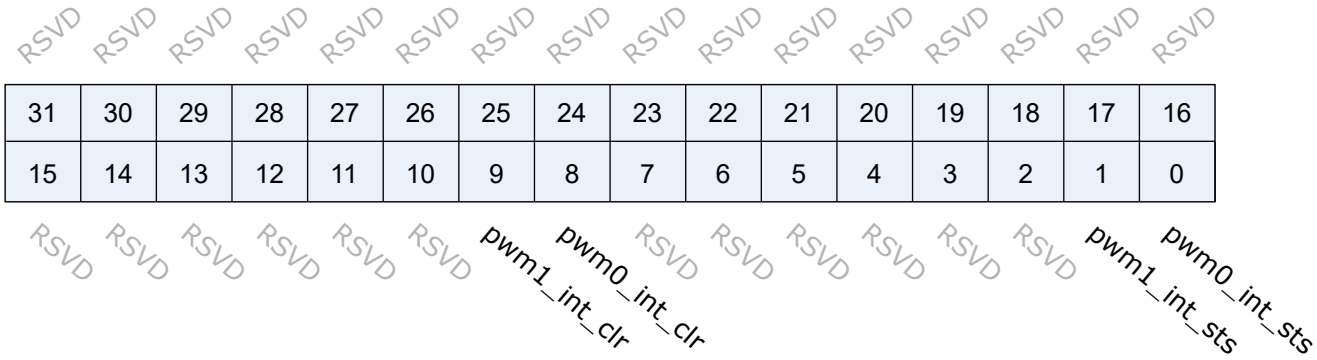
18.4 Register description

Name	Description
pwm_int_config	
pwm_mc0_config0	
pwm_mc0_config1	
pwm_mc0_period	
pwm_mc0_dead_time	
pwm_mc0_ch0_thre	
pwm_mc0_ch1_thre	
pwm_mc0_ch2_thre	
pwm_mc0_ch3_thre	
pwm_mc0_int_sts	
pwm_mc0_int_mask	
pwm_mc0_int_clear	
pwm_mc0_int_en	
pwm_mc1_config0	
pwm_mc1_config1	
pwm_mc1_period	
pwm_mc1_dead_time	
pwm_mc1_ch0_thre	
pwm_mc1_ch1_thre	
pwm_mc1_ch2_thre	
pwm_mc1_ch3_thre	
pwm_mc1_int_sts	
pwm_mc1_int_mask	

Name	Description
pwm_mc1_int_clear	
pwm_mc1_int_en	

18.4.1 pwm_int_config

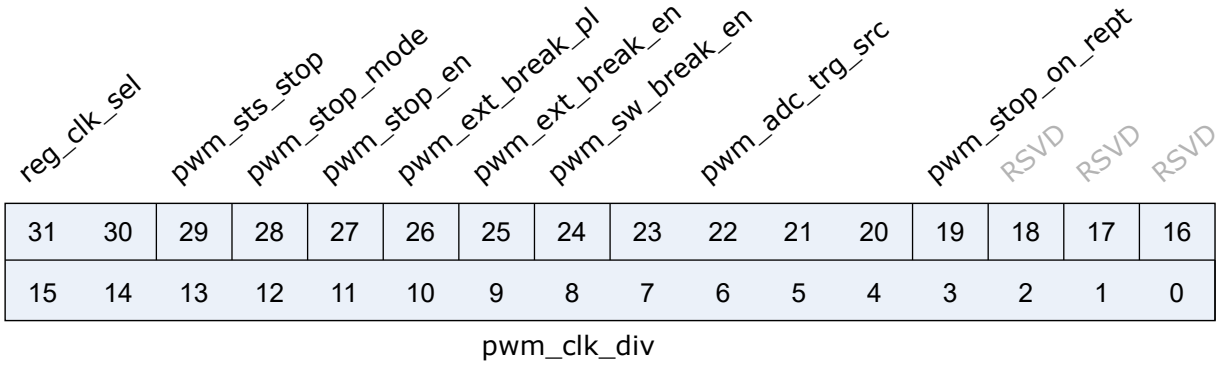
Address：0x2000a400



Bits	Name	Type	Reset	Description
31:10	RSVD			
9	pwm1_int_clr	w1c	1'b0	PWM 1 interrupt clear (clear pwm_mc1_int_sts[10:0] all)
8	pwm0_int_clr	w1c	1'b0	PWM 0 interrupt clear (clear pwm_mc0_int_sts[10:0] all)
7:2	RSVD			
1	pwm1_int_sts	r	1'b0	PWM 1 interrupt status (Check pwm_mc1_int_sts for detailed interrupt status)
0	pwm0_int_sts	r	1'b0	PWM 0 interrupt status (Check pwm_mc0_int_sts for detailed interrupt status)

18.4.2 pwm_mc0_config0

Address：0x2000a440

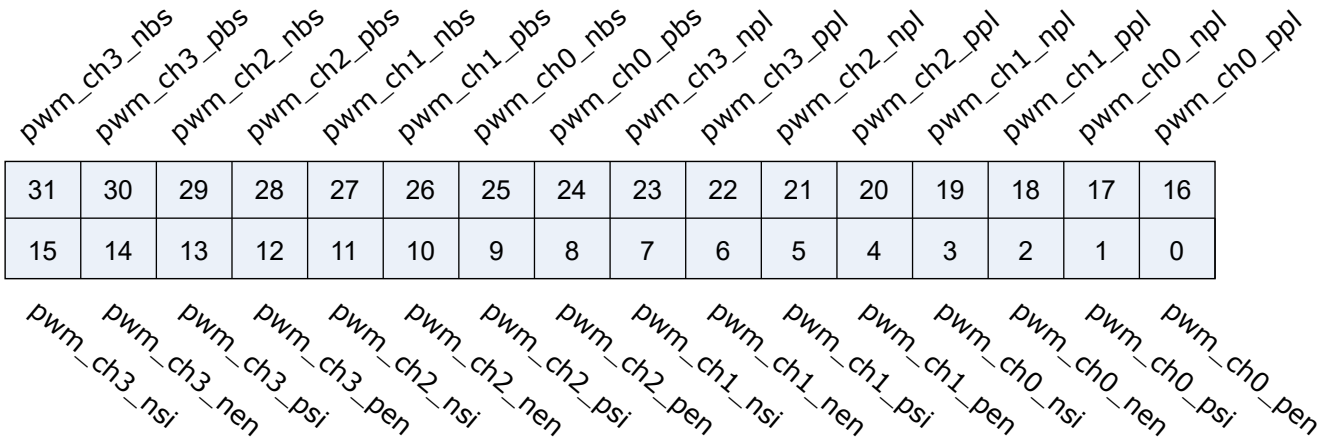


Bits	Name	Type	Reset	Description
31:30	reg_clk_sel	r/w	2'd0	PWM clock source select, 2'b00-xclk ; 2'b01-bclk ; others-f32k_clk
29	pwm_sts_stop	r	1'b0	PWM stop status
28	pwm_stop_mode	r/w	1'b1	PWM stop mode, 1'b1 - graceful ; 1'b0 - abrupt
27	pwm_stop_en	r/w	1'b0	PWM stop enable
26	pwm_ext_break_pl	r/w	1'b0	PWM external break source polarity 1'b0: Active-LOW 1'b1: Active-HIGH
25	pwm_ext_break_en	r/w	1'b0	PWM external break source enable 1'b0: Disabled, external break signal is masked 1'b1: Enabled, external break signal is effective
24	pwm_sw_break_en	r/w	1'b0	PWM break enable 1'b0: Disabled, normal operation 1'b1: 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

Bits	Name	Type	Reset	Description
19	pwm_stop_on_rept	r/w	1'b0	PWM stopped when rept_int is asserted 1'b0: Disabled, PWM keeps running when rept_int is asserted 1'b1: Enabled, PWM stops when rept_int is asserted. Clear rept_int to restore operation
18:16	RSVD			
15:0	pwm_clk_div	r/w	16'b0	PWM clock division

18.4.3 pwm_mc0_config1

Address：0x2000a444



Bits	Name	Type	Reset	Description
31	pwm_ch3_nbs	r/w	1'b0	PWM channel 3 negative break state
30	pwm_ch3_pbs	r/w	1'b0	PWM channel 3 positive break state
29	pwm_ch2_nbs	r/w	1'b0	PWM channel 2 negative break state
28	pwm_ch2_pbs	r/w	1'b0	PWM channel 2 positive break state
27	pwm_ch1_nbs	r/w	1'b0	PWM channel 1 negative break state
26	pwm_ch1_pbs	r/w	1'b0	PWM channel 1 positive break state
25	pwm_ch0_nbs	r/w	1'b0	PWM channel 0 negative break state
24	pwm_ch0_pbs	r/w	1'b0	PWM channel 0 positive break state
23	pwm_ch3_npl	r/w	1'b1	PWM channel 3 negative polarity
22	pwm_ch3_ppl	r/w	1'b1	PWM channel 3 positive polarity
21	pwm_ch2_npl	r/w	1'b1	PWM channel 2 negative polarity

Bits	Name	Type	Reset	Description
20	pwm_ch2_ppl	r/w	1'b1	PWM channel 2 positive polarity
19	pwm_ch1_npl	r/w	1'b1	PWM channel 1 negative polarity
18	pwm_ch1_ppl	r/w	1'b1	PWM channel 1 positive polarity
17	pwm_ch0_npl	r/w	1'b1	PWM channel 0 negative polarity
16	pwm_ch0_ppl	r/w	1'b1	PWM channel 0 positive polarity
15	pwm_ch3_nsi	r/w	1'b1	PWM channel 3 negative set idle state
14	pwm_ch3_nen	r/w	1'b0	PWM channel 3 negative enable pwm out
13	pwm_ch3_psi	r/w	1'b0	PWM channel 3 positive set idle state
12	pwm_ch3_pen	r/w	1'b0	PWM channel 3 positive enable pwm out
11	pwm_ch2_nsi	r/w	1'b1	PWM channel 2 negative set idle state
10	pwm_ch2_nen	r/w	1'b0	PWM channel 2 negative enable pwm out
9	pwm_ch2_psi	r/w	1'b0	PWM channel 2 positive set idle state
8	pwm_ch2_pen	r/w	1'b0	PWM channel 2 positive enable pwm out
7	pwm_ch1_nsi	r/w	1'b1	PWM channel 1 negative set idle state
6	pwm_ch1_nen	r/w	1'b0	PWM channel 1 negative enable pwm out
5	pwm_ch1_psi	r/w	1'b0	PWM channel 1 positive set idle state
4	pwm_ch1_pen	r/w	1'b0	PWM channel 1 positive enable pwm out
3	pwm_ch0_nsi	r/w	1'b1	PWM channel 0 negative set idle state
2	pwm_ch0_nen	r/w	1'b0	PWM channel 0 negative enable pwm out
1	pwm_ch0_psi	r/w	1'b0	PWM channel 0 positive set idle state
0	pwm_ch0_pen	r/w	1'b0	PWM channel 0 positive enable pwm out

18.4.4 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

18.4.5 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)
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)

Bits	Name	Type	Reset	Description
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)

18.4.6 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

18.4.7 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

18.4.8 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

18.4.9 pwm_mc0_ch3_thre

Address：0x2000a45c

pwm_ch3_threH

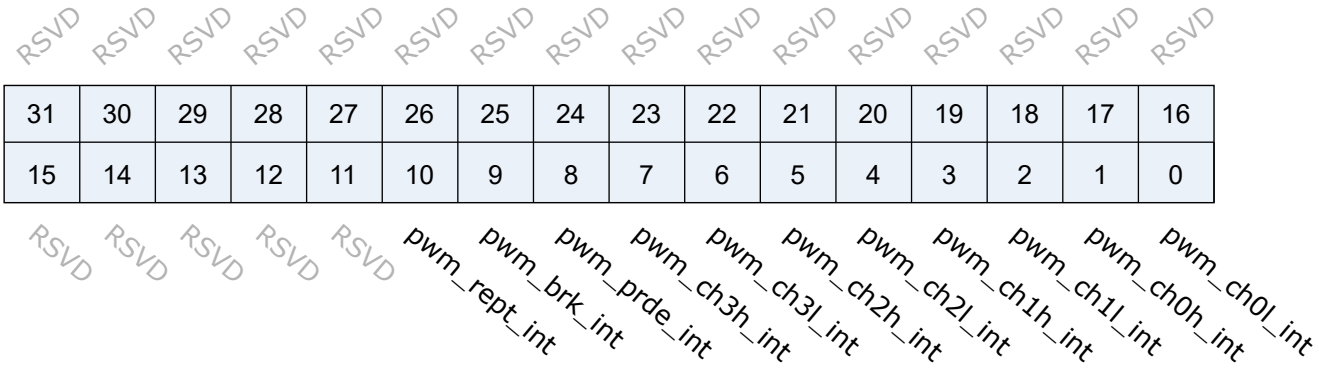
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

18.4.10 pwm_mc0_int_sts

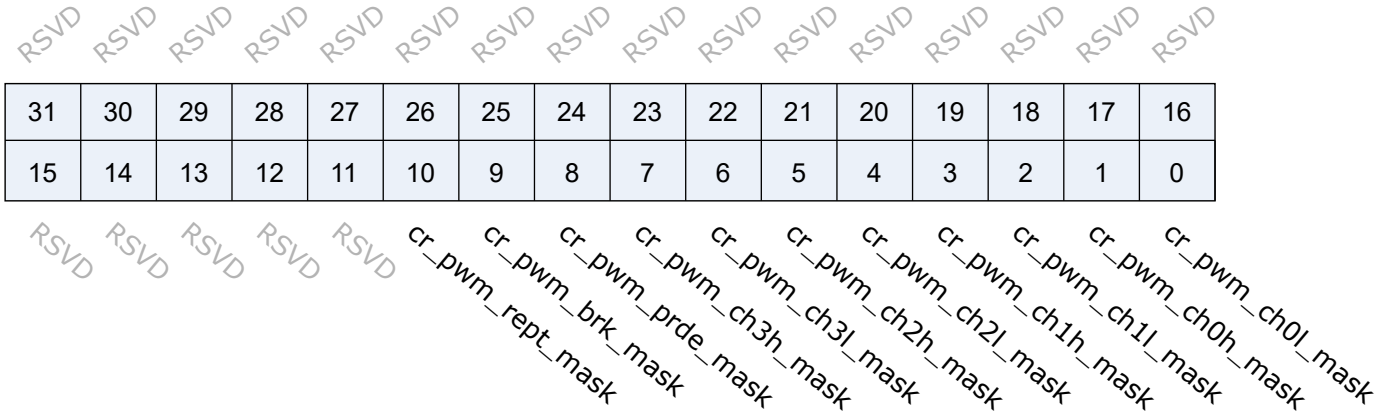
Address：0x2000a460



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	pwm_rept_int	r	1'b0	PWM repeat count interrupt status
9	pwm_brk_int	r	1'b0	PWM break interrupt status, triggered when ext_break is asserted Note: pwm_sw_break_en will NOT trigger this interrupt
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
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

18.4.11 pwm_mc0_int_mask

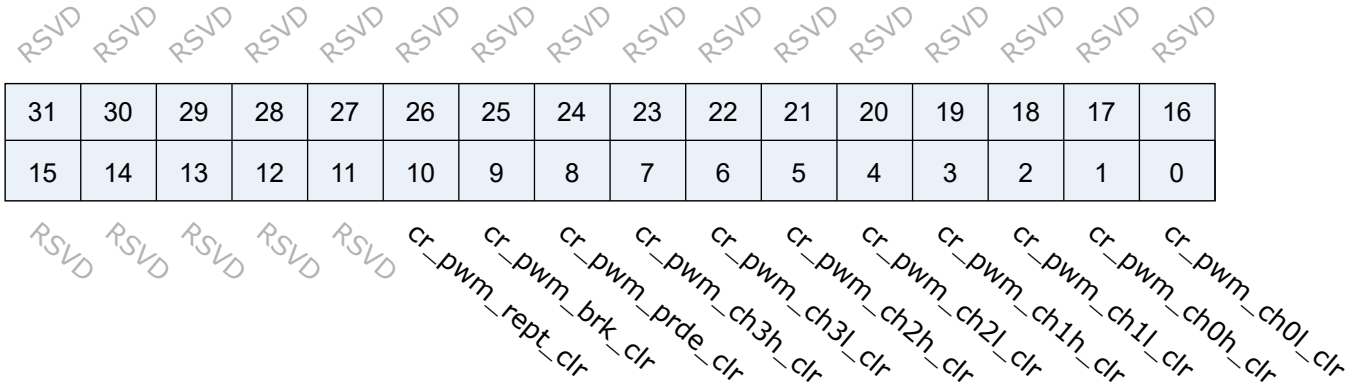
Address：0x2000a464



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_mask	r/w	1'b1	Interrupt mask of pwm_rept_int
9	cr_pwm_brk_mask	r/w	1'b1	Interrupt mask of pwm_brk_int
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

18.4.12 pwm_mc0_int_clear

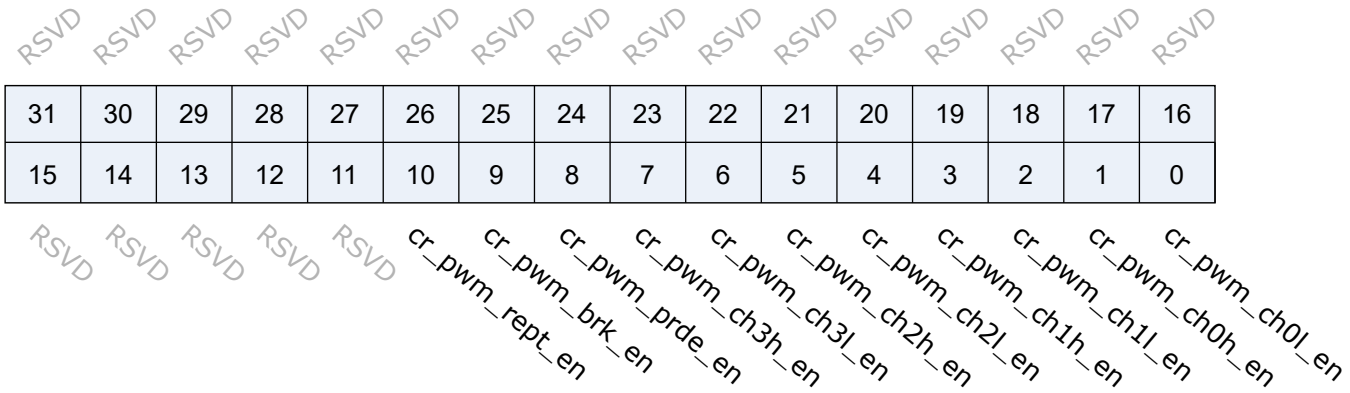
Address：0x2000a468



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_clr	w1c	1'b0	Interrupt clear of pwm_rept_int
9	cr_pwm_brk_clr	w1c	1'b0	Interrupt clear of pwm_brk_int
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

18.4.13 pwm_mc0_int_en

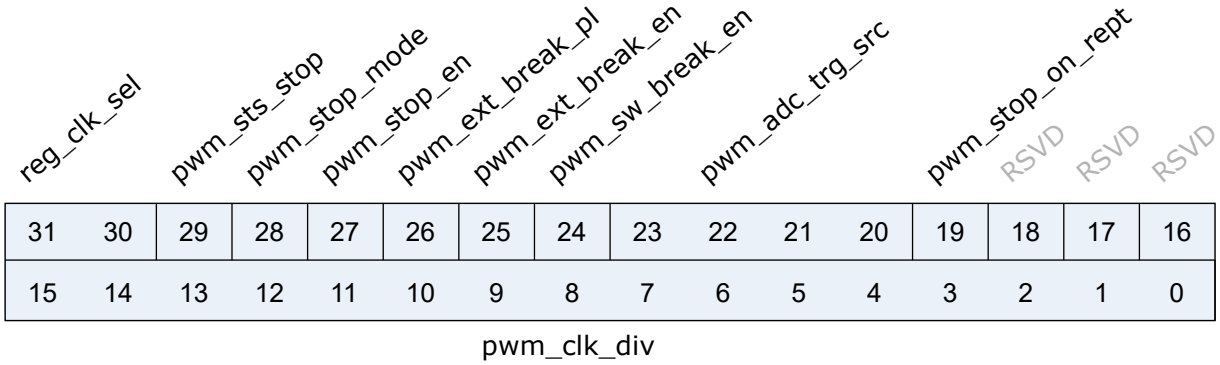
Address：0x2000a46c



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_en	r/w	1'b1	Interrupt enable of pwm_rept_int
9	cr_pwm_brk_en	r/w	1'b1	Interrupt enable of pwm_brk_int
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

18.4.14 pwm_mc1_config0

Address：0x2000a480

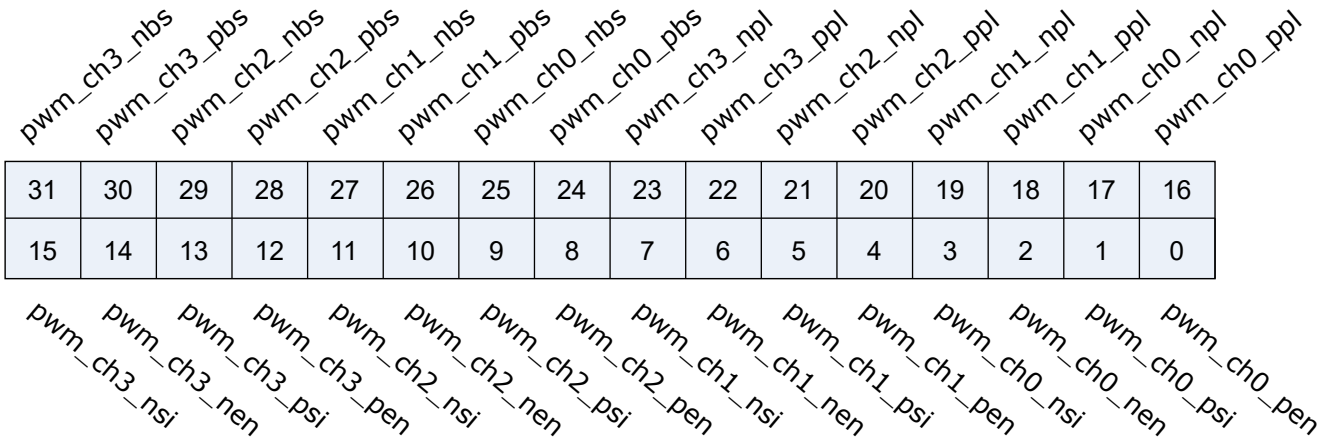


Bits	Name	Type	Reset	Description
31:30	reg_clk_sel	r/w	2'd0	PWM clock source select, 2'b00-xclk ; 2'b01-bclk ; others-f32k_clk
29	pwm_sts_stop	r	1'b0	PWM stop status
28	pwm_stop_mode	r/w	1'b1	PWM stop mode, 1'b1 - graceful ; 1'b0 - abrupt
27	pwm_stop_en	r/w	1'b0	PWM stop enable
26	pwm_ext_break_pl	r/w	1'b0	PWM external break source polarity 1'b0: Active-LOW 1'b1: Active-HIGH
25	pwm_ext_break_en	r/w	1'b0	PWM external break source enable 1'b0: Disabled, external break signal is masked 1'b1: Enabled, external break signal is effective
24	pwm_sw_break_en	r/w	1'b0	PWM break enable 1'b0: Disabled, normal operation 1'b1: 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

Bits	Name	Type	Reset	Description
19	pwm_stop_on_rept	r/w	1'b0	PWM stopped when rept_int is asserted 1'b0: Disabled, PWM keeps running when rept_int is asserted 1'b1: Enabled, PWM stops when rept_int is asserted. Clear rept_int to restore operation
18:16	RSVD			
15:0	pwm_clk_div	r/w	16'b0	PWM clock division

18.4.15 pwm_mc1_config1

Address：0x2000a484



Bits	Name	Type	Reset	Description
31	pwm_ch3_nbs	r/w	1'b0	PWM channel 3 negative break state
30	pwm_ch3_pbs	r/w	1'b0	PWM channel 3 positive break state
29	pwm_ch2_nbs	r/w	1'b0	PWM channel 2 negative break state
28	pwm_ch2_pbs	r/w	1'b0	PWM channel 2 positive break state
27	pwm_ch1_nbs	r/w	1'b0	PWM channel 1 negative break state
26	pwm_ch1_pbs	r/w	1'b0	PWM channel 1 positive break state
25	pwm_ch0_nbs	r/w	1'b0	PWM channel 0 negative break state
24	pwm_ch0_pbs	r/w	1'b0	PWM channel 0 positive break state
23	pwm_ch3_npl	r/w	1'b1	PWM channel 3 negative polarity
22	pwm_ch3_ppl	r/w	1'b1	PWM channel 3 positive polarity
21	pwm_ch2_npl	r/w	1'b1	PWM channel 2 negative polarity

Bits	Name	Type	Reset	Description
20	pwm_ch2_ppl	r/w	1'b1	PWM channel 2 positive polarity
19	pwm_ch1_npl	r/w	1'b1	PWM channel 1 negative polarity
18	pwm_ch1_ppl	r/w	1'b1	PWM channel 1 positive polarity
17	pwm_ch0_npl	r/w	1'b1	PWM channel 0 negative polarity
16	pwm_ch0_ppl	r/w	1'b1	PWM channel 0 positive polarity
15	pwm_ch3_nsi	r/w	1'b1	PWM channel 3 negative set idle state
14	pwm_ch3_nen	r/w	1'b0	PWM channel 3 negative enable pwm out
13	pwm_ch3_psi	r/w	1'b0	PWM channel 3 positive set idle state
12	pwm_ch3_pen	r/w	1'b0	PWM channel 3 positive enable pwm out
11	pwm_ch2_nsi	r/w	1'b1	PWM channel 2 negative set idle state
10	pwm_ch2_nen	r/w	1'b0	PWM channel 2 negative enable pwm out
9	pwm_ch2_psi	r/w	1'b0	PWM channel 2 positive set idle state
8	pwm_ch2_pen	r/w	1'b0	PWM channel 2 positive enable pwm out
7	pwm_ch1_nsi	r/w	1'b1	PWM channel 1 negative set idle state
6	pwm_ch1_nen	r/w	1'b0	PWM channel 1 negative enable pwm out
5	pwm_ch1_psi	r/w	1'b0	PWM channel 1 positive set idle state
4	pwm_ch1_pen	r/w	1'b0	PWM channel 1 positive enable pwm out
3	pwm_ch0_nsi	r/w	1'b1	PWM channel 0 negative set idle state
2	pwm_ch0_nen	r/w	1'b0	PWM channel 0 negative enable pwm out
1	pwm_ch0_psi	r/w	1'b0	PWM channel 0 positive set idle state
0	pwm_ch0_pen	r/w	1'b0	PWM channel 0 positive enable pwm out

18.4.16 pwm_mc1_period

Address：0x2000a488

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

18.4.17 pwm_mc1_dead_time

Address：0x2000a48c

		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)
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)

Bits	Name	Type	Reset	Description
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)

18.4.18 pwm_mc1_ch0_thre

Address：0x2000a490

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

18.4.19 pwm_mc1_ch1_thre

Address：0x2000a494

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

18.4.20 pwm_mc1_ch2_thre

Address：0x2000a498

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

18.4.21 pwm_mc1_ch3_thre

Address：0x2000a49c

pwm_ch3_threH

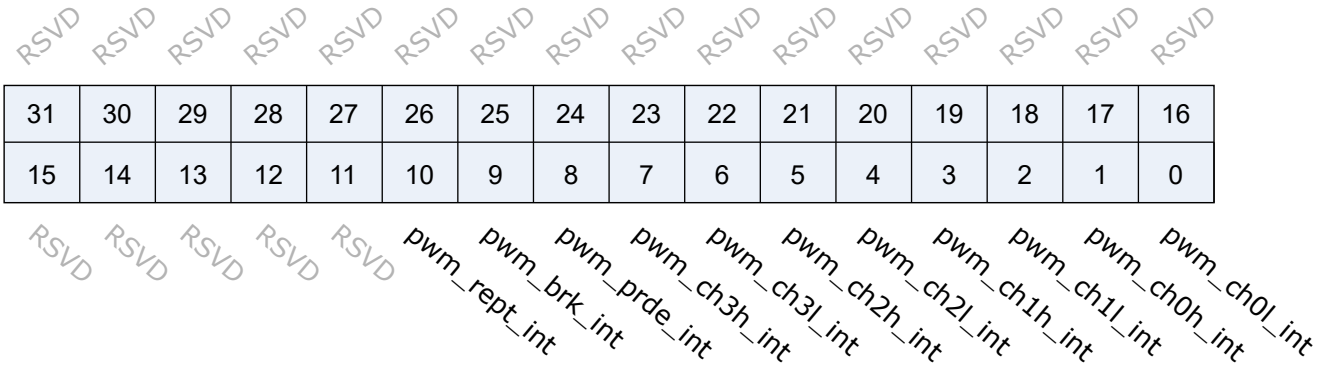
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

18.4.22 pwm_mc1_int_sts

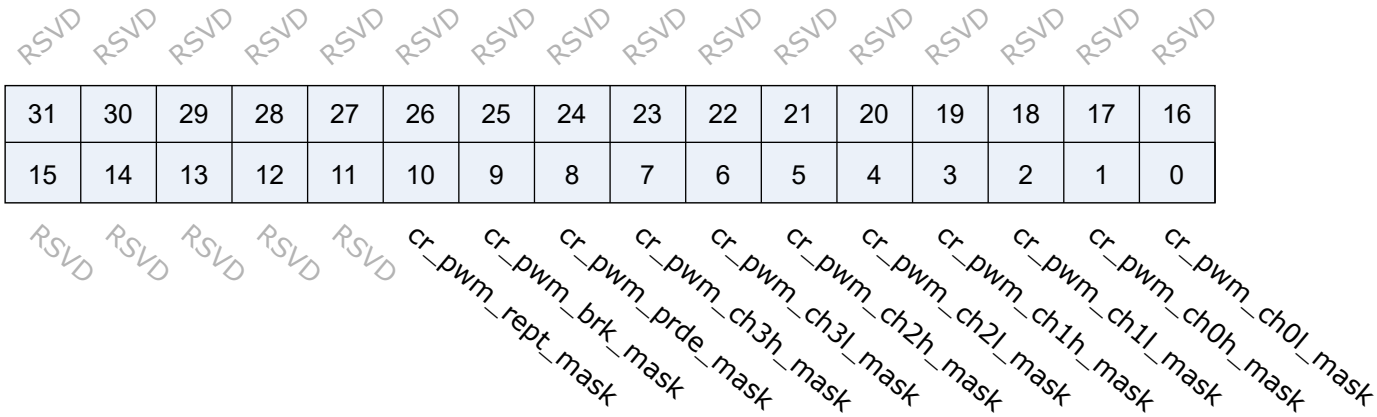
Address：0x2000a4a0



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	pwm_rept_int	r	1'b0	PWM repeat count interrupt status
9	pwm_brk_int	r	1'b0	PWM break interrupt status, triggered when ext_break is asserted Note: pwm_sw_break_en will NOT trigger this interrupt
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
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

18.4.23 pwm_mc1_int_mask

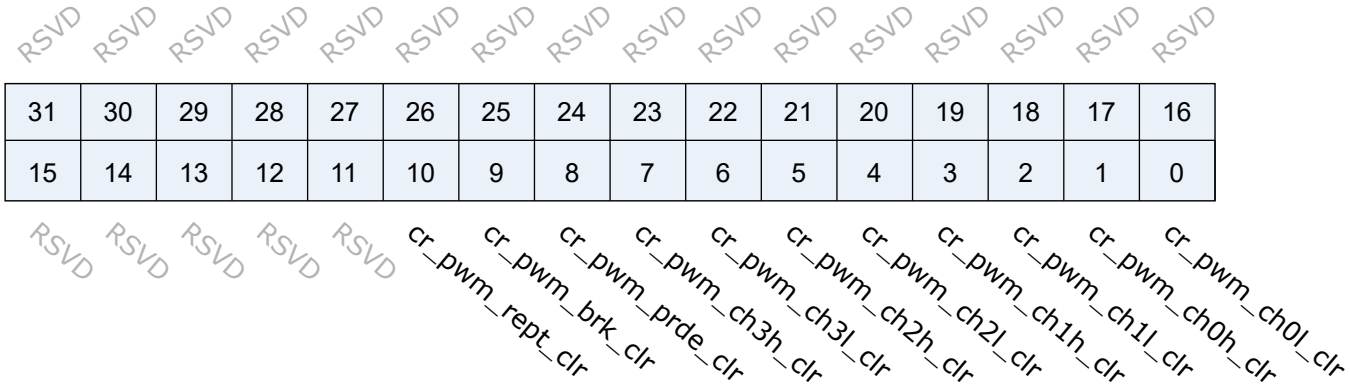
Address：0x2000a4a4



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_mask	r/w	1'b1	Interrupt mask of pwm_rept_int
9	cr_pwm_brk_mask	r/w	1'b1	Interrupt mask of pwm_brk_int
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

18.4.24 pwm_mc1_int_clear

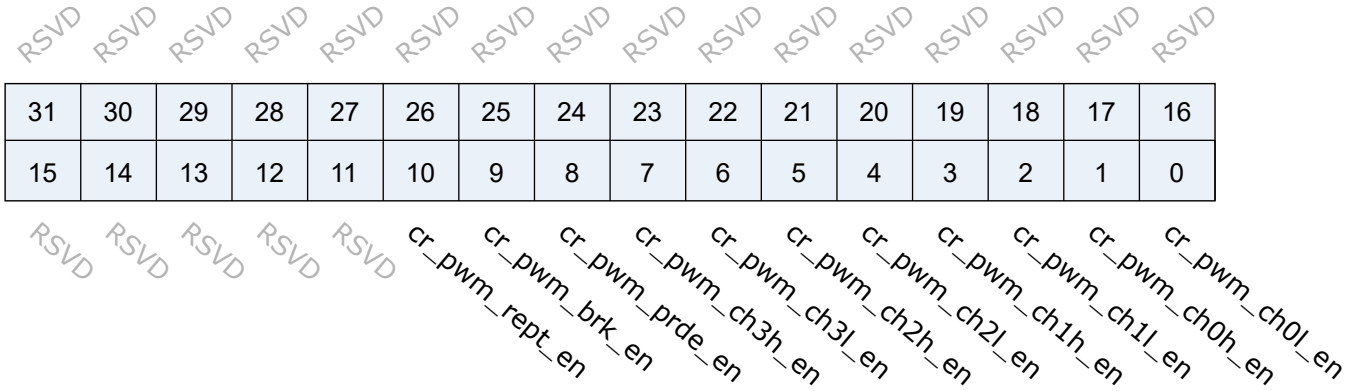
Address：0x2000a4a8



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_clr	w1c	1'b0	Interrupt clear of pwm_rept_int
9	cr_pwm_brk_clr	w1c	1'b0	Interrupt clear of pwm_brk_int
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

18.4.25 pwm_mc1_int_en

Address：0x2000a4ac



Bits	Name	Type	Reset	Description
31:11	RSVD			
10	cr_pwm_rept_en	r/w	1'b1	Interrupt enable of pwm_rept_int
9	cr_pwm_brk_en	r/w	1'b1	Interrupt enable of pwm_brk_int
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

19 TIMER

19.1 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.

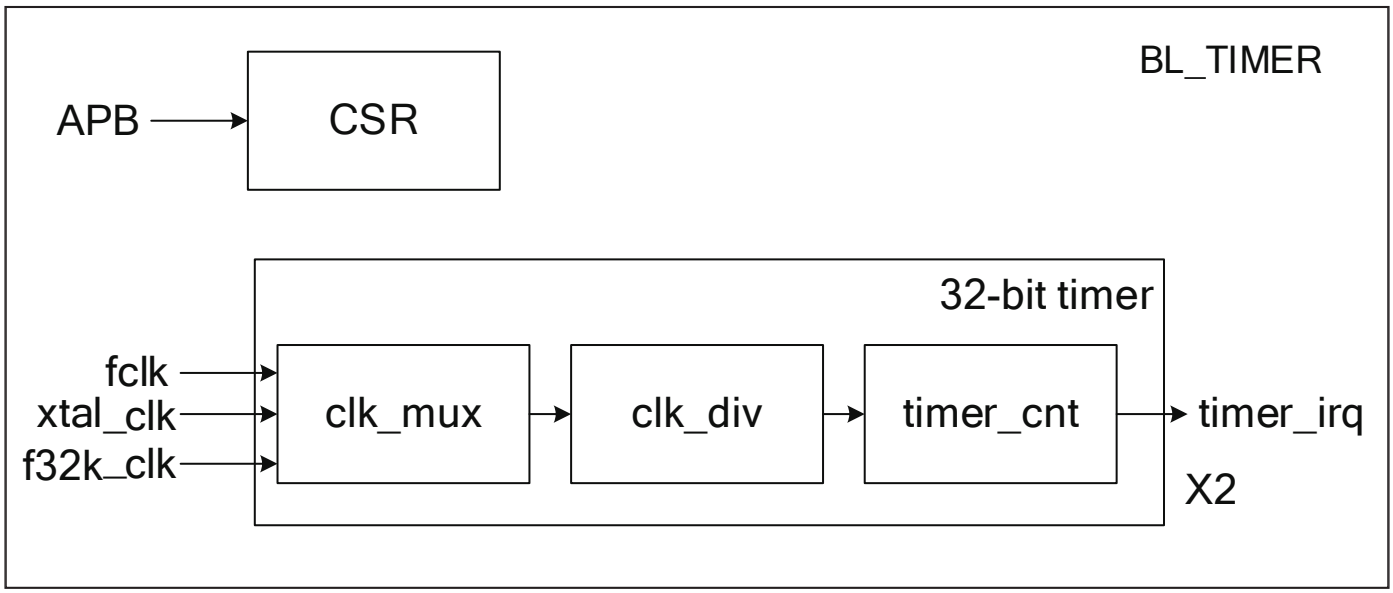


Fig. 19.1: Block diagram of timer

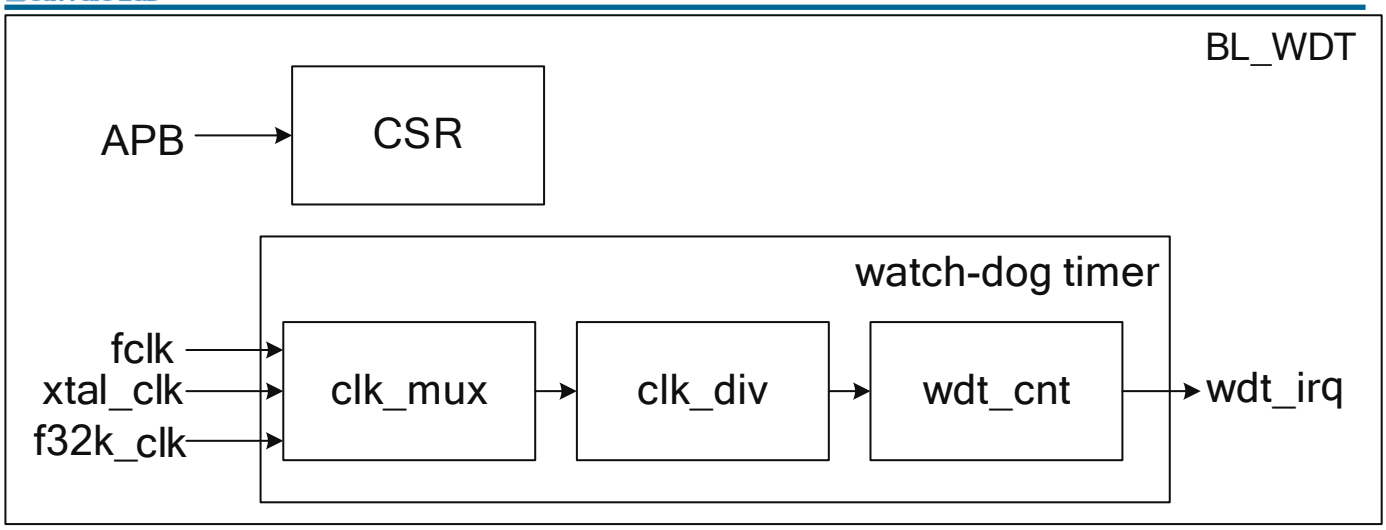


Fig. 19.2: Block diagram of watchdog timer

19.2 Features

- 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
- 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
- Supports measuring the pulse width of external GPIO

19.3 Functional Description

There are 5 types of watchdog timer clocks:

- Bclk-bus clock
- 32K-32K clock
- 1K-1K clock
- Xtal-external crystal oscillator
- GPIO-external GPIO

It is configured by cs_wdt in the register TCCR.

There are 5 types of timer clock sources:

- Bclk-bus clock
- 32K-32K clock
- 1K-1K clock(32K frequency division)
- Xtal-external crystal oscillator
- GPIO-external GPIO

It is configured by cs_2 and cs_3 in the register TCCR.

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.

It is configured by tcdr2, tcdr3, and wcdr in the register TCDR.

19.3.1 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 TMR2 comparator 0 by setting tclr2_0, that of TMR2 comparator 1 by setting tclr2_1, and that of TMR2 comparator 2 by setting tclr2_2 in the register TICR2. The tplvr2 in the register TPLVR2 sets the TMR2 preload value.

You can configure the value of TMR3 comparator 0 by setting tclr3_0, that of TMR3 comparator 1 by setting tclr3_1, and that of TMR3 comparator 2 by setting tclr3_2 in the register TICR3. The tplvr3 in the register TPLVR3 sets the TMR3 preload value.

The counter’s initial value depends on the timing mode. In the FreeRun mode, this initial value is 0, and then the counter counts up cumulatively. After reaching the maximum value, it starts counting again from 0.

In the PreLoad mode, this initial value is the value of the PreLoad register, and then counts up cumulatively. When the PreLoad condition is met, the counter’s value is set to the value of the PreLoad register, and then the counter starts counting up cumulatively again. In the counting process of the timer’s counter, once the counter’s value is consistent with one comparison value of three comparators, the comparison flag of the comparator will be set, and the corresponding comparison interrupt can be generated.

You can configure the counting mode of TMR2 by setting timer2_mode and that of TMR3 by setting timer3_mode in the register TCMR.

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:

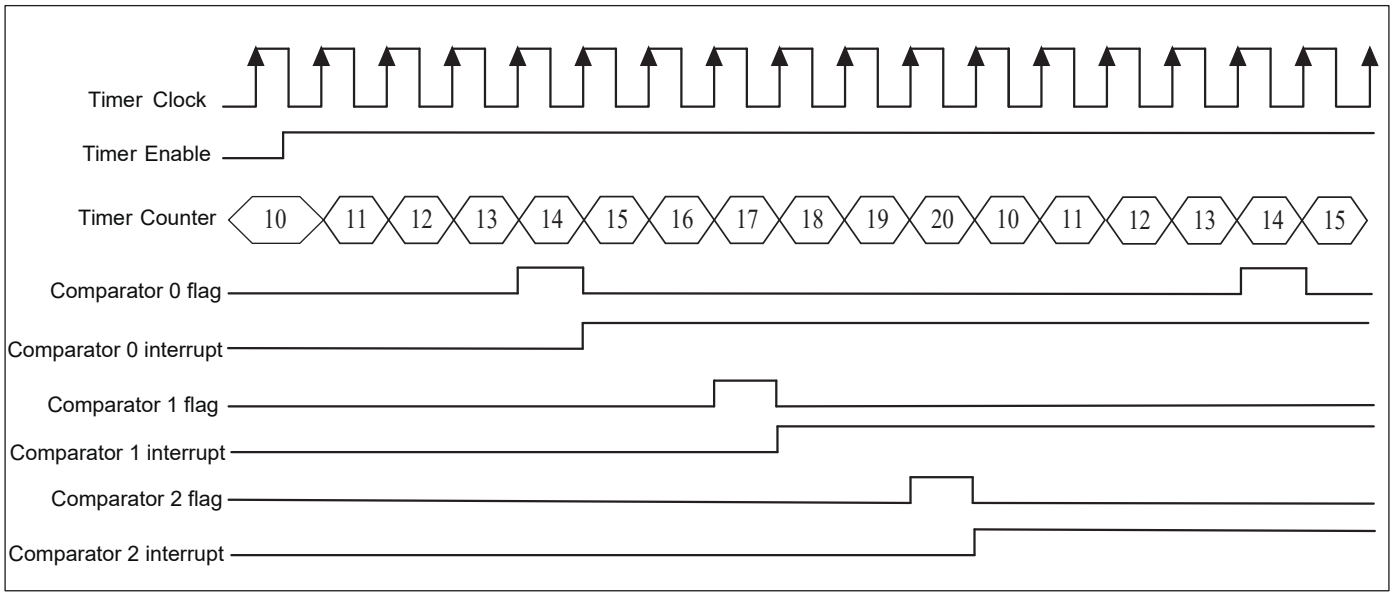


Fig. 19.3: Working sequence of timer in preLoad mode

In FreeRun mode, the working timing of the timer is basically the same as that of PreLoad, except that the counter will accumulate from 0 to the maximum value, and the mechanism of comparison flag and comparison interrupt generated during this period is the same as that of FreeRun mode.

TMR2 can use the internal clock source to calculate the pulse width of the external gpio. This function is enabled by setting timer2_gpio_en in the register GPIO. By setting the timer2_gpio_inv bit, it is judged whether the high level or low level of the external gpio is obtained. If the bit is 0, it means high level; if the bit is 1, it means Low level; in addition, the external gpio function needs to be set to the gpio_tmr_clk function. By configuring the gpio_tmr_clk_sel[13:12] bits in the register dig_clk_cfg2 in the GLB module; at the same time, you need to configure a bit in the register dig_clk_cfg2[11:8] to 0, which needs to be used in conjunction with gpio_tmr_clk_sel. details as follows:

- If gpio_tmr_clk_sel[13:12] is configured as 0, then chip_clk_out_0_en in register dig_clk_cfg2 is set to 0
- If gpio_tmr_clk_sel[13:12] is configured as 1, then chip_clk_out_1_en in register dig_clk_cfg2 is set to 0
- If gpio_tmr_clk_sel[13:12] is configured as 2, then chip_clk_out_2_en in register dig_clk_cfg2 is set to 0
- If gpio_tmr_clk_sel[13:12] is configured as 3, then chip_clk_out_3_en in register dig_clk_cfg2 is set to 0

After the configuration is complete, enable the timer. When the gpio_lat_ok in the register GPIO is set to 1, the values of the register GPIO_LAT2 and the register GPIO_LAT1 are obtained. 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. 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;

19.3.2 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:

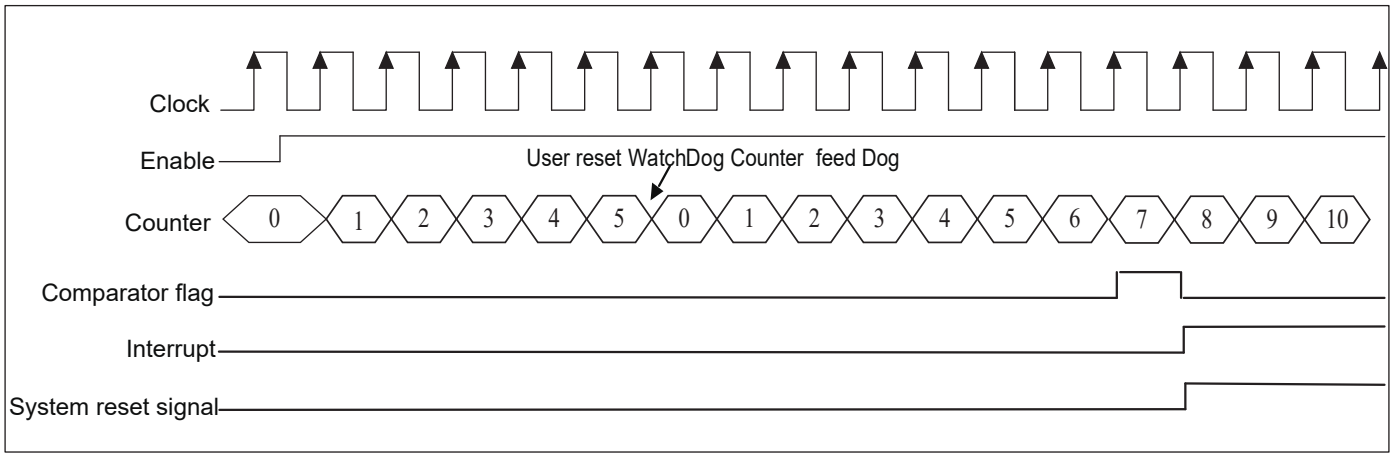


Fig. 19.4: Working sequence of watchdog

19.3.3 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.

19.3.4 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.

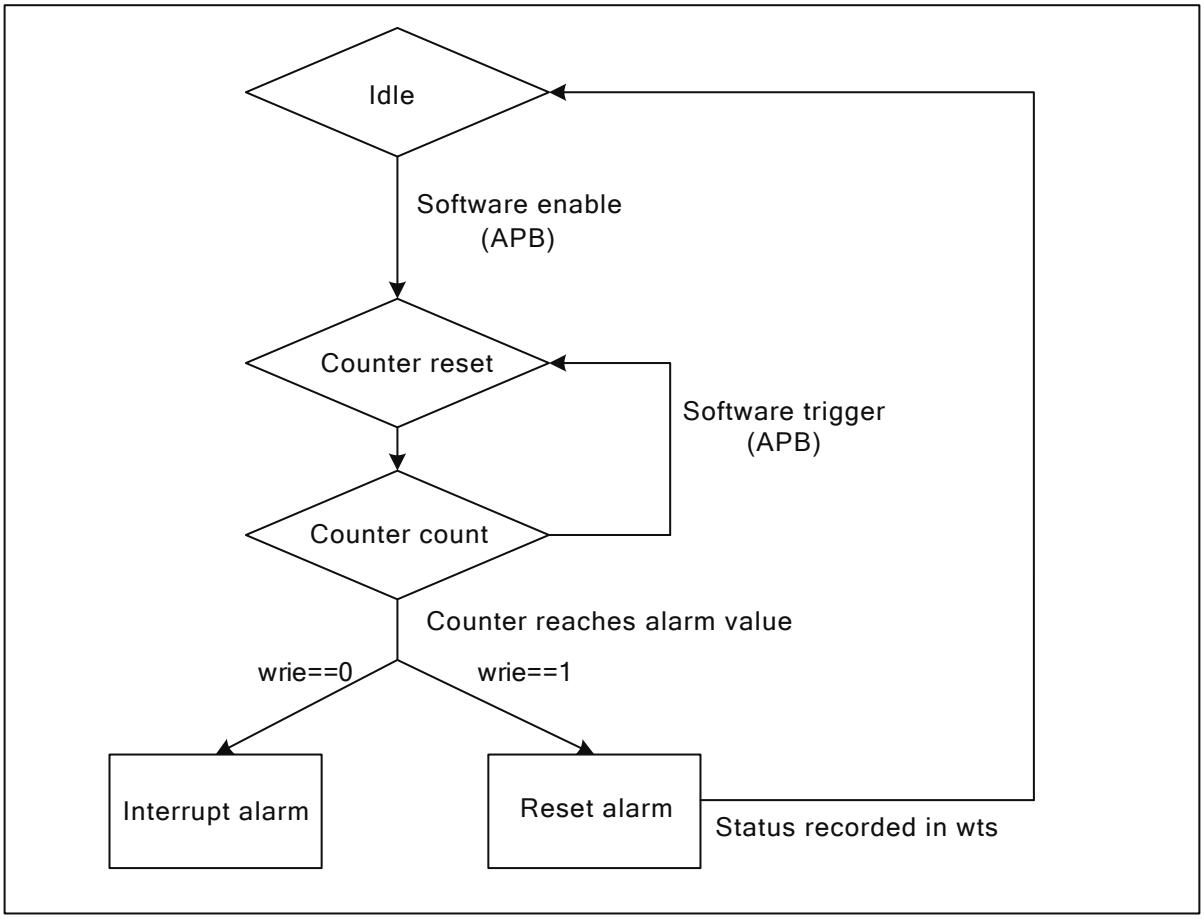


Fig. 19.5: Watchdog alarm mechanism

19.4 Register description

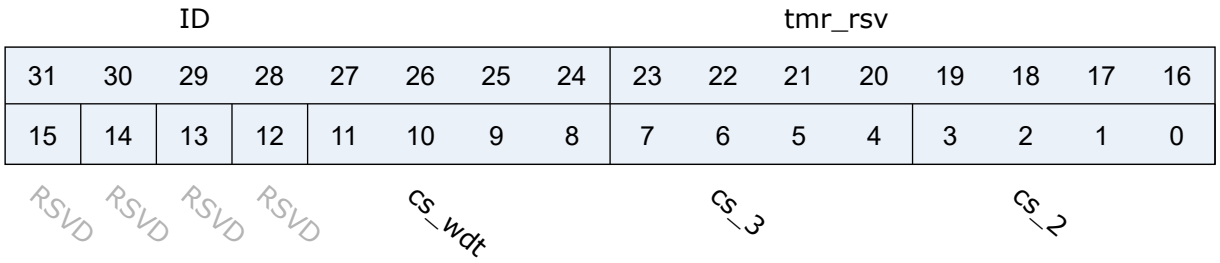
Name	Description
TCCR	Timer Clock Source
TMR2_0	Timer2 Match Value 0
TMR2_1	Timer2 Match Value 1
TMR2_2	Timer2 Match Value 2

Name	Description
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
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
TCVWR2	Timer2 Counter Latch Value
TCVWR3	Timer3 Counter Latch Value

Name	Description
TCVSYN2	Timer2 Counter Sync Value
TCVSYN3	Timer3 Counter Sync Value
TCDR	Timer Division
GPIO	GPIO Mode
GPIO_LAT1	GPIO Latch Value1
GPIO_LAT2	GPIO Latch Value2
TCDR_FORCE	Timer Division Force

19.4.1 TCCR

Address：0x2000a500



Bits	Name	Type	Reset	Description
31:24	ID	r	8'ha5	
23:16	tmr_rsv	rsvd	0	
15:12	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

19.4.2 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

19.4.3 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

19.4.4 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

19.4.5 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

19.4.6 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

19.4.7 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

19.4.8 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

19.4.9 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

19.4.10 TSR2

Address：0x2000a538

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

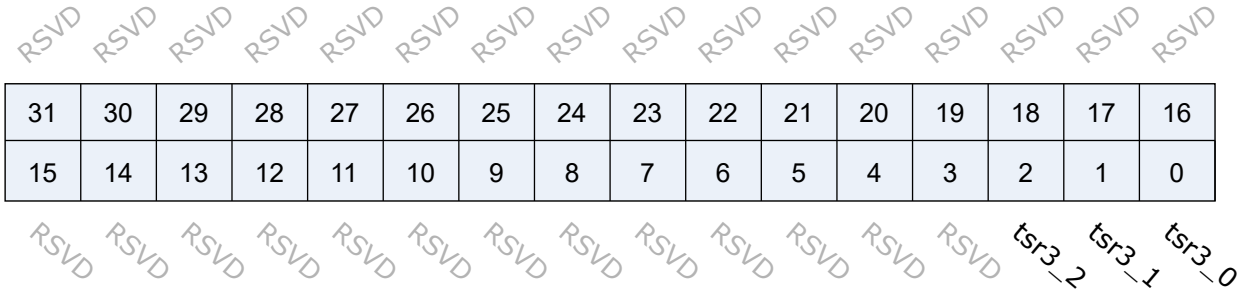
RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD tsr2_2 tsr2_1 tsr2_0

Bits	Name	Type	Reset	Description
31:3	RSVD			

Bits	Name	Type	Reset	Description
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

19.4.11 TSR3

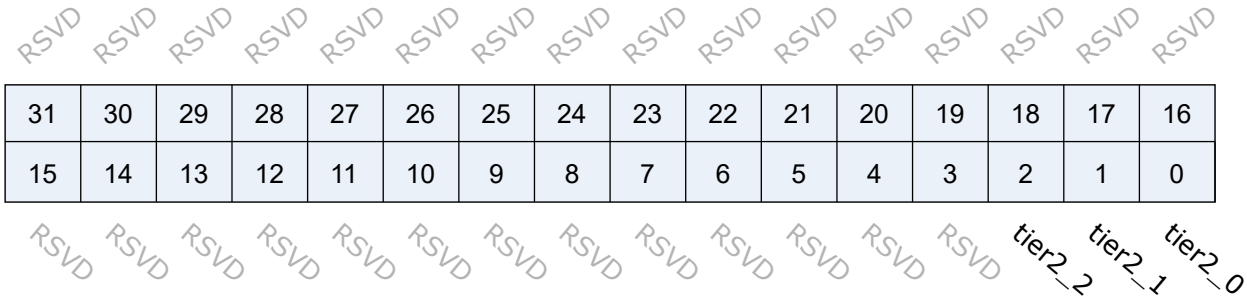
Address：0x2000a53c



Bits	Name	Type	Reset	Description
31:3	RSVD			
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

19.4.12 TIER2

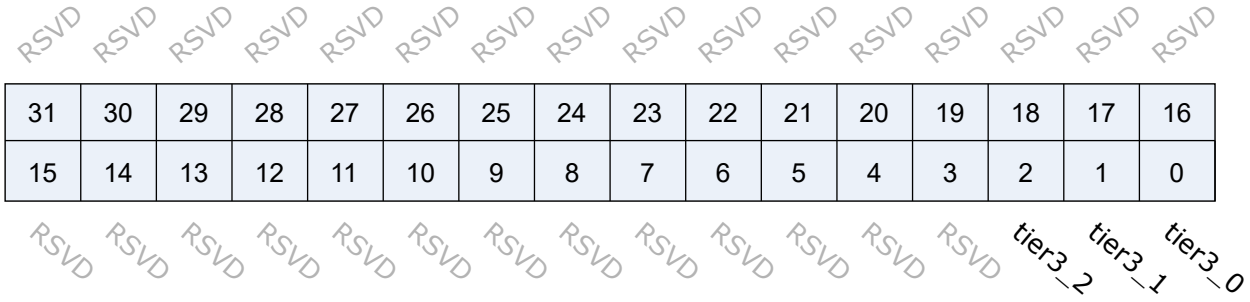
Address：0x2000a544



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

19.4.13 TIER3

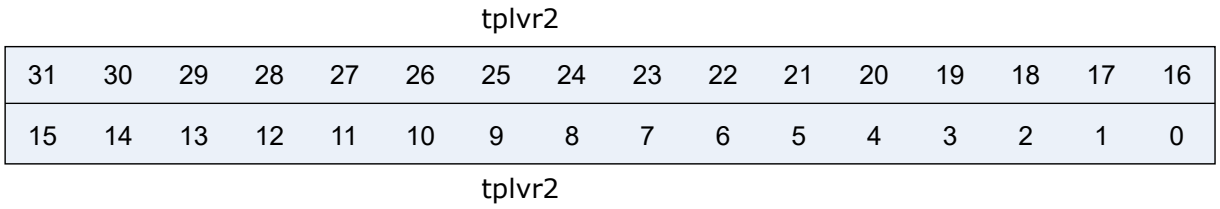
Address：0x2000a548



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

19.4.14 TPLVR2

Address：0x2000a550



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

19.4.15 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

19.4.16 TPLCR2

Address：0x2000a55c

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

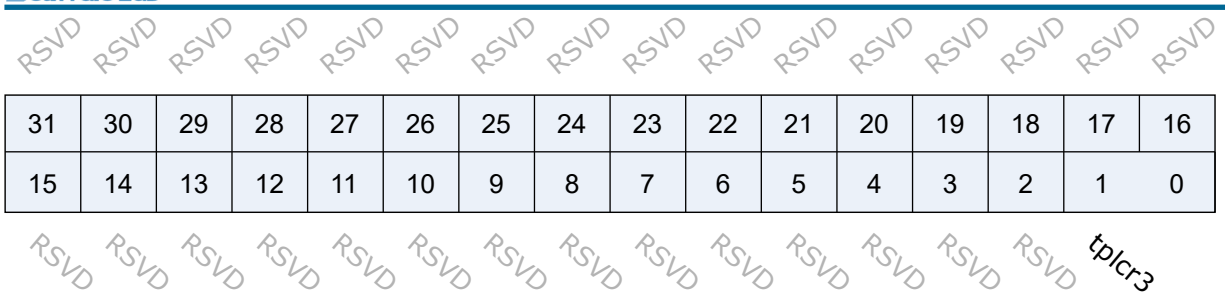
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

19.4.17 TPLCR3

Address：0x2000a560

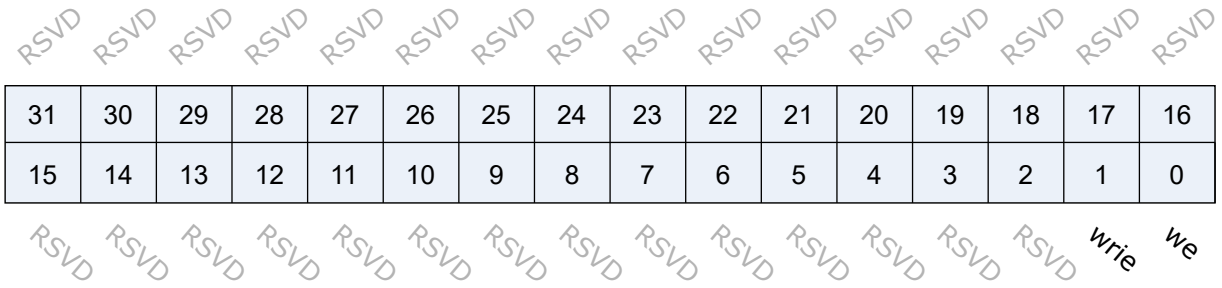


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

19.4.18 WMER

Address：0x2000a564

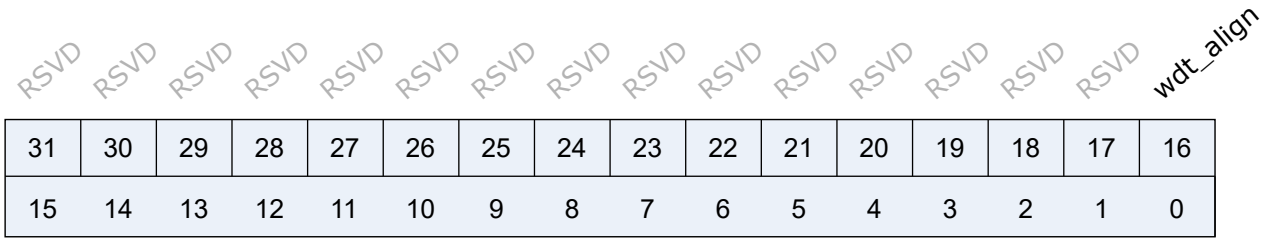


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

19.4.19 WMR

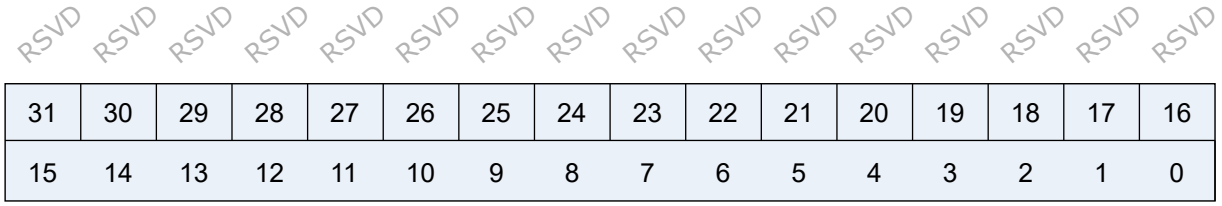
Address：0x2000a568



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

19.4.20 WVR

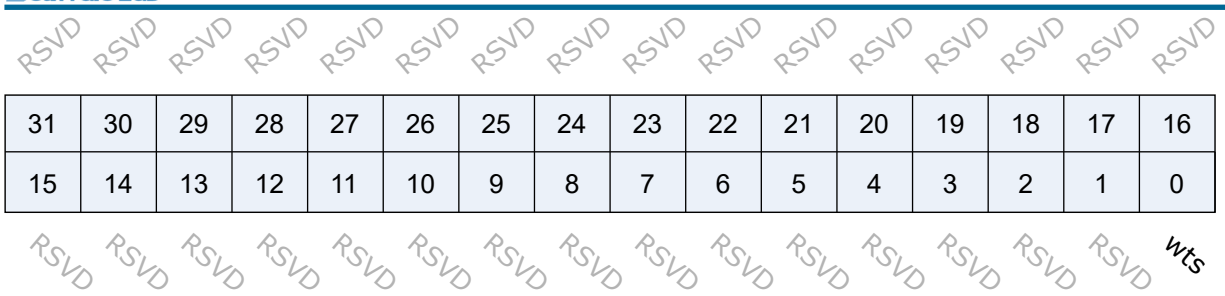
Address：0x2000a56c



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

19.4.21 WSR

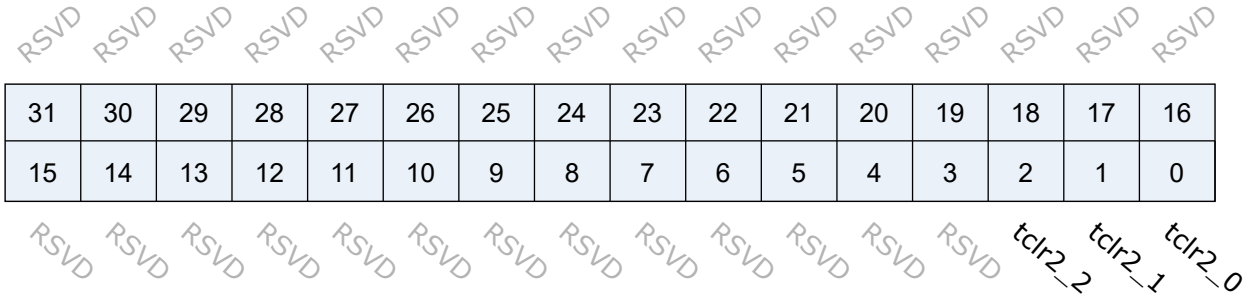
Address：0x2000a570



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

19.4.22 TICR2

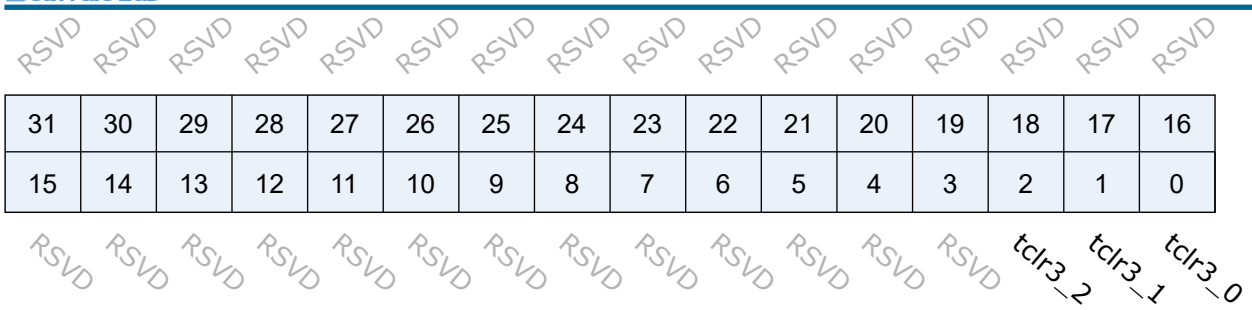
Address：0x2000a578



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

19.4.23 TICR3

Address：0x2000a57c

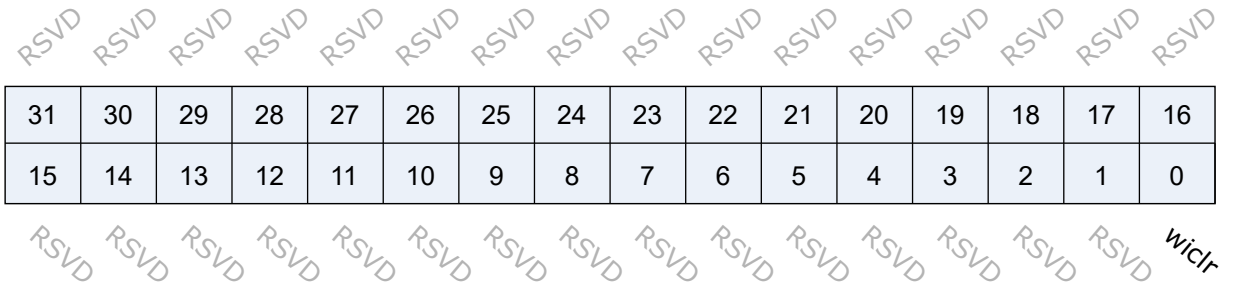


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

19.4.24 WICR

Address：0x2000a580

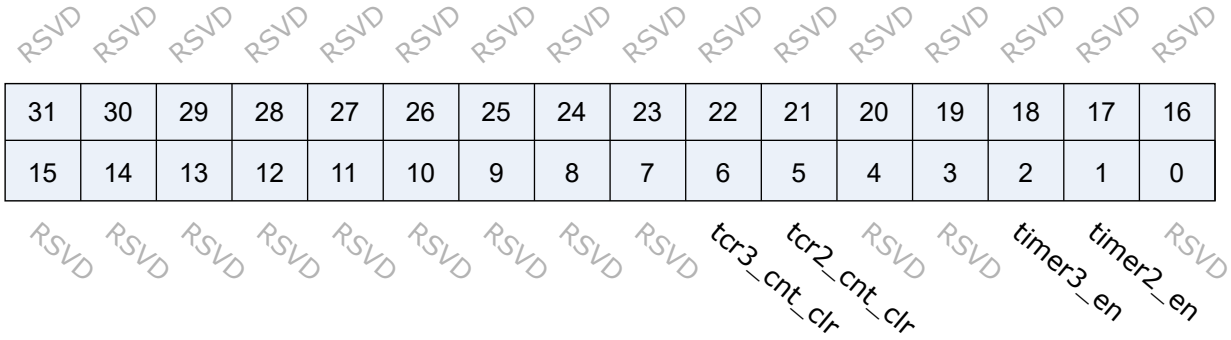


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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

19.4.25 TCER

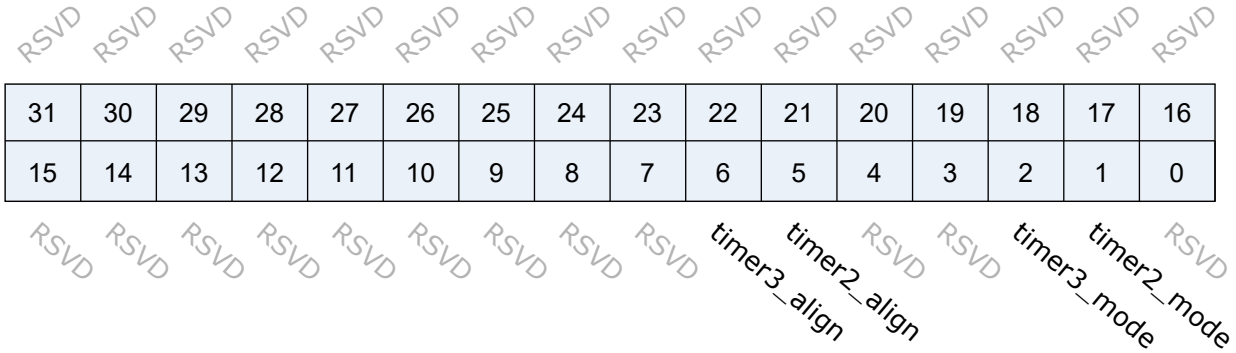
Address：0x2000a584



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			

19.4.26 TCMR

Address：0x2000a588



Bits	Name	Type	Reset	Description
31:7	RSVD			
6	timer3_align	r/w	0	Timer3 compare value update align interrupt

Bits	Name	Type	Reset	Description
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			

19.4.27 TILR2

Address：0x2000a590

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	tilr2_2	tilr2_1	tilr2_0

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
0	tilr2_0	r/w	0	0:level 1:edge

19.4.28 TILR3

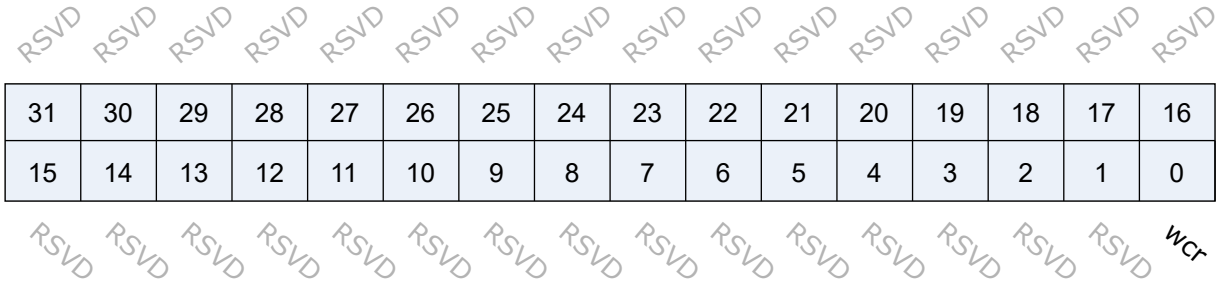
Address：0x2000a594

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	tilr3_2	tilr3_1	tilr3_0

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

19.4.29 WCR

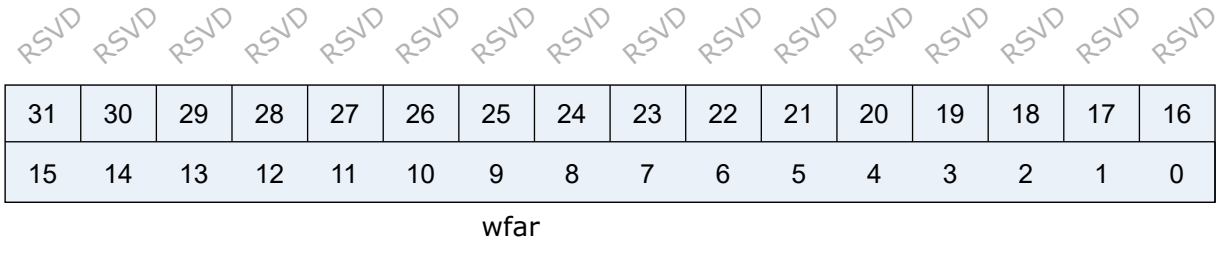
Address：0x2000a598



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

19.4.30 WFAR

Address：0x2000a59c



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

19.4.31 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

19.4.32 TCVWR2

Address：0x2000a5a8

tcr2_cnt_lat															
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_lat															

Bits	Name	Type	Reset	Description
31:0	tcr2_cnt_lat	r	0	Timer2 Counter Latch Value

19.4.33 TCVWR3

Address：0x2000a5ac

tcr3_cnt_lat															
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_lat															

Bits	Name	Type	Reset	Description
31:0	tcr3_cnt_lat	r	0	Timer3 Counter Latch Value

19.4.34 TCVSYN2

Address：0x2000a5b4

tcr2_cnt_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

tcr2_cnt_sync

Bits	Name	Type	Reset	Description
31:0	tcr2_cnt_sync	r	0	Timer2 Counter Sync Value (continue readable)

19.4.35 TCVSYN3

Address：0x2000a5b8

tcr3_cnt_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

tcr3_cnt_sync

Bits	Name	Type	Reset	Description
31:0	tcr3_cnt_sync	r	0	Timer3 Counter Sync Value (continue readable)

19.4.36 TCDR

Address：0x2000a5bc

wcdr tcdr3

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

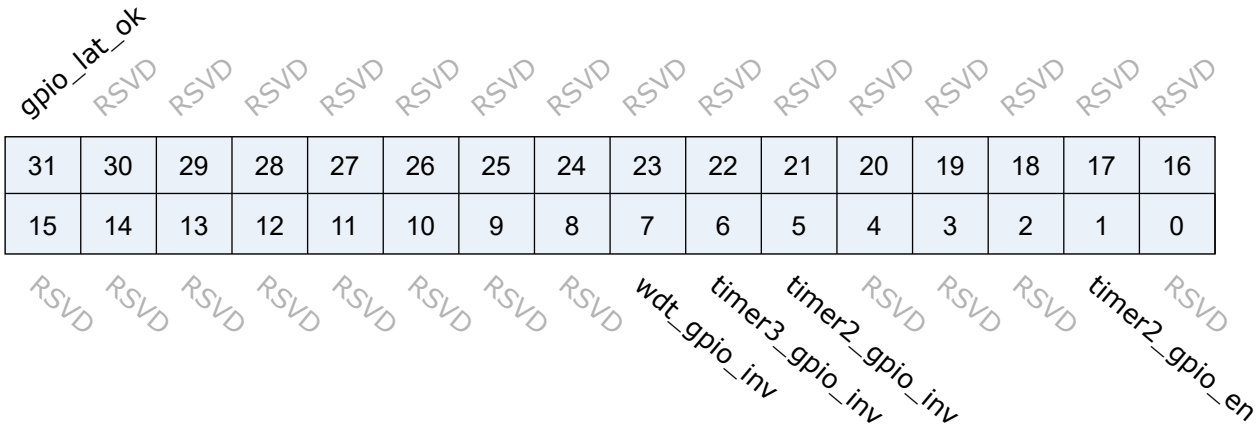
tcdr2 RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD

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

Bits	Name	Type	Reset	Description
15:8	tcdr2	r/w	0	Timer2 clock division value register
7:0	RSVD			

19.4.37 GPIO

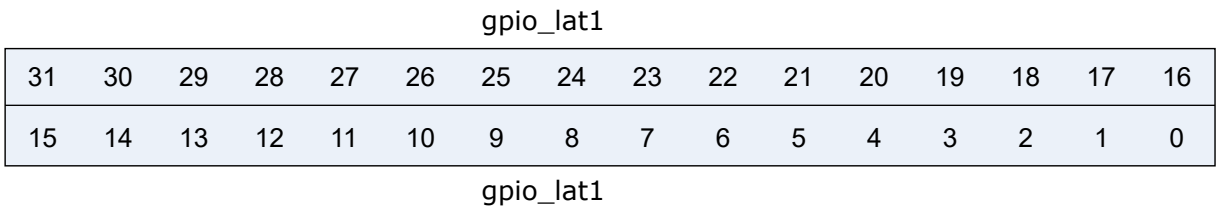
Address：0x2000a5c0



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
4:2	RSVD			
1	timer2_gpio_en	r/w	0	Timer2 gpio measure enable
0	RSVD			

19.4.38 GPIO_LAT1

Address：0x2000a5c4



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

19.4.39 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

19.4.40 TCDR_FORCE

Address：0x2000a5cc

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	wcdr_force	RSVD	tcdr3_force	tcdr2_force	RSVD

Bits	Name	Type	Reset	Description
31:5	RSVD			
4	wcdr_force	r/w	0	Force WDT clock division value to counter
3	RSVD			
2	tcdr3_force	r/w	0	Force Timer3 clock division value to counter
1	tcdr2_force	r/w	0	Force Timer2 clock division value to counter
0	RSVD			

20 I2S

20.1 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 transmits clock signals and data signals separately, so that the receiver does not need to restore the clock from the data signal, reducing the design difficulty of the receiver.

20.2 Features

- Supports master and slave modes
- Supported data formats: LeftJustified/RightJustified/Normal I2S/DSP
- Supports 8/16/24/32-bit data width
- Supports data MSB/LSB switching
- Supports DMA transfer mode
- Supports 4-channel and 6-channel modes in addition to mono/2-channel mode
- Supports playing of mono audio in the 2-channel mode and channel mode swapping
- The 2-channel recording data with less than 16 bits can be merged into FIFO data with less than 32 bits
- Supports 16/32/48/64-bit frame size
- Supports dynamic mute switch function
- TX FIFO has a width of 32 bits and a depth of 16
- RX FIFO has a width of 32 bits and a depth of 16

20.3 Functional Description

Table 20.1: I2S pin list

Name	Type	Description
I2Sx_DI	Input	Serial data input
I2Sx_DO	Output	Serial data output
I2Sx_BCLK	Input/output	Synchronous transmission clock: output (master)/input (slave)
I2Sx_FS	Input/output	Data start/end signal: output (master)/input (slave)

20.4 Functional Description

20.4.1 Data Formats

I2S supports Normal I2S/LeftJustified/RightJustified modes. Normal I2S is a special case of the LeftJustified mode. The mode is set by I2S_CONFIG[17:16].

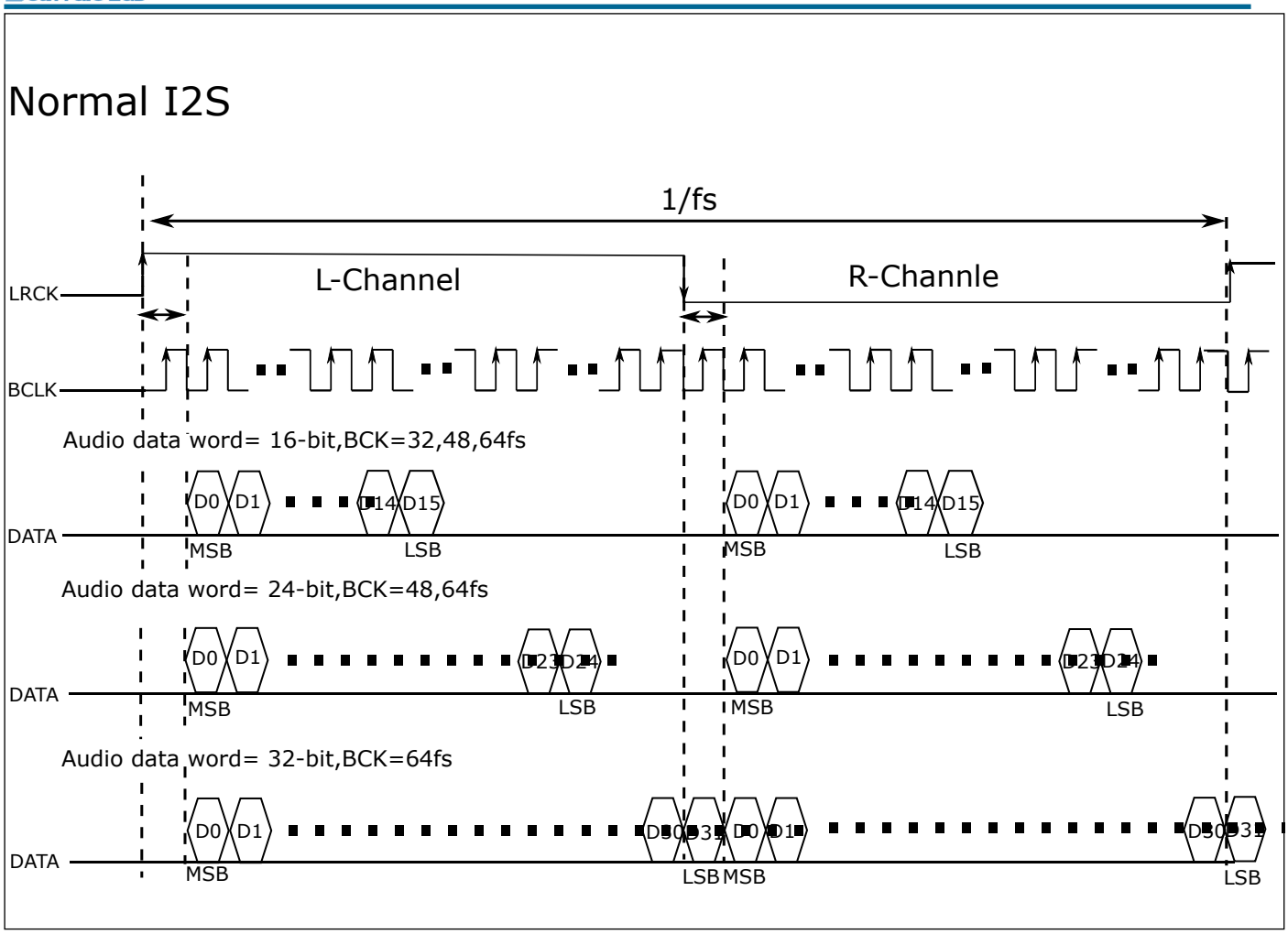


Fig. 20.1: Normal I2S

OFFET is configured by I2S_CONFIG[25:20]. The only difference between LeftJustified and Normal I2S modes is the different configuration of I2S_CONFIG[25:20].

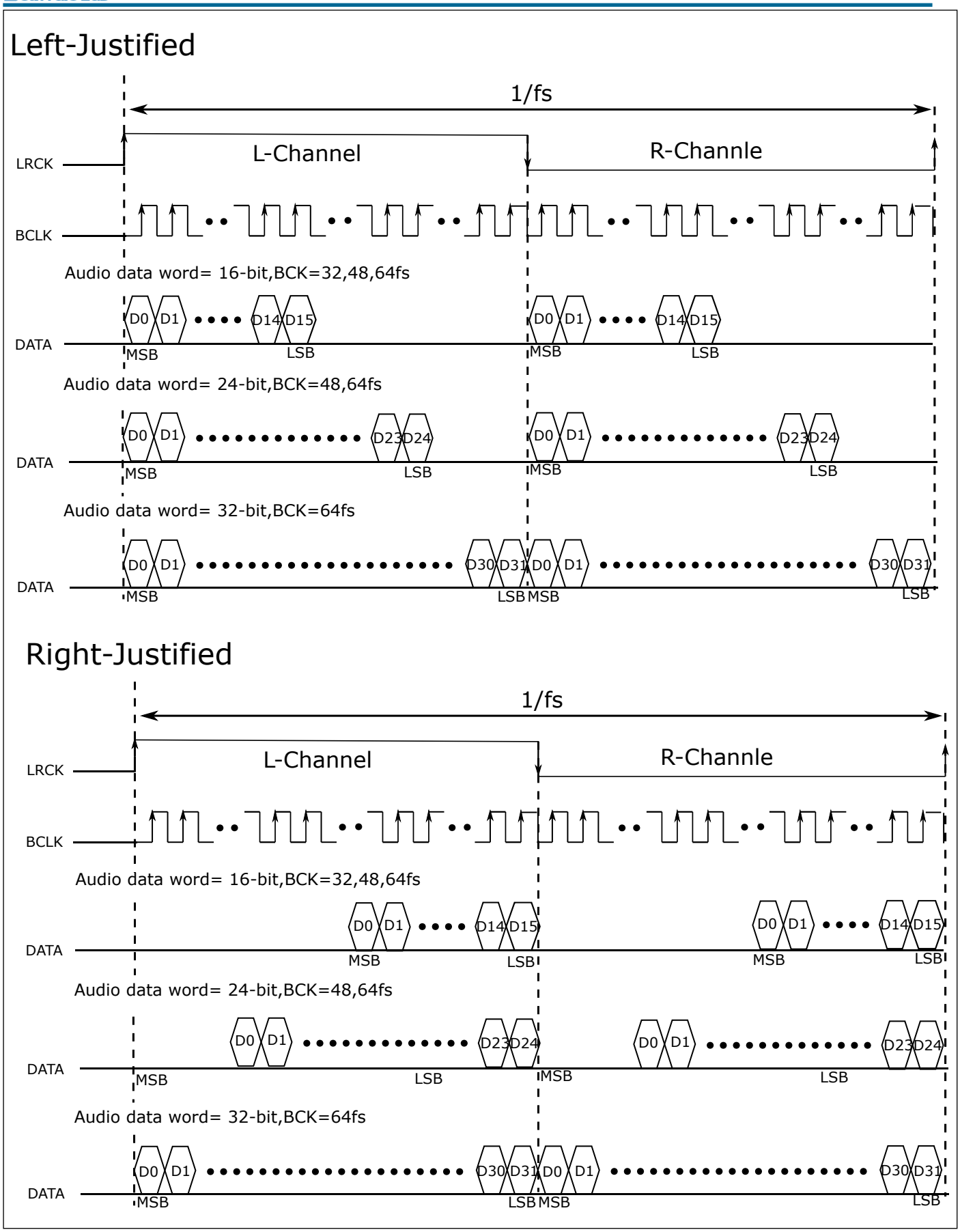


Fig. 20.2: I2S LeftJustified/RightJustified

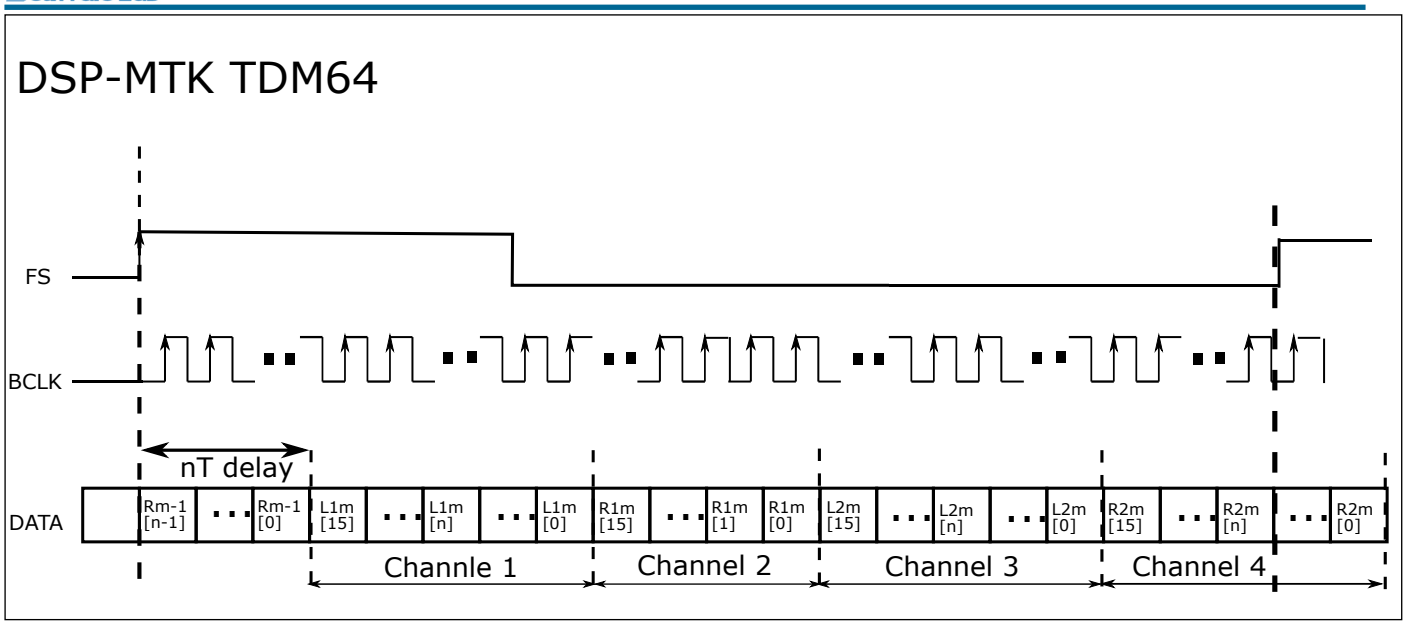


Fig. 20.3: I2S TDM64 Mode—6-Channel Recording

The width of a single pulse is controlled by I2S_CONFIG[6]. When it is set to 0, the width of high-level pulse of FS signal line is the width of data size, and when it is set to 1, that width is 1. Generally, this register is set to 1 in the multi-channel TDM64 mode.

20.4.2 Basic architecture

20.4.3 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:

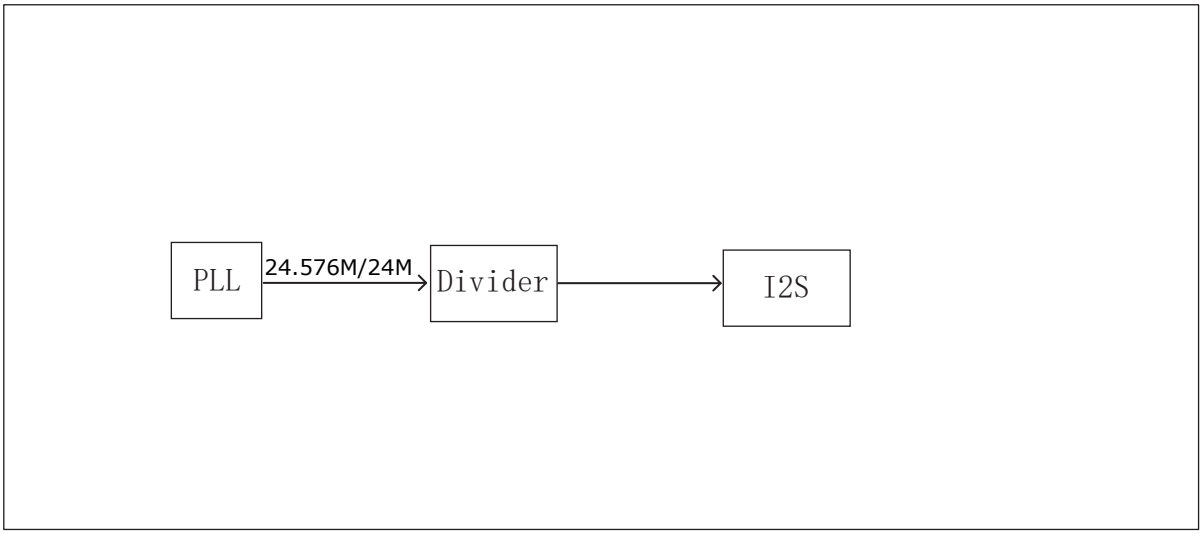


Fig. 20.4: I2S clock

20.4.4 I2S Interrupt

I2S supports the following interrupt control modes:

- TX FIFO request interrupt
- RX FIFO request interrupt
- Overrun/undderun error 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.

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.

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.

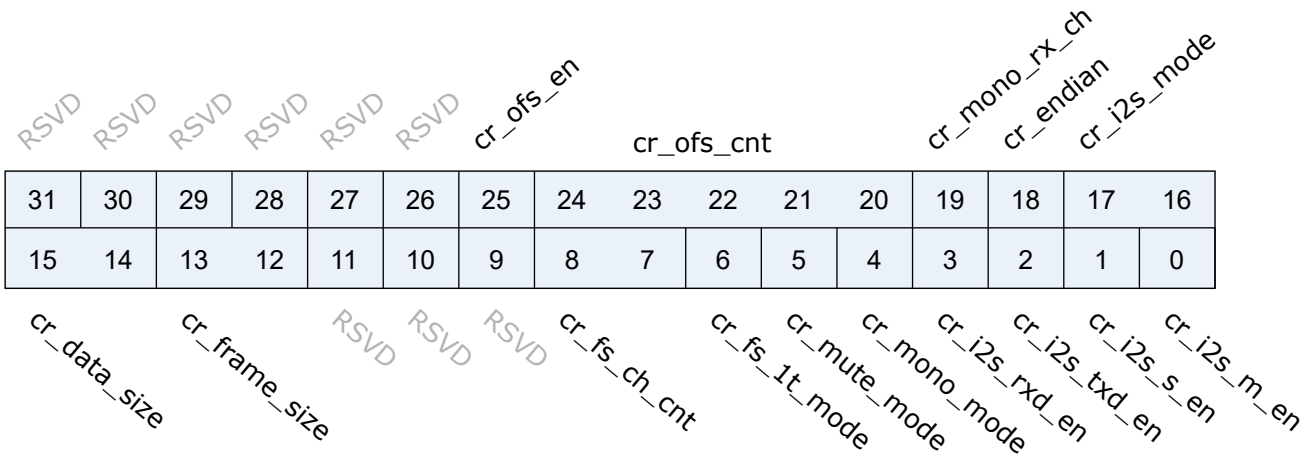
All the interrupt enable bits and interrupt flag bits of I2S are in the I2S_INT_STS register.

20.5 Register description

Name	Description
i2s_config	
i2s_int_sts	
i2s_bclk_config	
i2s_fifo_config_0	
i2s_fifo_config_1	
i2s_fifo_wdata	
i2s_fifo_rdata	
i2s_io_config	

20.5.1 i2s_config

Address：0x2000ab00

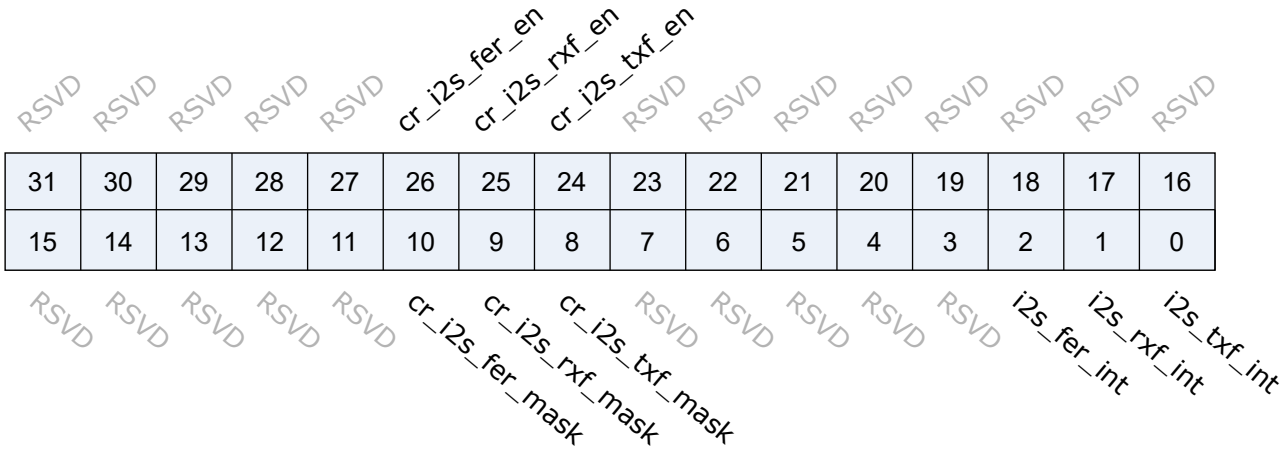


Bits	Name	Type	Reset	Description
31:26	RSVD			
25	cr_ofs_en	r/w	1'b0	Offset enable 1'b0: Disabled, 1'b1: Enabled
24:20	cr_ofs_cnt	r/w	5'd0	Offset cycle count (unit: cycle of I2S BCLK) 5'd0: 1 cycle 5'd1: 2 cycles ...
19	cr_mono_rx_ch	r/w	1'b0	RX mono mode channel select signal 1'b0: L-channel 1'b1: R-channel
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			

Bits	Name	Type	Reset	Description
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

20.5.2 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

20.5.3 i2s_bclk_config

Address：0x2000ab10

RSVD RSVD RSVD RSVD cr_bclk_div_h

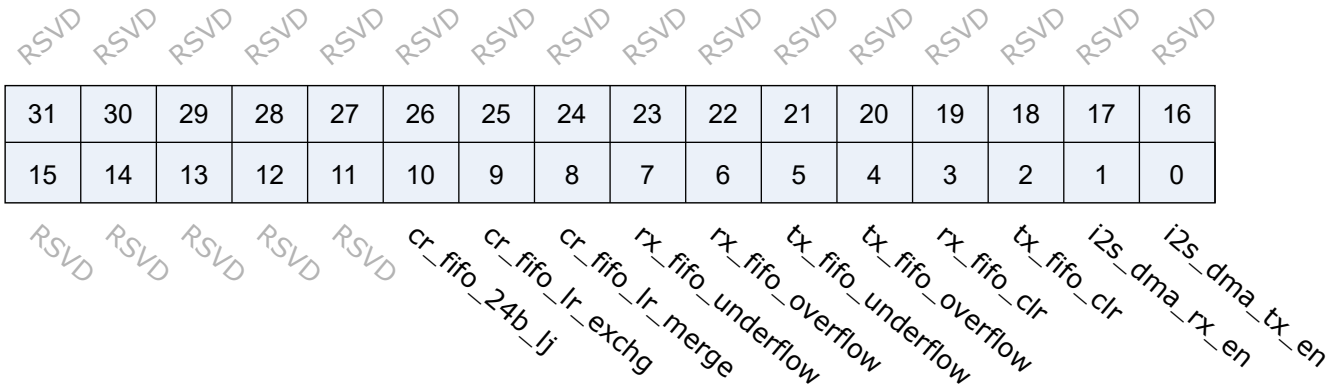
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD cr_bclk_div_l

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)

20.5.4 i2s_fifo_config_0

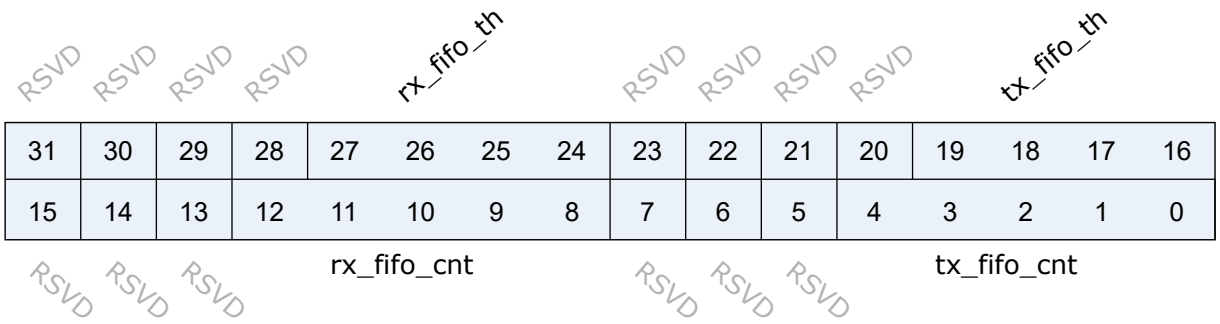
Address：0x2000ab80



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)
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

20.5.5 i2s_fifo_config_1

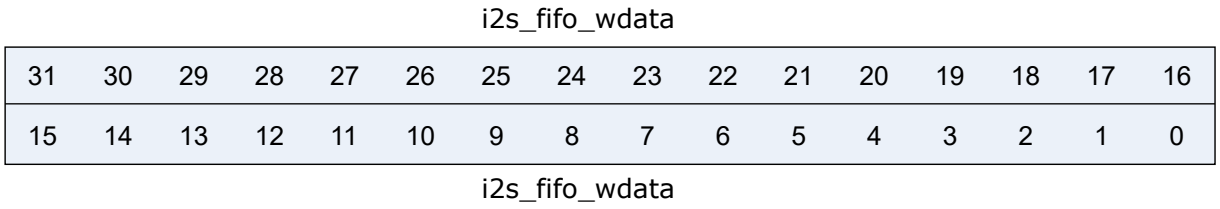
Address：0x2000ab84



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 rx_-fifo_cnt is less than this value
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

20.5.6 i2s_fifo_wdata

Address：0x2000ab88



Bits	Name	Type	Reset	Description
31:0	i2s_fifo_wdata	w	x	

20.5.7 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	

20.5.8 i2s_io_config

Address：0x2000abfc

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_deg_en cr_deg_cnt cr_i2s_bclk_inv cr_i2s_fs_inv cr_i2s_rxd_inv cr_i2s_txd_inv

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

Bits	Name	Type	Reset	Description
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

21 PDM

21.1 Overview

A PDM audio processing module is built in the chip, supporting recording via the PDM interface microphone.

21.2 Features

- Three 20-bit ADCs are integrated to support three digital PDM inputs.
  - Sampling rate: 8k~96k
  - Signal to noise ratio (AW): 97 dB @ 0 dB gain
  - Harmonic distortion + noise: -87dB @ 0dB gain
  - Analog pre-amplifier gain: 0dB, 6~42 dB, 3dB per gear
- Adjustable high-pass filter and independent digital volume control for ADC channel
- Supports digital PDM interface, with input GPIO multiplexed
- Independent digital volume control
- TX/RX FIFO of 32-bit width
- Supports DMA transfer mode

21.3 Functional Description

The block diagram of PDM module is shown as follows.

PDM has three PDM digital interface demodulators. It passes through the decimation and HPF filters and then goes to the volume control module. Users can control mute/unmute, volume, and audio fade-in/fade-out effect through the volume control module. The recorded data is stored in the FIFO with a depth of 32, and the storage format in FIFO is controlled by the register RX_FIFO_CTRL[25:24]. PDM_RX_FIFO_CTRL[5] configures the resolution of sampling.

PDM only supports PDM audio digital interface input.

21.3.1 PDM Interrupt

PDM 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.

21.3.2 FIFO Format Control

PDM_RX_FIFO_CTRL can control the format of the audio data to be stored in FIFO.

The user sets the resolution of audio by configuring PDM_RX_FIFO_CTRL[5].

When 16-bit resolution is selected, the FIFO controller supports the following four data storage formats, which are determined by FIFO_CTRL[25:24].

- Mode 0:

  DATA[31:0] = {FIFO[19:14],16’h0}

- Mode 1:

  DATA[31:0] = {8{FIFO[19]},FIFO[19:4],8’h0}

- Mode 2:

  DATA[31:0] = {12{FIFO[19]},FIFO[19:4],4’h0}

- Mode 3:

  DATA[31:0] = {16{FIFO[19]},FIFO[19:4]}

During recording or playing, the conversion result of 32-bit data or the digital audio source to be demodulated is on the left, denoted as DATA[31:0]. The format of data to be stored in FIFO is shown on the right. When the actual resolution of the recording is 20-bit, if the 16-bit resolution is selected, the data with 20-bit resolution must be properly cut. Thus, the high 16 bits of the data with 20-bit resolution are selected as the final result FIFO[19:14] and stored in the position of the high 16 bits, while the low 16 bits are filled with 0. Modes 1, 2, and 3 are expressed in the same

way as Mode 0. The 8{FIFO[19]} symbol indicates that the high 8 bits are padded with the value of bit[19].

Therefore, when a 16-bit resolution is selected, PDM provides four modes to store the digital results of conversion/output.

In the case of 20-bit resolution, the storage formats in the four modes are as follows:

- Mode 0:

  DATA[31:0] = {FIFO[19:0],12’h0}

- Mode 1:

  DATA[31:0] = {8{FIFO[19]},FIFO[19:0],4’h0}

- Mode 2:

  DATA[31:0] = {12{FIFO[19]},FIFO[19:0]}

- Mode 3:

  DATA[31:0] = {16{FIFO[19]},FIFO[19:4]}

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

21.3.3 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 count threshold (8/16/32) for initiating DMA request is selected by configuring FIFO_CTRL[15:14], or can be determined by FIFO_CTRL[22:16].

When the count value is greater than the set threshold, and the FIFO of the channel corresponding to PDM_RX_-FIFO_CTRL[12:8] is enabled, a DMA transfer is initiated.

When TX FIFO is started, if there is no valid data in TX FIFO, the tx underrun error will be triggered. Therefore, the

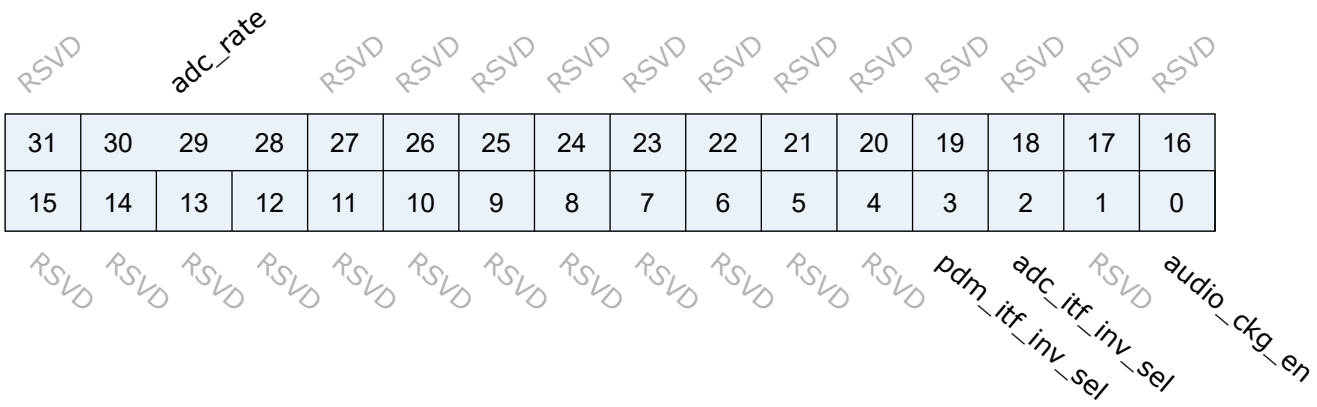
software configuration sequence must be followed.

21.4 Register description

Name	Description
audpdm_top	
audpdm_itf	
pdm_adc_0	
pdm_adc_1	
pdm_dac_0	
pdm_pdm_0	
pdm_rsvd0	
pdm_dbg_0	
pdm_dbg_1	
pdm_dbg_2	
pdm_dbg_3	
pdm_dbg_4	
pdm_adc_s0	
pdm_adc_s1	
pdm_adc_s2	
pdm_rx_fifo_ctrl	
pdm_rx_fifo_status	
pdm_rx_fifo_data	

21.4.1 audpdm_top

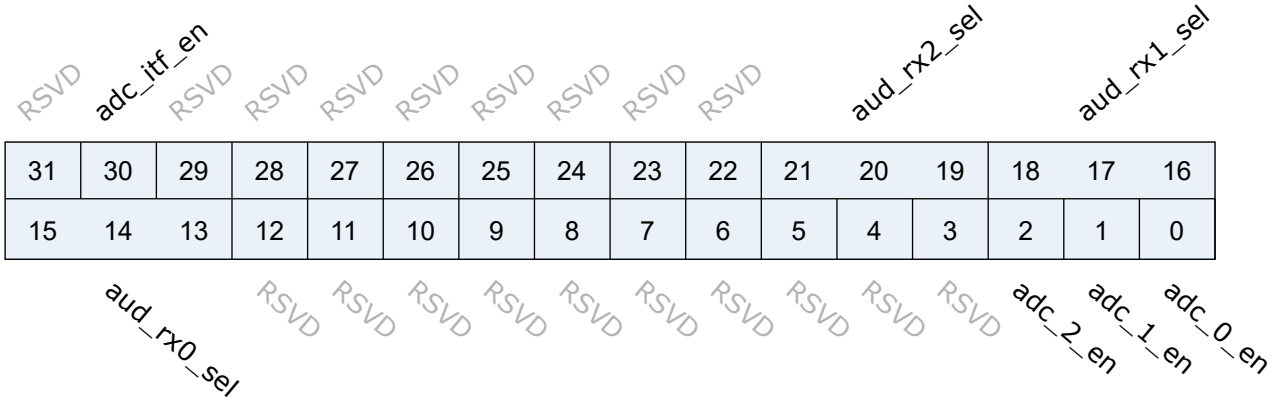
Address：0x2000ac00



Bits	Name	Type	Reset	Description
31	RSVD			
30:28	adc_rate	r/w	3'd1	adc fs 0:8kHz,1:16kHz, 2:24kHz(22.05k), 3:32kHz, 4:48kHz(44.1k), 5:96kHz, 6:reserved, 7:manual
27	RSVD			
26:24	dac_rate	r/w	3'd4	dac fs 0:8kHz,1:16kHz, 2:24kHz(22.05k), 3:32kHz, 4:48kHz(44.1k), 5:96kHz, 6:192kHz, 7:manual
23:4	RSVD			
3	pdm_itf_inv_sel	r/w	1'd0	1:invert clk_pdm_inf
2	adc_itf_inv_sel	r/w	1'd0	1:invert clk_adc_itf
1	dac_itf_inv_sel	r/w	1'd1	1:invert clk_dac_itf
0	audio_ckg_en	r/w	1'd0	1:enable audio clock generator

21.4.2 audpdm_itf

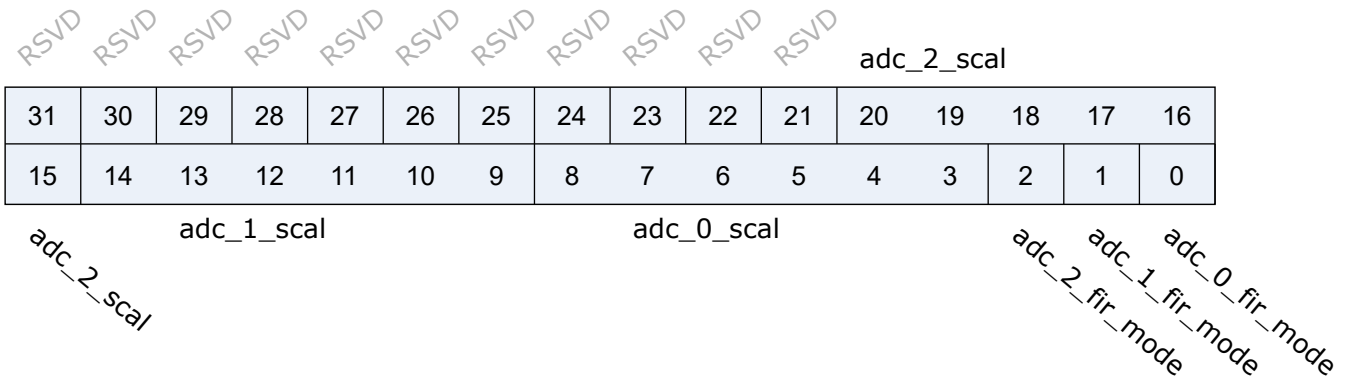
Address：0x2000ac04



Bits	Name	Type	Reset	Description
31	dac_itf_en	r/w	1'd0	1:enable dac to audio dma interface
30	adc_itf_en	r/w	1'd0	1:enable adc to audio dma interface
29	aud_tx1_sel	r/w	1'd1	audio tx1 source select; 0:dac ch0, 1:dac ch1
28	aud_tx0_sel	r/w	1'd0	audio tx0 source select; 0:dac ch0, 1:dac ch1
27:25	aud_rx4_sel	r/w	3'd4	audio rx4 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
24:22	aud_rx3_sel	r/w	3'd3	audio rx3 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
21:19	aud_rx2_sel	r/w	3'd2	audio rx2 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
18:16	aud_rx1_sel	r/w	3'd1	audio rx1 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
15:13	aud_rx0_sel	r/w	3'd0	audio rx0 source select; 0:adc ch0, 1:adc ch1, 2:adc ch2, 3:aec ch0, 4:aec ch1
12:7	RSVD			
6	aec_1_en	r/w	1'd0	1:enable aec ch1
5	aec_0_en	r/w	1'd0	1:enable aec ch0
4	dac_1_en	r/w	1'd0	1:enable dac ch1
3	dac_0_en	r/w	1'd0	1:enable dac ch0
2	adc_2_en	r/w	1'd0	1:enable adc ch2
1	adc_1_en	r/w	1'd0	1:enable adc ch1
0	adc_0_en	r/w	1'd0	1:enable adc ch0

21.4.3 pdm_adc_0

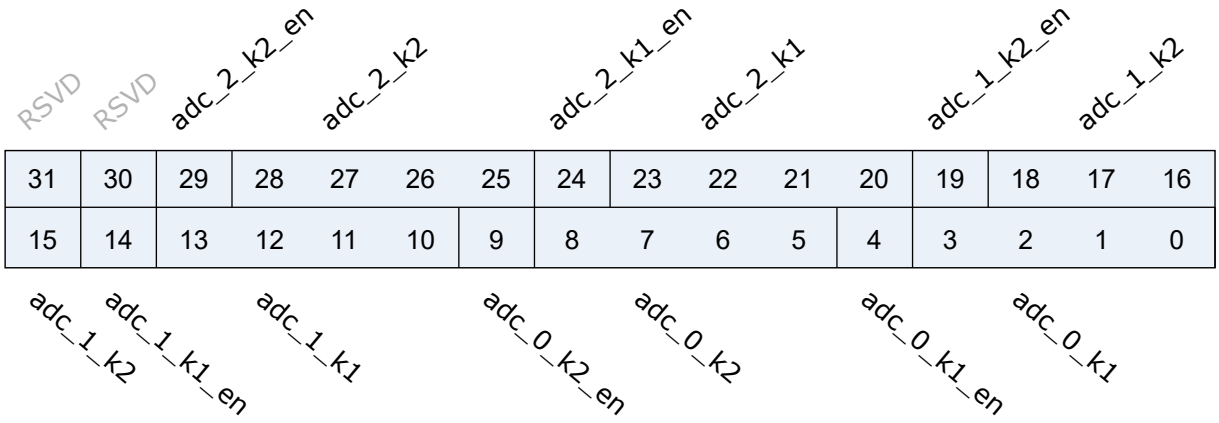
Address：0x2000ac08



Bits	Name	Type	Reset	Description
31:30	adc_lfsr_mode	r/w	2'd0	0:LFSR32, 1:LFSR24, 2:LFSR16, 3:LFSR12
29	adc_dither_data	r	1'd1	read LFSR out
28:21	RSVD			
20:15	adc_2_scal	r/w	6'd32	adc ch2 scaling value; u6.5
14:9	adc_1_scal	r/w	6'd32	adc ch1 scaling value; u6.5
8:3	adc_0_scal	r/w	6'd32	adc ch0 scaling value; u6.5
2	adc_2_fir_mode	r/w	1'd0	adc fir mode
1	adc_1_fir_mode	r/w	1'd0	adc fir mode
0	adc_0_fir_mode	r/w	1'd0	adc fir mode

21.4.4 pdm_adc_1

Address：0x2000ac0c

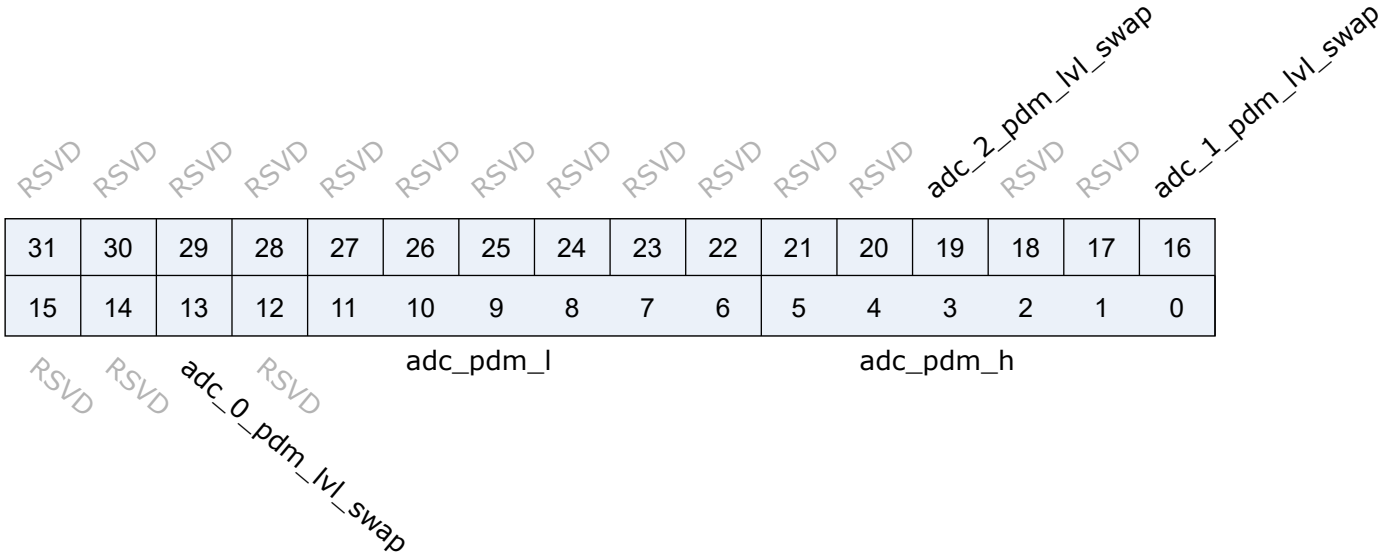


Bits	Name	Type	Reset	Description
31:30	RSVD			
29	adc_2_k2_en	r/w	1'd0	adc ch2 hpf parameter k2 enable
28:25	adc_2_k2	r/w	4'd13	adc ch2 hpf parameter k2
24	adc_2_k1_en	r/w	1'd1	adc ch2 hpf parameter k1 enable
23:20	adc_2_k1	r/w	4'd8	adc ch2 hpf parameter k1
19	adc_1_k2_en	r/w	1'd0	adc ch1 hpf parameter k2 enable
18:15	adc_1_k2	r/w	4'd13	adc ch1 hpf parameter k2
14	adc_1_k1_en	r/w	1'd1	adc ch1 hpf parameter k1 enable
13:10	adc_1_k1	r/w	4'd8	adc ch1 hpf parameter k1

Bits	Name	Type	Reset	Description
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

21.4.5 pdm_dac_0

Address：0x2000ac10



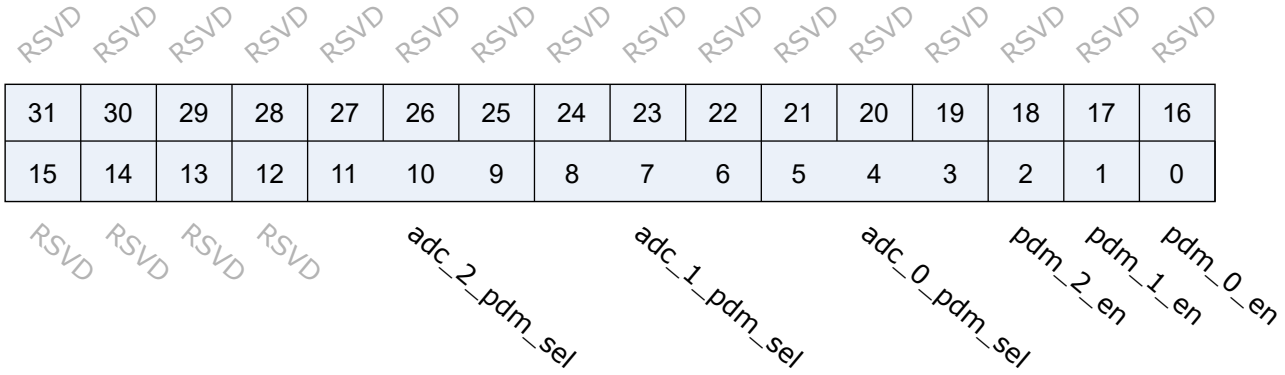
Bits	Name	Type	Reset	Description
31	RSVD			
30:28	mix_0_att_mode2	r/w	3'd0	0: 0db,1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
27:25	mix_0_att_mode1	r/w	3'd0	0: 0db,1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
24:23	mix_0_mode	r/w	2'd0	0: no mix, 1: mix second input, 2: mix sidetone/loopback
22:21	mix_0_sel	r/w	2'd0	0: 0, 1:adc ch0, 2: adc ch1, 3:adc ch2
20	adc_2_mash_bit_swap	r/w	1'd0	1:swap adc1_do1 and adc1_do2
19	adc_2_pdm_lvl_swap	r/w	1'd0	1:invert pdm input data
18	adc_2_src	r/w	1'd0	0:adc, 1:pdm
17	adc_1_mash_bit_swap	r/w	1'd0	1:swap adc2_do1 and adc2_do2
16	adc_1_pdm_lvl_swap	r/w	1'd0	1:invert pdm input data

Bits	Name	Type	Reset	Description
15	adc_1_src	r/w	1'd0	0:adc, 1:pdm
14	adc_0_mash_bit_swap	r/w	1'd0	1:swap adc3_do1 and adc3_do2
13	adc_0_pdm_lvl_swap	r/w	1'd0	1:invert pdm input data
12	adc_0_src	r/w	1'd0	0:adc, 1:pdm
11:6	adc_pdm_l	r/w	6'h3f	pdm low value
5:0	adc_pdm_h	r/w	6'h1	pdm high value
-1:31	RSVD			
30:28	mix_1_att_mode2	r/w	3'd0	0: 0db,1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
27:25	mix_1_att_mode1	r/w	3'd0	0: 0db,1:6db, 2:12db, 3:18db, 4:36db, 5:54db, 6:72db, 7:mute
24:23	mix_1_mode	r/w	2'd0	0: no mix, 1: mix second input, 2: mix sidetone/loopback
22:21	mix_1_sel	r/w	2'd0	0: 0, 1:adc ch0, 2: adc ch1, 3:adc ch2
20:17	RSVD			
16:15	dac_dsm_dither_prbs_mode	r/w	1'd0	dac dsm dither lfsr mode:0:LFSR32, 1:LFSR24, 2:LFSR16, 3:LFSR12
14	dac_dsm_dither_en	r/w	1'd1	1:enable dac dsm dither
13:11	dac_dsm_dither_amp	r/w	3'd0	dac dsm dither amplitue
10	dac_dsm_scaling_en	r/w	1'd1	1:enable dac dsm scaling
9:6	dac_dsm_scaling_factor	r/w	4'd15	dac dsm scaling value; u4.4
5	dac_dsm_order	r/w	1'd0	0: 2-order, 1: 3-order
4:2	RSVD			
1	dac_dem_out_swap	r/w	1'd0	1:swap dacldata and dacrdata
0	dac_dem_bypass	r/w	1'd0	1:bypass dac dwa
-1:14	RSVD			
13	aec_record_vld_4s_en	r/w	1'd0	0:aec record vld auto controlled by dac_rate, 1:enable aec record vld manual mode
12:11	aec_record_vld_4s_div	r/w	2'd0	aec record vld manual divide rate
10:8	aec_1_atten_mode	r/w	3'd0	aec ch1 attenuation mode: 0:no attenuation, 1:drop 1LSB, 2:drop 2LSB, 3:drop 3LSB, 4:drop 6LSB, 5:drop 9LSB, 6:drop 12LSB

Bits	Name	Type	Reset	Description
7:5	aec_0_atten_mode	r/w	3'd0	aec ch0 attenuation mode: 0:no attenuation, 1:drop 1LSB, 2:drop 2LSB, 3:drop 3LSB, 4:drop 6LSB, 5:drop 9LSB, 6:drop 12LSB
4:0	RSVD			

21.4.6 pdm_pdm_0

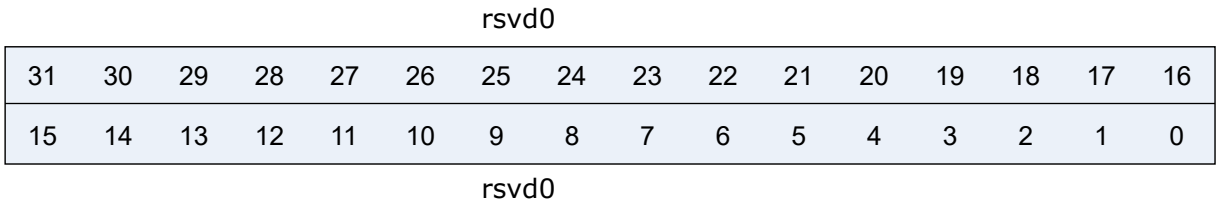
Address：0x2000ac1c



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:9	adc_2_pdm_sel	r/w	3'd2	adc ch2 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
8:6	adc_1_pdm_sel	r/w	3'd1	adc ch1 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
5:3	adc_0_pdm_sel	r/w	3'd0	adc ch0 source select: 0:pdm_0_l, 1:pdm_0_r, 2:pdm_1_l, 3:pdm_1_r, 4:pdm_2_l, 5:pdm_2_r
2	pdm_2_en	r/w	1'd0	1:enable pdm_2
1	pdm_1_en	r/w	1'd0	1:enable pdm_1
0	pdm_0_en	r/w	1'd0	1:enable pdm_0

21.4.7 pdm_rsvd0

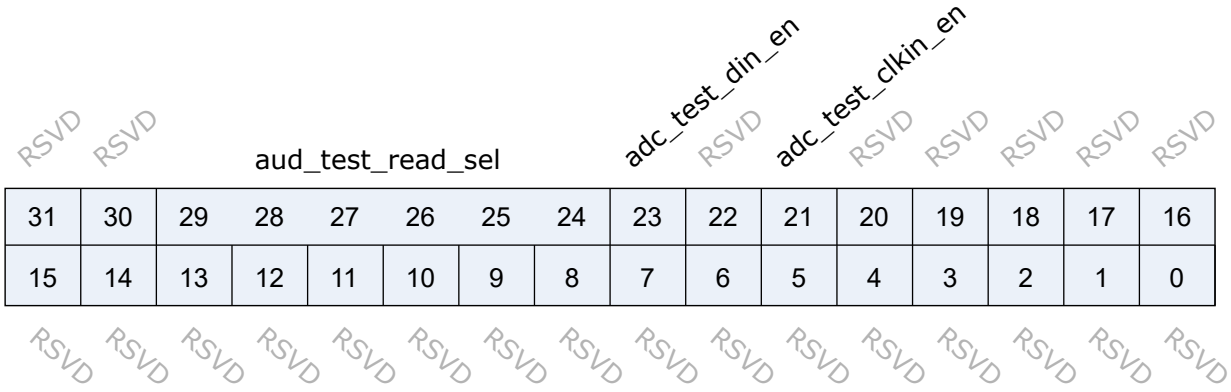
Address：0x2000ac20



Bits	Name	Type	Reset	Description
31:0	rsvd0	r/w	32'hffff	

21.4.8 pdm_dbg_0

Address：0x2000ac24



Bits	Name	Type	Reset	Description
31:30	RSVD			
29:24	aud_test_read_sel	r/w	6'd0	select aud_test_read(0x28) test point
23	adc_test_din_en	r/w	1'd0	1:enable adc test data input from GPIO
22	dac_test_din_en	r/w	1'd0	1:enable dac test data input from GPIO
21	adc_test_clkin_en	r/w	1'd0	1:enable adc test clcok input from GPIO
20	dac_test_clkin_en	r/w	1'd0	1:enable dac test clcok input from GPIO
19:18	audio_test_out_sel	r/w	2'd0	audio test data to GPIO select: 0:no data, 1:adc ch0/1/2, 2:dac ch0 dwa, 2:dac ch1 dwa
17:4	RSVD			
3:1	aud_sin_step	r/w	3'd2	step of dac audio sin table @FS=192k 0:div1, 1:div2, 2:div4, 3:div6, 4:div8, 5:div12, 6:div24

Bits	Name	Type	Reset	Description
0	aud_sin_en	r/w	1'd0	1:enable audio dac sin generator

21.4.9 pdm_dbg_1

Address：0x2000ac28

aud_test_read

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

aud_test_read

Bits	Name	Type	Reset	Description
31:0	aud_test_read	r	32'd0	audio test read value

21.4.10 pdm_dbg_2

Address：0x2000ac2c

RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	adc_in_2_test_sel	adc_in_1_test_sel	adc_in_0_test_sel	adc_2_fir_4s_en	adc_1_fir_4s_en	adc_0_fir_4s_en	adc_fir_4s_val			
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

adc_fir_4s_val

Bits	Name	Type	Reset	Description
31:26	RSVD			
25	adc_in_2_test_sel	r/w	1'd0	adc ch2 test data select; 1:from adc sin generator, 0:from fir force value
24	adc_in_1_test_sel	r/w	1'd0	adc ch1 test data select; 1:from adc sin generator, 0:from fir force value
23	adc_in_0_test_sel	r/w	1'd0	adc ch0 test data select; 1:from adc sin generator, 0:from fir force value
22	adc_2_fir_4s_en	r/w	1'd0	1:force adc ch2 fir output

Bits	Name	Type	Reset	Description
21	adc_1_fir_4s_en	r/w	1'd0	1:force adc ch1 fir output
20	adc_0_fir_4s_en	r/w	1'd0	1:force adc ch0 fir output
19:0	adc_fir_4s_val	r/w	20'd0	force value of adc fir output

21.4.11 pdm_dbg_3

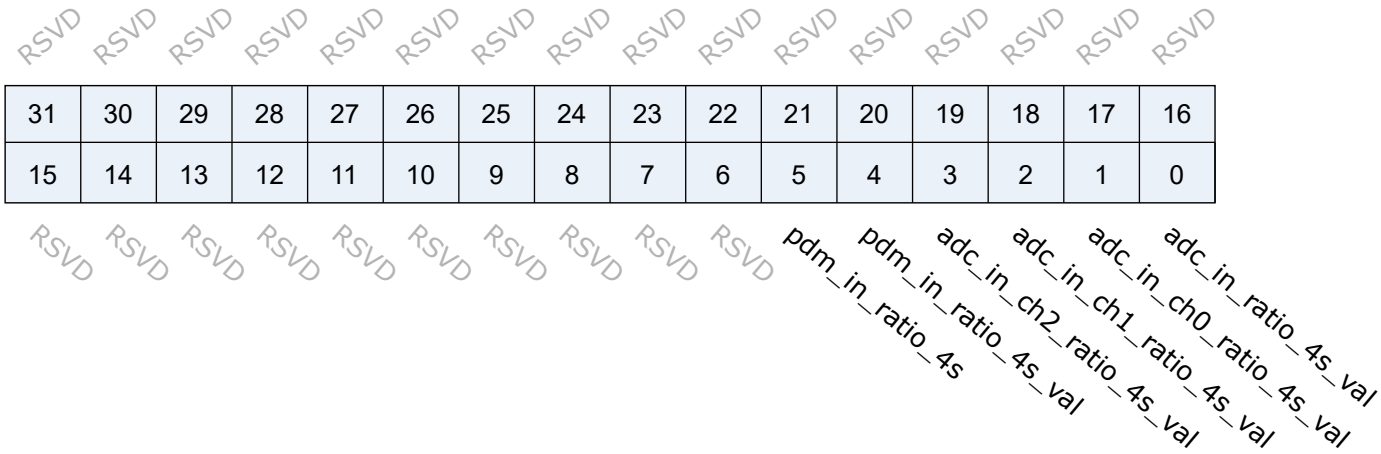
Address：0x2000ac30

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	RSVD

Bits	Name	Type	Reset	Description
31:24	RSVD			
23	dac_in_1_test_sel	r/w	1'd0	dac ch1 test data select; 0:from dac sin generator, 1:from dac force value
22	dac_in_0_test_sel	r/w	1'd0	dac ch0 test data select; 0:from dac sin generator, 1:from dac force value
21	dac_dwa_1_4s_en	r/w	1'd0	1:force dac ch1 dwa data from dac_4s_val[6:0]
20	dac_dwa_0_4s_en	r/w	1'd0	1:force dac ch0 dwa data from dac_4s_val[6:0]
19:0	dac_4s_val	r/w	20'd0	dac force value

21.4.12 pdm_dbg_4

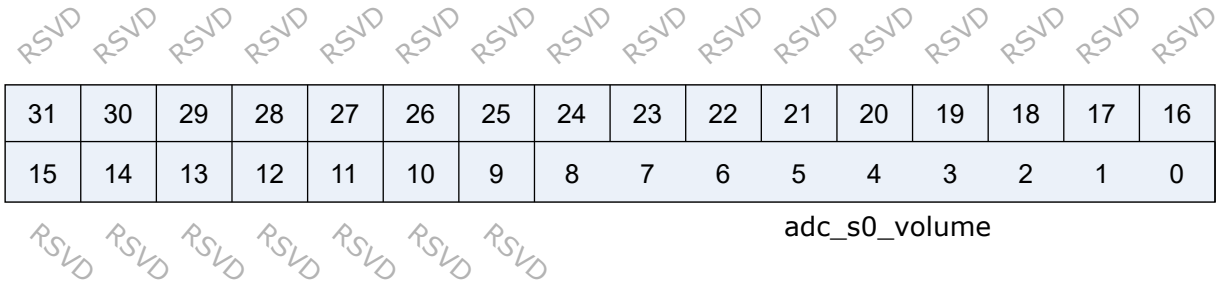
Address：0x2000ac34



Bits	Name	Type	Reset	Description
31:11	RSVD			
10:8	aec_fs_rate_4s_val	r/w	3'd0	
7:6	dac_out_ratio_4s_val	r/w	2'd0	
5	pdm_in_ratio_4s	r/w	1'd0	
4	pdm_in_ratio_4s_val	r/w	1'd0	
3	adc_in_ch2_ratio_4s_val	r/w	1'd0	
2	adc_in_ch1_ratio_4s_val	r/w	1'd0	
1	adc_in_ch0_ratio_4s_val	r/w	1'd0	
0	adc_in_ratio_4s_val	r/w	1'd0	

21.4.13 pdm_adc_s0

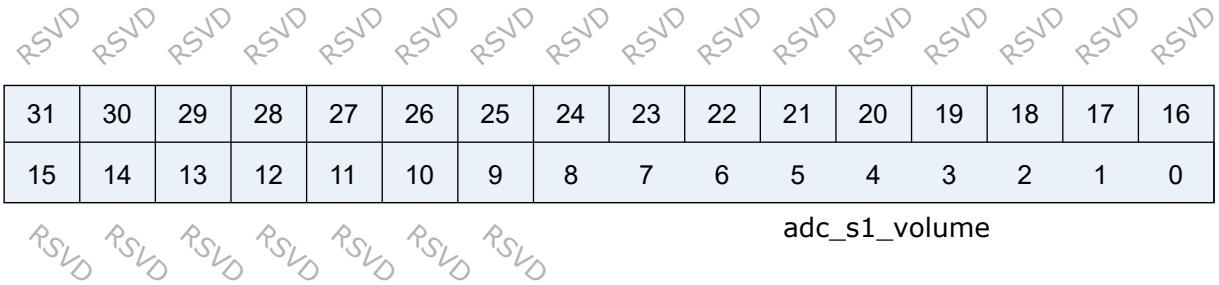
Address：0x2000ac38



Bits	Name	Type	Reset	Description
31:9	RSVD			
8:0	adc_s0_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

21.4.14 pdm_adc_s1

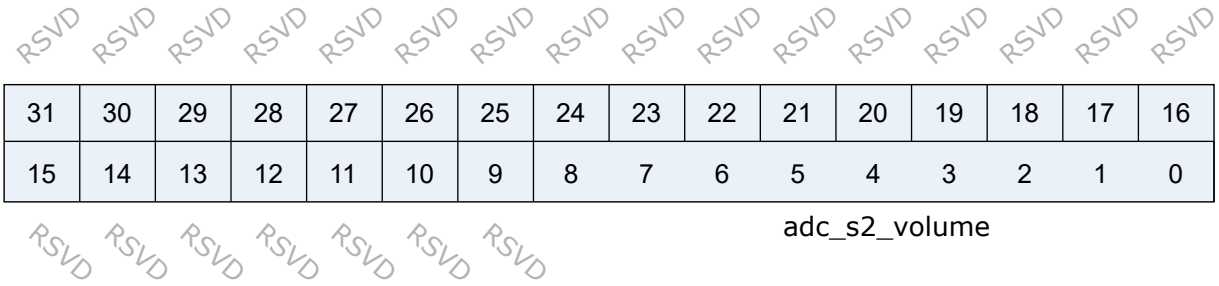
Address：0x2000ac3c



Bits	Name	Type	Reset	Description
31:9	RSVD			
8:0	adc_s1_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

21.4.15 pdm_adc_s2

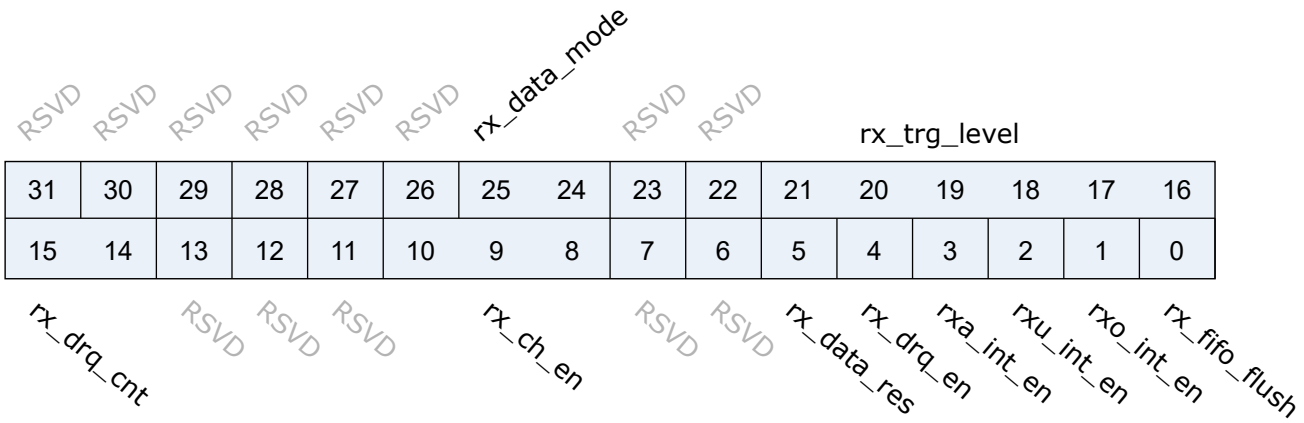
Address：0x2000ac40



Bits	Name	Type	Reset	Description
31:9	RSVD			
8:0	adc_s2_volume	r/w	9'd0	volume s9.1, -95.5dB +18dB in 0.5dB step

21.4.16 pdm_rx_fifo_ctrl

Address：0x2000ac80



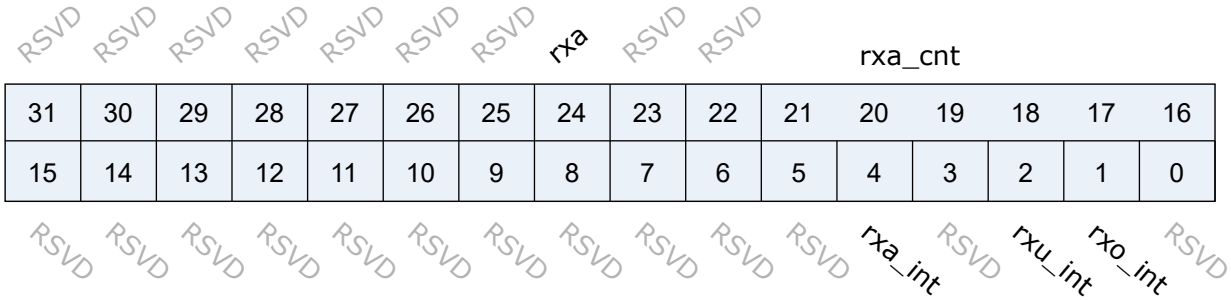
Bits	Name	Type	Reset	Description
31:26	RSVD			
25:24	rx_data_mode	r/w	2'b0	RX_FIFO_DATOUT_MODE. RX FIFO DATA Output Mode (Mode 0, 1, 2, 3) Mode 0: Valid data's MSB is at [31] of RX_FIFO register Mode 1: Valid data's MSB is at [23] of RX_FIFO register Mode 2: Valid data's MSB is at [19] of RX_FIFO register 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 20-bit received audio sample resolution: Mode 0: RXDATA[31:0] = FIFO_O[19:0], 12' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[19], FIFO_O[19:0], 4' h0 Mode 2: RXDATA[31:0] = 12FIFO_O[19], FIFO_O[19:0] Mode 3: RXDATA[31:0] = 16FIFO_O[19], FIFO_O[19:4] For 16-bit received audio sample resolution: Mode 0: RXDATA[31:0] = FIFO_O[19:4], 16' h0 Mode 1: RXDATA[31:0] = 8FIFO_O[19], FIFO_O[19:4], 8' h0 Mode 2: RXDATA[31:0] = 12FIFO_O[19], FIFO_O[19:4], 4'h0 Mode 3: RXDATA[31:0] = 16FIFO_O[19], FIFO_O[19:4]
23:22	RSVD			

Bits	Name	Type	Reset	Description
21:16	rx_trg_level	r/w	6'd23	RX_FIFO_TRG_LEVEL. RX FIFO Trigger Level (RXTL[5:0]) Interrupt and DMA request trigger level for RX FIFO Data Available condition IRQ/DRQ Generated when WLEVEL > RXTL[5:0] Notes: WLEVEL represents the number of valid samples in the RX FIFO
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[5:0] 01: IRQ/DRQ de-asserted when WLEVEL < 8 10: IRQ/DRQ de-asserted when WLEVEL < 16 11: IRQ/DRQ de-asserted when WLEVEL < 32 WLEVEL represents the number of valid samples in the RX FIFO
13:11	RSVD			
10:8	rx_ch_en	r/w	3'b0	RX_FIFO_DATIN_SRC. RX FIFO Data Input Source Select. 0: Disable 1: Enable Bit10: ADC3 data Bit9: ADC2 data Bit8: ADC1 data When some of the above bits set to ' 1', these data are always arranged in order from low-bit to high-bit.(bit8->bit10)
7:6	RSVD			
5	rx_data_res	r/w	1'b0	RX_SAMPLE_BITS. Receiving Audio Sample Resolution 0: 16 bits 1: 20 bits
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

Bits	Name	Type	Reset	Description
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
0	rx_fifo_flush	w1c	1'b0	ADC_FIFO_FLUSH. ADC FIFO Flush. Write '1' to flush TX FIFO, self clear to '0'.

21.4.17 pdm_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)
23:22	RSVD			
21:16	rxa_cnt	r	6'h0	RXA_CNT. RX FIFO Available Sample Word Counter
15:5	RSVD			

Bits	Name	Type	Reset	Description
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
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			

21.4.18 pdm_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.

22 AUDIO

22.1 Overview

An AUDIO processing module is built in the chip, supporting pdm/analog interface microphone recording and stereo playback.

22.2 Features

- Three 20-bit ADCs are integrated to support three analog mic differential inputs
  - Sampling rate: 8k~96k
  - Signal to noise ratio (AW): 97 dB @ 0 dB gain
  - Harmonic distortion + noise: -87 dB @ 0 dB gain
  - Analog pre-amplifier gain: 0 dB, 6~42 dB, 3 dB per gear
- A low-noise bias power supply with adjustable voltage supplies power to the analog MIC, and the adjustment range is 1.8 V~2.5 V
- Adjustable high-pass filter and independent digital volume control for ADC channel
- A 5-band parametric equalizer for ADC channel
- Support digital mic interface, with input GPIO multiplexed
- Integrated with two 20-bit DACs that support two analog LINEOUT differential outputs with ramp power-on function
  - Sampling rate: 8k~192k
  - Signal to noise ratio (AW): 103dB @ 0dB gain
  - Harmonic distortion + noise: -90dB @ 0dB gain
  - Analog output gain: 0 dB, 6~42 dB, 3 dB per gear

- Independent digital volume control that supports soft volume adjustment/mute, suitable for DAC playback channel
- A 10-band parametric equalizer supporting dynamic range control and flexible adjustment, suitable for DAC playback channel
- Supports voice activity detection that suits low-power voice wakeup applications
- TX/RX FIFO of 32-bit width
- Supports DMA transfer mode

22.3 Functional Description

The block diagram of AUDIO is shown as follows.

AUIDO has three ADCs, which are connected to the volume control module after performing PGA gain and passing through decimation and HPF filters. Users can control mute/unmute, volume, and audio fade-in/fade-out effect through the volume control module. The 5-band EQ filter can improve the audio performance, perform gain and attenuation in each frequency band of the audio, and thus optimize the recording timbre. AGC will adjust volume according to the amplitude of audio data, to enhance the recording performance and reduce the top distortion of recording. The built-in VAD wakeup module can distinguish the noise signal from the useful audio signal in the environment, and when a valid audio signal is detected, a VAD_INT event will be generated. The recorded data will be stored in the FIFO with a depth of 32, and the storage format in FIFO is controlled by the register RX_FIFO_CTRL[25:24]. RX_FIFO_CTRL[5] configures the resolution of sampling. The PDM digital interface or analog microphone interface can be used as the recording data interface, and the interface type is controlled by the DAC_0[12] DAC_0[15] DAC_0[18] bit.

AUDIO has two DACs. After being taken out from FIFO, processed by the audio mixer and EQ filter, undergone volume control and inserted into the filter, the data is modulated into analog signal for driving the speaker to play. The audio mixer can mix the audio data read out from FIFO with the recording data of ADC channel. The EQ filter performs gain reduction on each frequency band, thus changing the timbre of the played audio. The volume control module will control the audio volume and the fade-in/fade-out effect.

22.3.1 AUDIO Interrupt

AUDIO 有着丰富的中断控制，包括以下几种中断模式：

AUDIO supports the following interrupt control modes:

- TX FIFO request interrupt
- TX FIFO underrun interrupt
- TX FIFO overrun interrupt
- RX FIFO request interrupt
- RX FIFO underrun interrupt

- RX FIFO overrun interrupt
- Recording volume adjustment interrupt
- Playback volume adjustment interrupt
- VAD interrupt

A TX FIFO request interrupt will be generated when TX_DRQ_CNT in TX_FIFO_CTRL is greater than TX_TRG_-LEVEL. When the condition is not met, the interrupt flag will be cleared automatically.

When there is no data in TX FIFO, but the user enables TX FIFO modulation through TX_CH_EN in TX_FIFO_CTRL, the TX FIFO underrun interrupt will be generated.

When the user fills in data that exceeds the maximum depth of TX FIFO, it will lead to TX FIFO overflow and cause a TX FIFO overrun interrupt.

A RX FIFO request interrupt will be generated when RX_DRQ_CNT in RX_FIFO_CTRL is greater than RX_TRG_-LEVEL. When the condition is not met, the interrupt flag will be 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 will be generated.

When the user fills in data that exceeds the maximum depth of RX FIFO, it will lead to RX FIFO overflow and cause a RX FIFO overrun interrupt.

When the recording volume of ADC is completely adjusted, it will trigger the ADC volume adjustment completion interrupt. In general application scenarios, the volume adjustment of ADC is completed immediately.

When the playback volume of DAC is completely adjusted, it will trigger the DAC volume adjustment completion interrupt. As the playback of DAC supports fade-in/fade-out effect, the volume adjustment may not be completed immediately. The adjustment time depends on the register DAC_H_CTRL_MODE.

22.3.2 FIFO Format Control

TX_FIFO_CTRL and RX_FIFO_CTRL (collectively FIFO_CTRL, as they are the same) decide the format of audio data stored in FIFO.

Users select the resolution of audio by configuring FIFO_CTRL[5].

When 16-bit resolution is selected, the FIFO controller supports the following four data storage formats, which are determined by FIFO_CTRL[25:24].

- Mode 0:
  - DATA[31:0] = {FIFO[19:14],16’h0}
- Mode 1:
  - DATA[31:0] = {8{FIFO[19]},FIFO[19:4],8’h0}

- Mode 2:
  - DATA[31:0] = {12{FIFO[19]},FIFO[19:4],4’h0}
- Mode 3:
  - DATA[31:0] = {16{FIFO[19]},FIFO[19:4]}

During recording or playing, the conversion result of 32-bit data or the digital audio source to be demodulated is on the left, denoted as DATA[31:0]. The format of data to be stored in FIFO is shown on the right. When the actual resolution of the recording is 20-bit, if the 16-bit resolution is selected, the data with 20-bit resolution must be properly cut. Thus, the high 16 bits of the data with 20-bit resolution are selected as the final result FIFO[19:14] and stored in the position of the high 16 bits, while the low 16 bits are filled with 0. Modes 1, 2, and 3 are expressed in the same way as Mode 0. The 8{FIFO[19]} symbol indicates that the high 8 bits will be padded with the value of bit[19].

Therefore, when a 16-bit resolution is selected, AUDIO provides four modes to store the digital results of conversion/output.

In the case of 20-bit resolution, the storage formats in the four modes are as follows:

- Mode 0:
  - DATA[31:0] = {FIFO[19:0],12’h0}
- Mode 1:
  - DATA[31:0] = {8{FIFO[19]},FIFO[19:0],4’h0}
- Mode 2:
  - DATA[31:0] = {12{FIFO[19]},FIFO[19:0]}
- Mode 3:
  - DATA[31:0] = {16{FIFO[19]},FIFO[19:4]}

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

22.3.3 Startup of FIFO and DMA transfer

The data in FIFO of the AUDIO module can be transferred by DMA.

The user can obtain the current amount of valid data in FIFO in real time through the register RX_FIFO_STATUS/TX_-FIFO_STATUS.

The FIFO count threshold (8/16/32) for initiating DMA request is selected by configuring FIFO_CTRL[15:14], or can be determined by FIFO_CTRL[22:16].

When the count value is greater than the set threshold, and the FIFO of the channel corresponding to TX_FIFO_-CTRL[9:8] or RX_FIFO_CTRL[12:8] is enabled, a DMA transfer will be initiated.

When TX FIFO is started, if there is no valid data in TX FIFO, the tx underrun error will be triggered. Therefore, the software configuration sequence must be followed.

23 PSRAM Contorller

23.1 Overview

The chip is equipped with a PSRAM controller that can drive various types of PSRAM.

23.2 Features

- Supports up to 200 MHz PSRAM clock
- Supports linear burst and warp burst modes

23.3 Functional Description

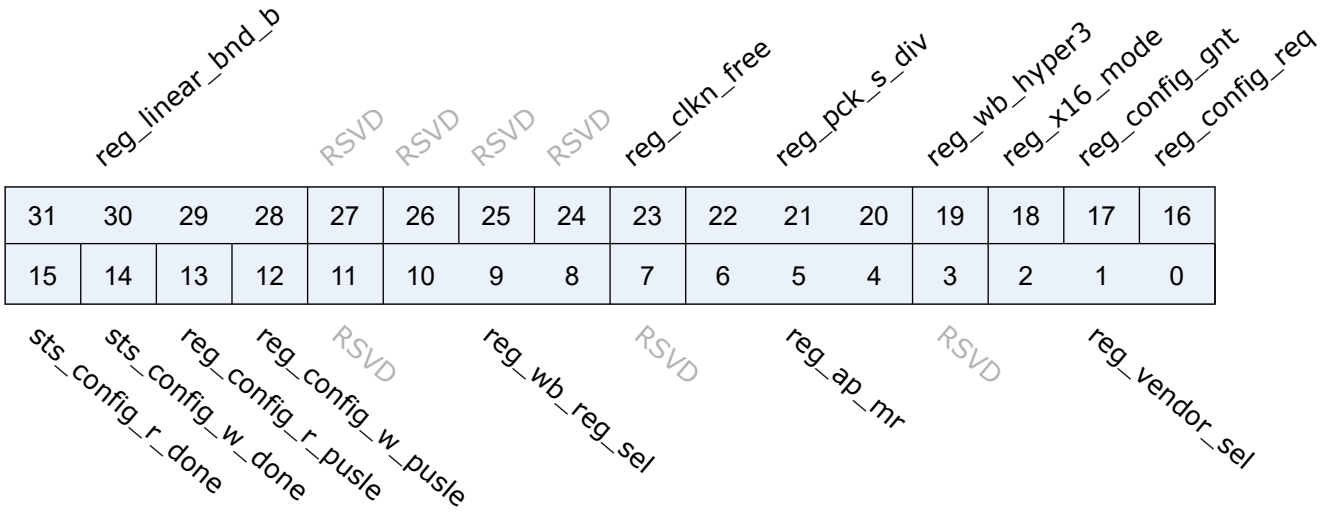
The PSRAM controller can read and write the internal registers and memory of various types of PSRAM.

23.4 Register description

Name	Description
psram_configure	

23.4.1 psram_configure

Address：0x20052000



Bits	Name	Type	Reset	Description
31:28	reg_linear_bnd_b	r/w	4'd10	Linear burst boundary shift bit
27:24	RSVD			
23	reg_clkn_free	r/w	1'b1	Clock# free run
22:20	reg_pck_s_div	r/w	3'd0	EMI AXI bus clock division from psram_ck
19	reg_wb_hyper3	r/w	1'b0	Winbond Hyper3 bus
18	reg_x16_mode	r/w	1'b0	16-bit pSRAM mode enable
17	reg_config_gnt	r	1'b0	pSRAM register configure grant
16	reg_config_req	r/w	1'b0	pSRAM register configure request
15	sts_config_r_done	r	1'b1	pSRAM register read done
14	sts_config_w_done	r	1'b1	pSRAM register write done
13	reg_config_r_pusle	w1p	1'b0	pSRAM configuration read enable
12	reg_config_w_pusle	w1p	1'b0	pSRAM configuration write enable
11	RSVD			
10:8	reg_wb_reg_sel	r/w	2'd0	Winbond pSRAM Register R/W selection 3'd0 - ID0 / 3'd1 - ID1 3'd2 - CR0 / 3'd3 - CR1 /3'd4 - CR2 3'd5 - CR3 / 3'd6 - CR4
7	RSVD			

Bits	Name	Type	Reset	Description
6:4	reg_ap_mr	r/w	3'd0	APMemory pSRAM Mode Register R/W selection 3'd0=MA0 / 3'd1=MA1 / 3'd2=MA2 / 3'd3=MA3 / 3'd4=MA4 / 3'd6=MA6 / 3'd7=MA8
3	RSVD			
2:0	reg_vendor_sel	r/w	3'b010	pSRAM vendor selection [0] - Winbond [1] - APM_XCELLA [2] - APM_HYPER
-1:32	RSVD			
31:16	sts_config_read	r	16'd0	full data from psram register read
15	reg_ck_edge_nali	r/w	1'b0	pSRAM clock 2x1x edge alignment 1'b0 - 2x/1x edge align 1'b1 - 2x/1x edge stagger
14	reg_psram_resetb	r/w	1'b1	pSRAM reset pin control
13	reg_force_ceb_high	r/w	1'b0	Force pSRAM ceb pin high
12	reg_force_ceb_low	r/w	1'b0	Force pSRAM ceb pin low(higher priority)
11	reg_wb_bl2_mask	r/w	1'b1	pSRAM Winbond burst length bit2 mask
10	reg_dqs_latch_inv	r/w	1'b0	pSRAM DQS latch invert
9	reg_state_hold_tick	r/w	1'b0	pSRAM hold state clock tick
8	reg_wc_sw_en	r/w	1'd0	pSRAM wait cycle sw mode enable
7	RSVD			
6:0	reg_wc_sw	r/w	7'd0	pSRAM sw mode wait cycle value
-1:32	RSVD			
31:16	reg_mask_r_fifo_rem	r/w	16'h0	Read data fifo remain cnt threshold
15:0	reg_mask_w_fifo_cnt	r/w	16'h0	Write data fifo data cnt threshold
-1:32	RSVD			
31:24	reg_addr_mask	r/w	8'h1f	pSRAM memory address mask for [27:20]
23	reg_pwrap_sw_en	r/w	1'd0	pSRAM WRAP sw setting enable (default off) Note - When using APS256XXN and setting burst length to 2'b11, it is necessary to set this register to 1
22:20	RSVD			

Bits	Name	Type	Reset	Description
19:16	reg_pwrap_sw_sht_b	r/w	4'd8	pSRAM WRAP sw setting value Note - When using APS256XXN and setting burst length to 2'b11, it is necessary to set this register to 4'd11
15	RSVD			
14:8	reg_dqs_rel_val	r/w	7'h20	pSRAM dqs_oen release counter value
7	reg_hc_sw_en	r/w	1'd0	pSRAM hold cycle sw mode enable
6:0	reg_hold_cycle_sw	r/w	7'd8	pSRAM sw mode hold cycle value
-1:32	RSVD			
31	reg_wb_sw_rst	r/w	1'b0	Winbond pSRAM CR1 - Software Reset Trigger software reset (if device supprts)
30	reg_wb_mclk_type	r/w	1'b1	Winbond pSRAM CR1 - Master Clock Type 0 - Differential CK#, CK 1 - Single Ended
29	reg_wb_ipd	r/w	1'b0	Winbond pSRAM CR1 - Input Power Down 0 - Normal 1 - Enter Input Power Down Mode
28:26	RSVD			
25	reg_wb_linear_dis	r/w	1'b0	Winbond pSRAM linear burst disable
24	reg_wb_hybrid_slp	r/w	1'b0	Winbond pSRAM CR1 - Hybrid Sleep Mode configuration 1'b0 - Normal operation 1'b1 - Enter Hybrid Sleep Mode
23:21	RSVD			
20:16	reg_wb_pasr	r/w	5'd0	Winbond pSRAM CR1 - Partial Array Refresh + Distributed Refresh Interval ———————————————————————————— For pSRAM >4MB [20:18] ; For pSRAM=4MB [18:16] 3'b000 - Full array / 3'b001 - Bottom 1/2 array / 3'b010 - Bottom 1/4 array / 3'b011 - Bottom 1/8 array 3'b100 - RSVD / 3'b101 - Top 1/2 array / 3'b110 - Top 1/4 array / 3'b111 - Top 1/8 array ———————————————————————————— For pSRAM >4MB [17:16] 2'b10 - 1us / 2'b01 - 4is
15:14	RSVD			

Bits	Name	Type	Reset	Description
13	reg_wb_dpd_dis	r/w	1'b1	Winbond pSRAM CR0 - Deep Power-Down Disable configuration 1'b0 - Deep sleep enable 1'b1 - Normal operation
12	reg_wb_fix_latency	r/w	1'b1	Winbond pSRAM CR1 - Fix Latency configuration 1'b0 - Variable 1'b1 - Fixed
11	RSVD			
10:8	reg_wb_burst_length	r/w	3'd7	Winbond pSRAM CR0 - Burst Length configuration ———————————— For Non-Hyper Bus3 3'b100 - 128 Byte 3'b101 - 64 Byte 3'b110 - 16 Byte 3'b111 - 32 Byte ———————————— For Hyper Bus3 3'b100 - 128 Beat 3'b101 - 64 Beat 3'b110 - 16 Beat 3'b111 - 32 Beat 3'b011 - 512 Beat
7	reg_wb_hybrid_en	r/w	1'b1	Winbond pSRAM CR0 - Hybrid Burst Enable 1'b0 - hybrid burst 1'b1 - wrapped burst
6:4	reg_wb_drive_st	r/w	3'd0	Winbond pSRAM CR0 - Drive Strength configuration for 4MB pSRAM 3'b00 - 50Ω / 3'b01 - 35Ω / 3'b10 - 100Ω / 3'b11 - 200Ω For 8MB pSRAM 3'b000 - 34Ω / 3'b001 - 115Ω / 3'b010 - 67Ω / 3'b011 - 46Ω 3'b100 - 34Ω / 3'b101 - 27Ω / 3'b110 - 22Ω / 3'b111 - 19Ω

Bits	Name	Type	Reset	Description
3:0	reg_wb_latency	r/w	4'd2	Winbond pSRAM CR0 - Latency Counter configuration ——————————————— For Non-Hyper Bus3 4'b0000 - Latency = 5 / Max freq. = 133 MHz 4'b0001 - Latency = 6 / Max freq. = 166 MHz 4'b0010 - Latency = 7 / Max freq. = 200 MHz 4'b1110 - Latency = 3 / Max freq. = 83 MHz 4'b1111 - Latency = 4 / Max freq. = 100 MHz Others - RSVD ——————————————— For Hyper Bus3 4'b0111 - Latency = 14 / Max freq. = 400 MHz 4'b1001 - Latency = 19 / Max freq. = 533 MHz
-1:31	RSVD			
30	sts_wb_mclk_type	r	1'b1	Winbond pSRAM CR1 - Master Clock Type 0 - Differential CK#, CK 1 - Single Ended
29:25	RSVD			
24	sts_wb_hybrid_slp	r	1'b0	Winbond pSRAM CR1 - Hybrid Sleep Mode configuration 1'b0 - Normal operation 1'b1 - Enter Hybrid Sleep Mode
23:21	RSVD			
20:16	sts_wb_pasr	r	5'd0	Winbond pSRAM CR1 - Partial Array Refresh + Distributed Refresh Interval For pSRAM >4MB [20:18] ; For pSRAM=4MB [18:16] 3'b000 - Full array / 3'b001 - Bottom 1/2 array / 3'b010 - Bottom 1/4 array / 3'b011 - Bottom 1/8 array 3'b100 - RSVD / 3'b101 - Top 1/2 array / 3'b110 - Top 1/4 array / 3'b111 - Top 1/8 array For pSRAM >4MB [17:16] 2'b10 - 1us / 2'b01 - 4is
15:14	RSVD			
13	sts_wb_dpd_dis	r	1'b1	Winbond pSRAM CR0 - Deep Power-Down Disable configuration 1'b0 - Deep sleep enable 1'b1 - Normal operation
12	sts_wb_fix_latency	r	1'b1	Winbond pSRAM CR0 - Fix Latency configuration 1'b0 - Variable 1'b1 - Fixed

Bits	Name	Type	Reset	Description
11	RSVD			
10:8	sts_wb_burst_length	r	3'd3	Winbond pSRAM CR0 - Burst Length configuration ———————————— For Non-Hyper Bus3 3'b100 - 128 Byte 3'b101 - 64 Byte 3'b110 - 16 Byte 3'b111 - 32 Byte ———————————— For Hyper Bus3 3'b100 - 128 Beat 3'b101 - 64 Beat 3'b110 - 16 Beat 3'b111 - 32 Beat 3'b011 - 512 Beat
7	sts_wb_hybrid_en	r	1'b1	Winbond pSRAM CR0 - Hybrid Burst Enable 1'b0 - hybrid burst 1'b1 - wrapped burst
6:4	sts_wb_drive_st	r	2'd0	Winbond pSRAM CR0 - Drive Strength configuration for 4MB pSRAM 3'b00 - 50Ω / 3'b01 - 35Ω / 3'b10 - 100Ω / 3'b11 - 200Ω For 8MB pSRAM 3'b000 - 34Ω / 3'b001 - 115Ω / 3'b010 - 67Ω / 3'b011 - 46Ω 3'b100 - 34Ω / 3'b101 - 27Ω / 3'b110 - 22Ω / 3'b111 - 19Ω
3:0	sts_wb_latency	r	4'd2	Winbond pSRAM CR0 - Latency Counter configuration ———————————————————— For Non-HS mode : 4'b0000 - Latency = 5 / Max freq. = 133 MHz 4'b0001 - Latency = 6 / Max freq. = 166 MHz 4'b0010 - Latency = 7 / Max freq. = 200 MHz 4'b1110 - Latency = 3 / Max freq. = 83 MHz 4'b1111 - Latency = 4 / Max freq. = 100 MHz Others - RSVD ———————————————————— For HS mode 4'b0111 - Latency = 14 / Max freq. = 400 MHz 4'b1001 - Latency = 19 / Max freq. = 533 MHz
-1:4	RSVD			

Bits	Name	Type	Reset	Description
3:0	reg_wb_zq_code	r/w	4'd0	Winbond pSRAM CR2 - ZQ Calibration Code 4'b1111 : ZQ Calibration command 4'b0011 : ZQ Reset
-1:29	RSVD			
28	reg_glb_reset_pulse	w1p	1'b0	AP Memory pSRAM global reset enable
27:26	RSVD			
25	reg_ap_linear_dis	r/w	1'b0	AP Memory pSRAM linear burst disable
24	reg_ap_r_latency_type	r/w	1'b0	AP Memory pSRAM configure MR0 - read latency type 1'b0 - Variable 1'b1 - Fixed
23	RSVD			
22:20	reg_ap_r_latency_code	r/w	3'b010	AP Memory pSRAM configure MR0 - read latency code 3'b000 - Latency = 3 / Max freq. = 66MHz 3'b001 - Latency = 4 / Max freq. = 104MHz 3'b010 - Latency = 5 / Max freq. = 133MHz 3'b011 - Latency = 6 / Max freq. = 166MHz 3'b100 - Latency = 7 / Max freq. = 200MHz Others - RSVD
19:18	reg_ap_rf	r/w	1'b0	AP Memory pSRAM configure MR4 - refresh frequency
17:16	reg_ap_drive_st	r/w	2'b01	AP Memory pSRAM configure MR0 - drive strength 2'b00 - Full (25Ω) 2'b01 - Half (50Ω) 2'b10 - 1/4 (100Ω) 2'b11 - 1/8 (200Ω)
15	RSVD			
14:12	reg_ap_w_latency_code	r/w	3'b010	AP Memory pSRAM configure MR4 - write latency code 3'b000 - Latency = 3 / Max freq. = 66MHz 3'b100 - Latency = 4 / Max freq. = 104MHz 3'b010 - Latency = 5 / Max freq. = 133MHz 3'b110 - Latency = 6 / Max freq. = 166MHz 3'b001 - Latency = 7 / Max freq. = 200MHz Others - RSVD
11	RSVD			

Bits	Name	Type	Reset	Description
10:8	reg_ap_pasr	r/w	3'b000	AP Memory pSRAM configure MR4 - partial array refresh 3'b000 - Full array 3'b001 - Bottom 1/2 array 3'b010 - Bottom 1/4 array 3'b011 - Bottom 1/8 array 3'b100 - RSVD 3'b101 - Top 1/2 array 3'b110 - Top 1/4 array 3'b111 - Top 1/8 array
7	reg_ap_sleep	r/w	1'b0	AP Memory pSRAM configure MR6 - Half Sleep enable
6	reg_ap_dpd	r/w	1'b0	AP Memory pSRAM configure MR6 - Deep Power Down enable
5	reg_ap_rbx	r/w	1'b0	AP Memory pSRAM configure MR8 - cross boundary enable 1'b0 - Read within 1K boundary 1'b1 - Read cross 1K boundary
4	reg_ap_burst_type	r/w	1'b1	AP Memory pSRAM configure MR8 - burst type 1'b0 - Normal Wrap 1'b1 - Hybrid Wrap
3:2	RSVD			
1:0	reg_ap_burst_length	r/w	2'b01	AP Memory pSRAM configure MR8 - burst length For APS6408L□OBx 2'd0 - 16 Byte 2'd1 - 32 Byte 2'd2 - 64 Byte 2'd3 - 1K Byte For APS256XXN-OBRx 2'd0 - 16 Beat 2'd1 - 32 Beat 2'd2 - 64 Beat 2'd3 - 2K Byte
-1:25	RSVD			
24	sts_ap_r_latency_type	r	1'b0	AP Memory pSRAM status MR0 - read latency type 1'b0 - Variable 1'b1 - Fixed
23	RSVD			

Bits	Name	Type	Reset	Description
22:20	sts_ap_r_latency_code	r	3'b010	AP Memory pSRAM status MR0 - read latency code 3'b000 - Latency = 3 / Mas freq. = 66MHz 3'b001 - Latency = 4 / Mas freq. = 104MHz 3'b010 - Latency = 5 / Mas freq. = 133MHz 3'b011 - Latency = 6 / Mas freq. = 166MHz 3'b100 - Latency = 7 / Mas freq. = 200MHz Others - RSVD
19:18	sts_ap_rf	r	1'b0	AP Memory pSRAM configure MR4 - refresh frequency
17:16	sts_ap_drive_st	r	2'b01	AP Memory pSRAM status MR0 - drive strength 2'b00 - Full (25Ω) 2'b01 - Half (50Ω) 2'b10 - 1/4 (100Ω) 2'b11 - 1/8 (200Ω)
15	RSVD			
14:12	sts_ap_w_latency_code	r	3'b010	AP Memory pSRAM status MR4 - write latency code 3'b000 - Latency = 3 / Mas freq. = 66MHz 3'b100 - Latency = 4 / Mas freq. = 104MHz 3'b010 - Latency = 5 / Mas freq. = 133MHz 3'b110 - Latency = 6 / Mas freq. = 166MHz 3'b001 - Latency = 7 / Mas freq. = 200MHz Others - RSVD
11	RSVD			
10:8	sts_ap_pasr	r	3'b000	AP Memory pSRAM status MR4 - partial array refresh 3'b000 - Full array 3'b001 - Bottom 1/2 array 3'b010 - Bottom 1/4 array 3'b011 - Bottom 1/8 array 3'b100 - RSVD 3'b101 - Top 1/2 array 3'b110 - Top 1/4 array 3'b111 - Top 1/8 array
7	RSVD			
6	sts_ap_x16_mode	r	1'b0	AP Memory pSRAM status MR8 - X16 mode enable
5	sts_ap_rbx	r	1'b0	AP Memory pSRAM status MR8 - cross boundary enable 1'b0 - Read within 1K boundary 1'b1 - Read cross 1K boundary
4	sts_ap_burst_type	r	1'b1	AP Memory pSRAM status MR8 - burst type 1'b0 - Normal Wrap 1'b1 - Hybrid Wrap

Bits	Name	Type	Reset	Description
3:2	RSVD			
1:0	sts_ap_burst_length	r	2'b01	AP Memory pSRAM configure MR8 - burst length For APS6408L□OBx 2'd0 - 16 Byte 2'd1 - 32 Byte 2'd2 - 64 Byte 2'd3 - 1K Byte For APS256XXN-OBRx 2'd0 - 16 Beat 2'd1 - 32 Beat 2'd2 - 64 Beat 2'd3 - 2K Byte
-1:9	RSVD			
8	reg_wrap2incr_en	r/w	1'b1	Auto wrap to increament enable
7	RSVD			
6:0	reg_adq_rel_val	r/w	7'h20	pSRAM adq_oen release counter value
-1:32	RSVD			
31:24	reg_delay_sel_o_clk	r/w	8'h0	Output Delay Chain Control - clk
23:16	reg_delay_sel_o_clk_n	r/w	8'h0	Output Delay Chain Control - clk_n
15:8	reg_delay_sel_o_ceb	r/w	8'h0	Output Delay Chain Control - ceb
7:0	reg_delay_sel_o_dqs_oen0	r/w	8'h0	Output Delay Chain Control - dqs_oen[0]
-1:32	RSVD			
31:24	reg_delay_sel_o_dqs0	r/w	8'h0	Output Delay Chain Control - dqs_o[0]
23:16	reg_delay_sel_o_adq_oen0	r/w	8'h0	Output Delay Chain Control - adq_oen[0]
15:8	reg_delay_sel_o_adq0	r/w	8'h0	Output Delay Chain Control - adq_o[0]
7:0	reg_delay_sel_o_adq1	r/w	8'h0	Output Delay Chain Control - adq_o[1]
-1:32	RSVD			
31:24	reg_delay_sel_o_adq2	r/w	8'h0	Output Delay Chain Control - adq_o[2]
23:16	reg_delay_sel_o_adq3	r/w	8'h0	Output Delay Chain Control - adq_o[3]
15:8	reg_delay_sel_o_adq4	r/w	8'h0	Output Delay Chain Control - adq_o[4]
7:0	reg_delay_sel_o_adq5	r/w	8'h0	Output Delay Chain Control - adq_o[5]
-1:32	RSVD			
31:24	reg_delay_sel_o_adq6	r/w	8'h0	Output Delay Chain Control - adq_o[6]
23:16	reg_delay_sel_o_adq7	r/w	8'h0	Output Delay Chain Control - adq_o[7]

Bits	Name	Type	Reset	Description
15:8	reg_delay_sel_i_adq0	r/w	8'h0	Input Delay Chain Control - adq_i[0]
7:0	reg_delay_sel_i_adq1	r/w	8'h0	Input Delay Chain Control - adq_i[1]
-1:32	RSVD			
31:24	reg_delay_sel_i_adq2	r/w	8'h0	Input Delay Chain Control - adq_i[2]
23:16	reg_delay_sel_i_adq3	r/w	8'h0	Input Delay Chain Control - adq_i[3]
15:8	reg_delay_sel_i_adq4	r/w	8'h0	Input Delay Chain Control - adq_i[4]
7:0	reg_delay_sel_i_adq5	r/w	8'h0	Input Delay Chain Control - adq_i[5]
-1:32	RSVD			
31:24	reg_delay_sel_i_adq6	r/w	8'h0	Input Delay Chain Control - adq_i[6]
23:16	reg_delay_sel_i_adq7	r/w	8'h0	Input Delay Chain Control - adq_i[7]
15:0	reg_delay_sel_i_dqs0	r/w	16'h0	Input Delay Chain Control - dqs_i[0]
-1:32	RSVD			
31:24	reg_delay_sel_o_dqs1	r/w	8'h0	Output Delay Chain Control - dqs_o[1]
23:16	reg_delay_sel_o_adq_oen1	r/w	8'h0	Output Delay Chain Control - adq_oen[1]
15:8	reg_delay_sel_o_adq8	r/w	8'h0	Output Delay Chain Control - adq_o[8]
7:0	reg_delay_sel_o_adq9	r/w	8'h0	Output Delay Chain Control - adq_o[9]
-1:32	RSVD			
31:24	reg_delay_sel_o_adq10	r/w	8'h0	Output Delay Chain Control - adq_o[10]
23:16	reg_delay_sel_o_adq11	r/w	8'h0	Output Delay Chain Control - adq_o[11]
15:8	reg_delay_sel_o_adq12	r/w	8'h0	Output Delay Chain Control - adq_o[12]
7:0	reg_delay_sel_o_adq13	r/w	8'h0	Output Delay Chain Control - adq_o[13]
-1:32	RSVD			
31:24	reg_delay_sel_o_adq14	r/w	8'h0	Output Delay Chain Control - adq_o[14]
23:16	reg_delay_sel_o_adq15	r/w	8'h0	Output Delay Chain Control - adq_o[15]
15:8	reg_delay_sel_i_adq8	r/w	8'h0	Input Delay Chain Control - adq_i[8]
7:0	reg_delay_sel_i_adq9	r/w	8'h0	Input Delay Chain Control - adq_i[9]
-1:32	RSVD			
31:24	reg_delay_sel_i_adq10	r/w	8'h0	Input Delay Chain Control - adq_i[10]
23:16	reg_delay_sel_i_adq11	r/w	8'h0	Input Delay Chain Control - adq_i[11]
15:8	reg_delay_sel_i_adq12	r/w	8'h0	Input Delay Chain Control - adq_i[12]
7:0	reg_delay_sel_i_adq13	r/w	8'h0	Input Delay Chain Control - adq_i[13]

Bits	Name	Type	Reset	Description
-1:32	RSVD			
31:24	reg_delay_sel_i_adq14	r/w	8'h0	Input Delay Chain Control - adq_i[14]
23:16	reg_delay_sel_i_adq15	r/w	8'h0	Input Delay Chain Control - adq_i[15]
15:0	reg_delay_sel_i_dqs1	r/w	16'h0	Input Delay Chain Control - dqs_i[1]
-1:16	RSVD			
15:8	reg_delay_sel_o_dqs_oen1	r/w	8'h0	Output Delay Chain Control - dqs_oen[1]
7:0	reg_delay_sel_o_dqs_mask	r/w	8'h0	Output Delay Chain Control - adq_mask
-1:8	RSVD			
7:4	reg_psram_dbg_sel	r/w	4'd0	pSRAM debug selection
3:1	RSVD			
0	reg_psram_dbg_en	r/w	1'b0	pSRAM debug enable
-1:32	RSVD			
31:0	reg_psram_dummy_reg	r/w	32'hFFFF0000	pSRAM Dummy register
-1:28	RSVD			
27:16	reg_timeout_cnt	r/w	12'h100	Timeout threshold
15:3	RSVD			
2	sts_timeout	r	1'h0	Timeout status
1	reg_timeout_clr	r/w	1'h0	Timeout status clear
0	reg_timeout_en	r/w	1'h0	Timeout enable
-1:32	RSVD			
31:24	reg_rough_sel_o_clk	r/w	8'h0	Output Delay Chain Control - clk
23:16	reg_rough_sel_o_clk_n	r/w	8'h0	Output Delay Chain Control - clk_n
15:8	reg_rough_sel_o_ceb	r/w	8'h0	Output Delay Chain Control - ceb
7:0	reg_rough_sel_o_dqs_oen0	r/w	8'h0	Output Delay Chain Control - dqs_oen[0]
-1:32	RSVD			
31:24	reg_rough_sel_o_dqs0	r/w	8'h0	Output Delay Chain Control - dqs_o[0]
23:16	reg_rough_sel_o_adq_oen0	r/w	8'h0	Output Delay Chain Control - adq_oen[0]
15:8	reg_rough_sel_o_adq0	r/w	8'h0	Output Delay Chain Control - adq_o[0]
7:0	reg_rough_sel_o_adq1	r/w	8'h0	Output Delay Chain Control - adq_o[1]
-1:32	RSVD			
31:24	reg_rough_sel_o_adq2	r/w	8'h0	Output Delay Chain Control - adq_o[2]

Bits	Name	Type	Reset	Description
23:16	reg_rough_sel_o_adq3	r/w	8'h0	Output Delay Chain Control - adq_o[3]
15:8	reg_rough_sel_o_adq4	r/w	8'h0	Output Delay Chain Control - adq_o[4]
7:0	reg_rough_sel_o_adq5	r/w	8'h0	Output Delay Chain Control - adq_o[5]
-1:32	RSVD			
31:24	reg_rough_sel_o_adq6	r/w	8'h0	Output Delay Chain Control - adq_o[6]
23:16	reg_rough_sel_o_adq7	r/w	8'h0	Output Delay Chain Control - adq_o[7]
15:8	reg_rough_sel_i_adq0	r/w	8'h0	Input Delay Chain Control - adq_i[0]
7:0	reg_rough_sel_i_adq1	r/w	8'h0	Input Delay Chain Control - adq_i[1]
-1:32	RSVD			
31:24	reg_rough_sel_i_adq2	r/w	8'h0	Input Delay Chain Control - adq_i[2]
23:16	reg_rough_sel_i_adq3	r/w	8'h0	Input Delay Chain Control - adq_i[3]
15:8	reg_rough_sel_i_adq4	r/w	8'h0	Input Delay Chain Control - adq_i[4]
7:0	reg_rough_sel_i_adq5	r/w	8'h0	Input Delay Chain Control - adq_i[5]
-1:32	RSVD			
31:24	reg_rough_sel_i_adq6	r/w	8'h0	Input Delay Chain Control - adq_i[6]
23:16	reg_rough_sel_i_adq7	r/w	8'h0	Input Delay Chain Control - adq_i[7]
15:0	reg_rough_sel_i_dqs0	r/w	16'h0	Input Delay Chain Control - dqs_i[0]
-1:32	RSVD			
31:24	reg_rough_sel_o_dqs1	r/w	8'h0	Output Delay Chain Control - dqs_o[1]
23:16	reg_rough_sel_o_adq_oen1	r/w	8'h0	Output Delay Chain Control - adq_oen[1]
15:8	reg_rough_sel_o_adq8	r/w	8'h0	Output Delay Chain Control - adq_o[8]
7:0	reg_rough_sel_o_adq9	r/w	8'h0	Output Delay Chain Control - adq_o[9]
-1:32	RSVD			
31:24	reg_rough_sel_o_adq10	r/w	8'h0	Output Delay Chain Control - adq_o[10]
23:16	reg_rough_sel_o_adq11	r/w	8'h0	Output Delay Chain Control - adq_o[11]
15:8	reg_rough_sel_o_adq12	r/w	8'h0	Output Delay Chain Control - adq_o[12]
7:0	reg_rough_sel_o_adq13	r/w	8'h0	Output Delay Chain Control - adq_o[13]
-1:32	RSVD			
31:24	reg_rough_sel_o_adq14	r/w	8'h0	Output Delay Chain Control - adq_o[14]
23:16	reg_rough_sel_o_adq15	r/w	8'h0	Output Delay Chain Control - adq_o[15]
15:8	reg_rough_sel_i_adq8	r/w	8'h0	Input Delay Chain Control - adq_i[8]

Bits	Name	Type	Reset	Description
7:0	reg_rough_sel_i_adq9	r/w	8'h0	Input Delay Chain Control - adq_i[9]
-1:32	RSVD			
31:24	reg_rough_sel_i_adq10	r/w	8'h0	Input Delay Chain Control - adq_i[10]
23:16	reg_rough_sel_i_adq11	r/w	8'h0	Input Delay Chain Control - adq_i[11]
15:8	reg_rough_sel_i_adq12	r/w	8'h0	Input Delay Chain Control - adq_i[12]
7:0	reg_rough_sel_i_adq13	r/w	8'h0	Input Delay Chain Control - adq_i[13]
-1:32	RSVD			
31:24	reg_rough_sel_i_adq14	r/w	8'h0	Input Delay Chain Control - adq_i[14]
23:16	reg_rough_sel_i_adq15	r/w	8'h0	Input Delay Chain Control - adq_i[15]
15:0	reg_rough_sel_i_dqs1	r/w	16'h0	Input Delay Chain Control - dqs_i[1]
-1:16	RSVD			
15:8	reg_rough_sel_o_dqs_oen1	r/w	8'h0	Output Delay Chain Control - dqs_oen[1]
7:0	reg_rough_sel_o_dqs_mask	r/w	8'h0	Output Delay Chain Control - adq_mask

24 Emac

24.1 Overview

The EMAC module is a 10/100Mbps 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.

24.2 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 10 Mbps and 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

24.3 Functional Description

The composition of EMAC module is as follows.

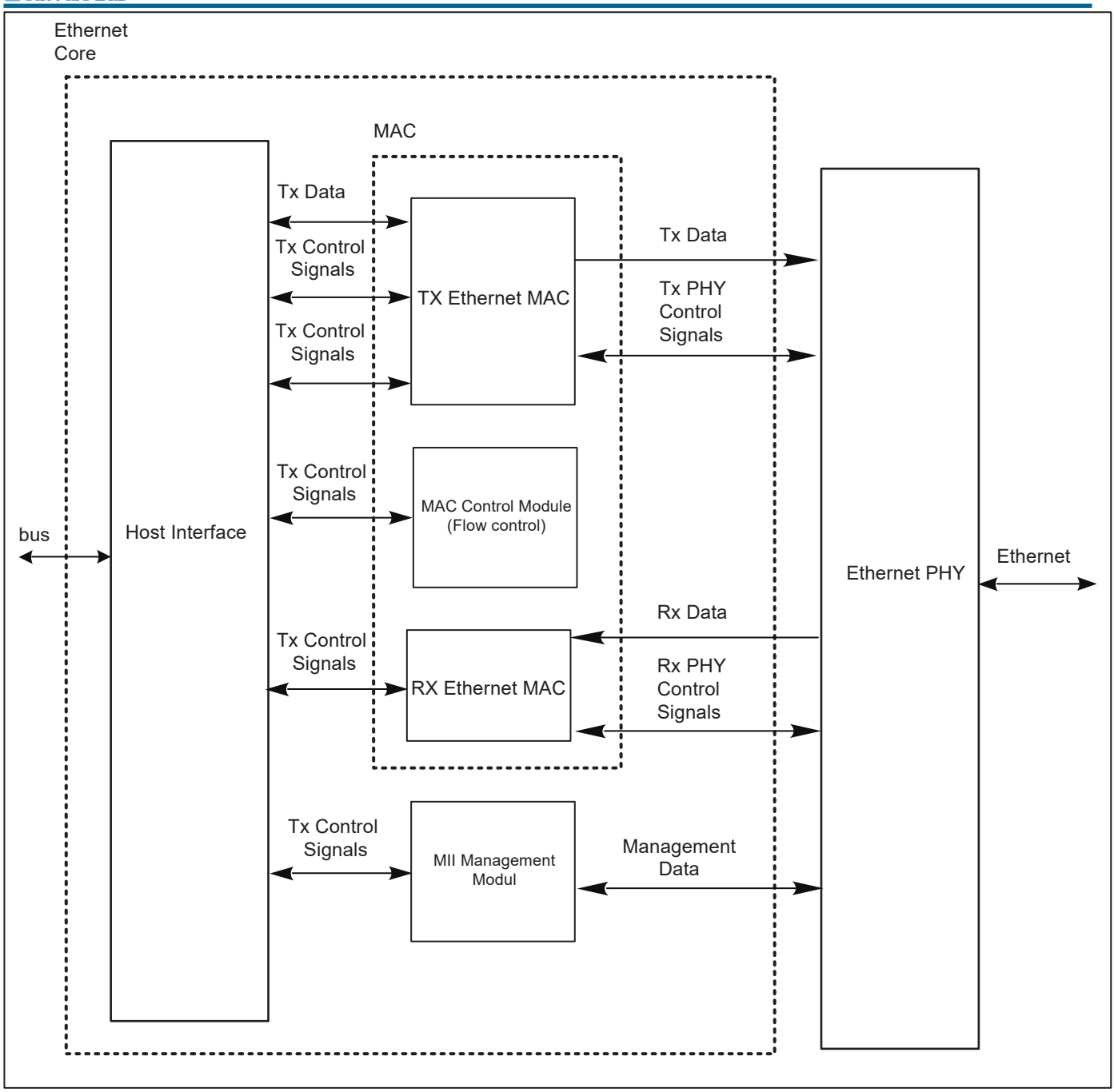


Fig. 24.1: Block diagram of EMAC

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.

24.4 Clock

EMAC needs a clock for synchronous transmission and reception (25 MHz (MII) or 50 MHz (RMII) at 100 Mbps, and 2.5 MHz at 10 Mbps). The clock must be synchronized between EMAC and PHY.

24.5 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 word with low address provides the length, control bits, and status bits of the data frames contained in this buffer. The word with high address is the memory pointer. More details of BD are presented in the register description section. 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.

24.6 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. Both MII and RMII modes support the transmission rates of 10 Mbps and 100 Mbps specified in IEEE 802.3u. The transmission signals of MII and RMII are described as follows.

Table 24.1: 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.

24.7 Programming Flow

24.7.1 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.

24.7.2 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

24.7.3 Receive Data Frame

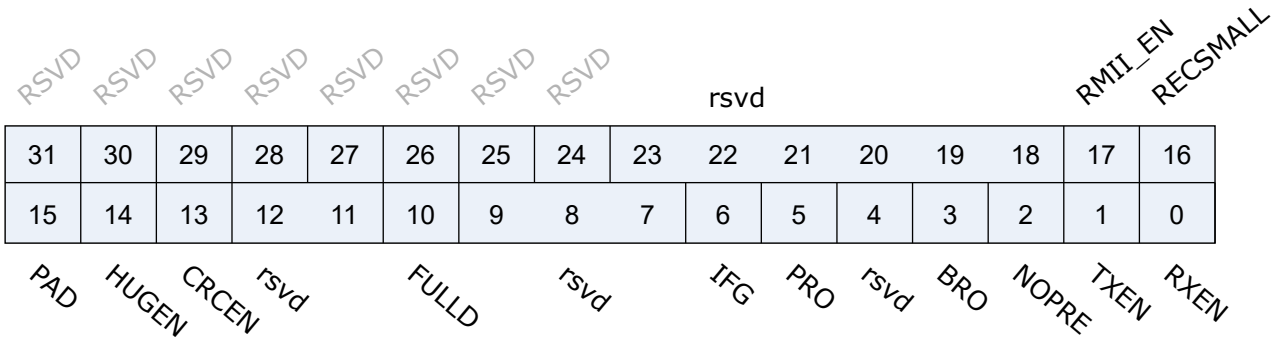
- Configure data frame format and interval bit fields in the register EMAC_MODE
- Specify the number of TX BDs by setting the TXBDNUM field in the register EMAC_TX_BD_NUM, so that the rest is the number of RX BDs
- Prepare an area in memory for receiving data
- Fill in the address of data frames in the 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

24.8 Register description

Name	Description
MODE	
INT_SOURCE	
INT_MASK	
IPGT	
PACKETLEN	
COLLCONFIG	
TX_BD_NUM	
MIIMODE	
MIICOMMAND	
MIIADDRESS	
MIITX_DATA	
MIIRX_DATA	
MIISTATUS	
MAC_ADDR0	
MAC_ADDR1	
HASH0_ADDR	
HASH1_ADDR	
TXCTRL	

24.8.1 MODE

Address：0x20070000



Bits	Name	Type	Reset	Description
31:24	RSVD			
23:18	rsvd	rsvd	6'h0	Reserved
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	rsvd	2'b0	Reserved
10	FULLD	r/w	1'b0	Full duplex 0: Half duplex mode. 1: Full duplex mode.
9:7	rsvd	rsvd	3'b0	Reserved
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.

Bits	Name	Type	Reset	Description
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.
4	rsvd	rsvd	1'b0	Reserved
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.

24.8.2 INT_SOURCE

Address：0x20070004

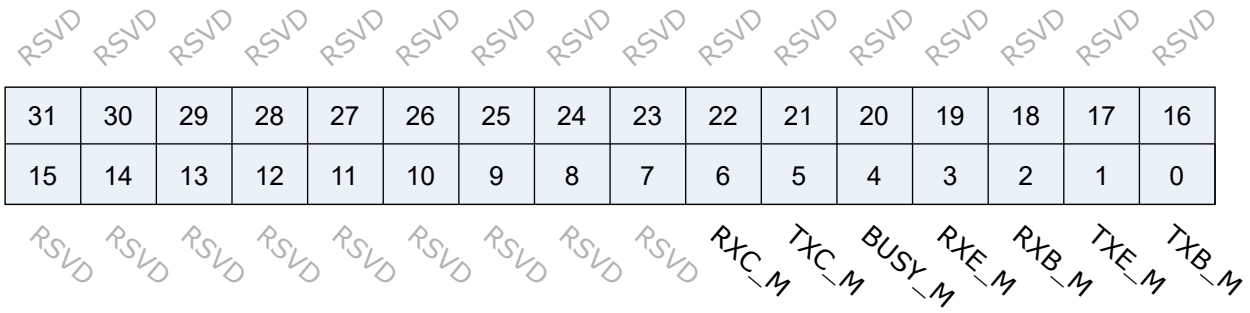
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RXC	TXC	BUSY	RXE	RXB	TXE	TXB

Bits	Name	Type	Reset	Description
31:7	RSVD			

Bits	Name	Type	Reset	Description
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.
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.

24.8.3 INT_MASK

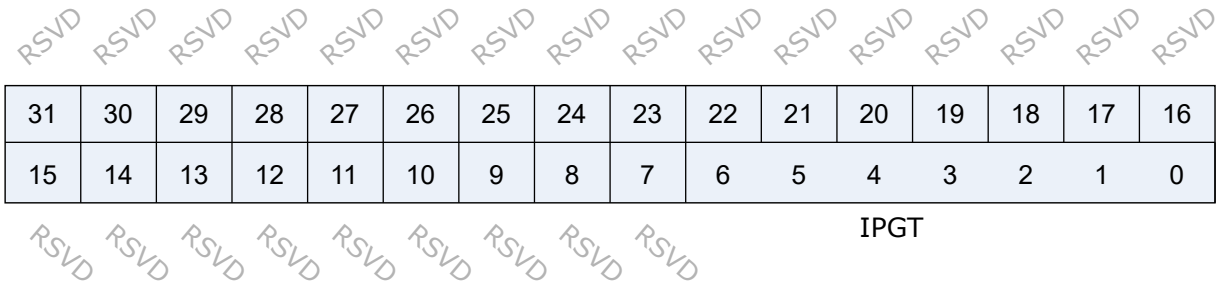
Address：0x20070008



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

24.8.4 IPGT

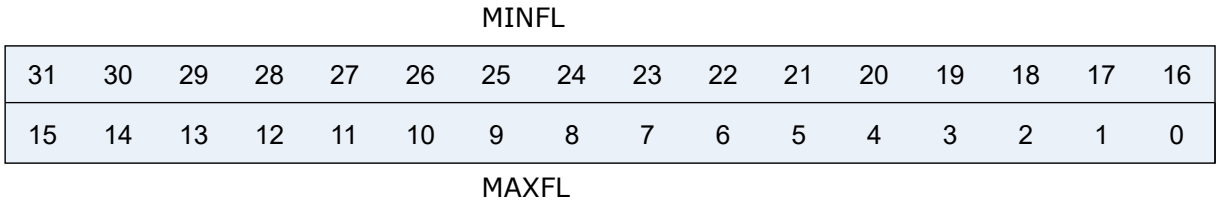
Address：0x2007000c



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 9.6 us for 10 Mbps and 0.96 us for 100 Mbps mode

24.8.5 PACKETLEN

Address：0x20070018

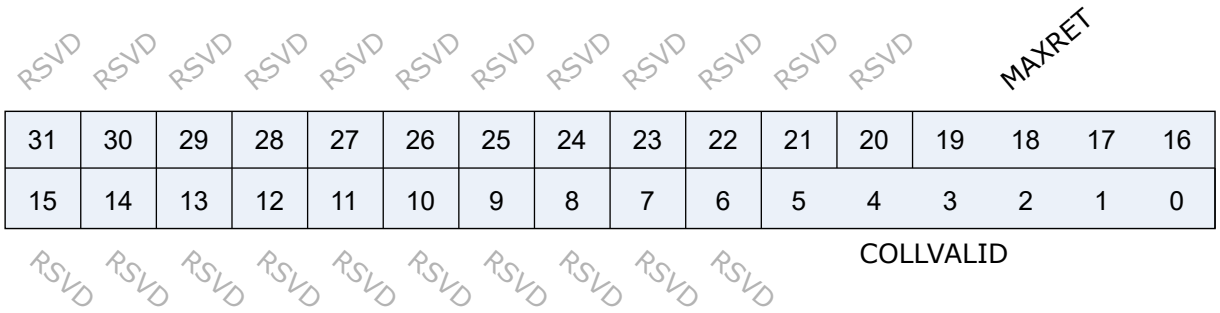


Bits	Name	Type	Reset	Description
31:16	MINFL	r/w	16'h40	Minimum frame length The minimum Ethernet packet is 64 bytes long (0x40). To receive small packets, assert the RECSMALL bit or change the MINFL value. To transmit small packets, assert the PAD bit or change the MINFL value.

Bits	Name	Type	Reset	Description
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.

24.8.6 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.

24.8.7 TX_BD_NUM

Address：0x20070020

RSVD RXBDPTR RSVD TXBDPTR

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 TXBDNUM

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.

24.8.8 MIIMODE

Address：0x20070028

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

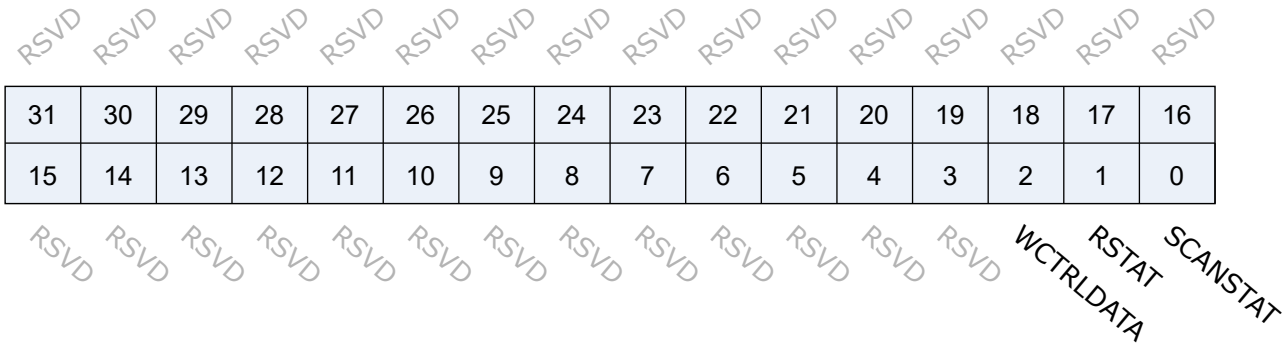
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RSVD RSVD RSVD RSVD RSVD RSVD RSVD MIINOPRE CLKDIV

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.

24.8.9 MIICOMMAND

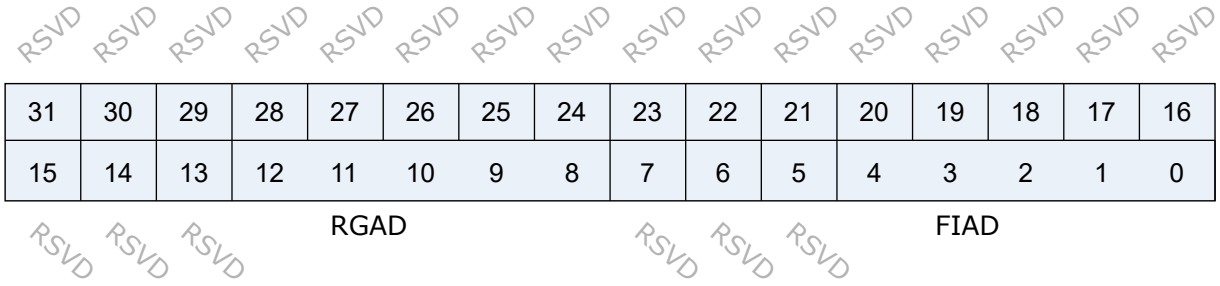
Address：0x2007002c



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

24.8.10 MIIADDRESS

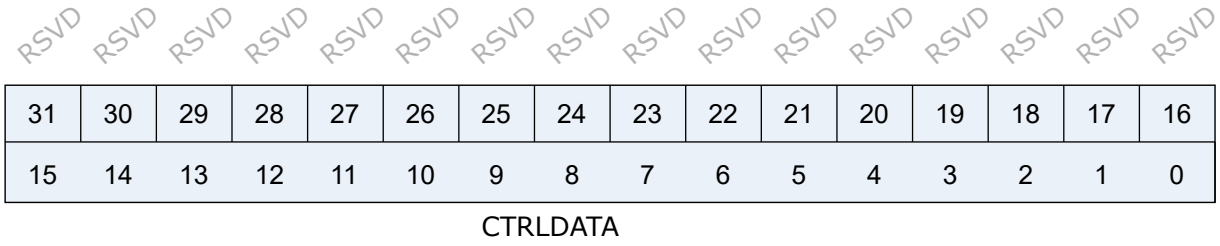
Address：0x20070030



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

24.8.11 MIITX_DATA

Address：0x20070034



Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	CTRLDATA	r/w	16'h0	Control Data to be written to PHY

24.8.12 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

24.8.13 MIISTATUS

Address：0x2007003c

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														MIIM_BUSY	MIIM_LINKFAIL

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

24.8.14 MAC_ADDR0

Address：0x20070040

MAC_B2								MAC_B3							
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_B4								MAC_B5							

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

24.8.15 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

24.8.16 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

24.8.17 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

24.8.18 TXCTRL

Address：0x20070050

RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD RSVD TXPAUSERQ

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

TXPAUSETV

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.

25 USB

25.1 Overview

Universal Serial Bus (USB) is an external bus standard that establishes specifications for the connection and communication between computers and external devices. xx supports USB 2.0 (HighSpeed + FullSpeed) and can be used as a host controller (Host) or a peripheral controller (Device). As a Host, it contains a USB Host that supports all-speed transactions. Without software intervention, the Host can handle the transaction-based data structure by itself to reduce the load of CPU, and automatically send and receive data on the USB bus. As a Device, each endpoint except endpoint 0 supports the transfer types in the USB specification to adapt to various applications. In addition, xx’s USB controller complies with the OTG standard and supports the session request protocol (SRP) and host negotiation protocol (HNP).

25.2 Features

- Compatible with USB 2.0 (HighSpeed + FullSpeed)
- Compatible with OTG revision1.3
- UTMI + Level 3
- OTG SRP and HNP protocols
- Host, OTG, and Device modes
- Supports LPM
- Compatible with EHCI data structure (FSTN and SITD unsupported)
- Nine endpoints (EP0: control endpoint EP1EP8: interrupt/isolation/bulk endpoint)
- One dedicated FIFO for control transmission and four general FIFOs (control FIFO: 64 bytes; general FIFO: 512 bytes)
- Supports 2-FIFO and 3-FIFO modes

- Supports DMA
- Supports VDMA

25.3 Functional Description

25.3.1 Operating Steps of USB

1. Configure internal transceiver
2. Configure USB controller
3. Configure USB interrupt
4. Configure USB DMA/VDMA (direct reading and writing of FIFO by CPU is unsupported)
5. Finish

25.3.2 Partial Register Configuration and Functional Description

25.3.3 Interrupt Processing Flow for USB Enumeration Phase

26 ISO11898

26.1 Overview

ISO11898, an international standard, is one of the most widely used fieldbus in the industrial and automotive fields.

26.2 Features

- Custom bit rate
- ISO11898 2.0A and 2.0B
- Self-test mode (self-sending and self-receiving)
- Frame filtering
- Silent mode (no reply, no valid error flag)
- Single transmission without retransmission
- Query of bits that lose the arbitration
- Any bus error can trigger an interrupt

26.3 Functional Description

26.3.1 TX Buffer (TXB)

TXB, an interface between CPU and BSP, can store complete messages for transmission over the ISO11898 network. The buffer length is 13 bytes, written by CPU and read by the BSP.

26.3.2 RX Buffer (RXB, RXFIFO)

RXB, an interface between access control filter and CPU, is used to store the filtered messages received from the ISO11898 bus. RXB represents a 13-byte window in the RX FIFO that can be accessed by CPU, with a total length of 64 bytes. RX FIFO allows CPU to receive other messages while processing one frame of message.

26.3.3 Access Control Filter (ACF)

ACF compares the received identifier with the contents of the ACF register and decides whether the message should be accepted. If accepted, the complete message will be stored in RX FIFO.

26.3.4 Bit Stream Processor (BSP)

BSP, a sequence generator, is used to process the data between TXB, RXB, and ISO11898 bus. It also performs error detection, arbitration, bit padding, and error handling on the ISO11898 bus.

26.3.5 Bit Timing Logic (BTL)

BTL monitors the serial ISO11898 bus and processes the bus related bit timing. It performs hard synchronization on the recessive-to-dominant transition of the bus at the beginning of the message, and soft synchronization again during subsequent message reception. BTL also provides a programmable time period to compensate propagation delay and phase shift (for example, due to oscillator drift), and can define sampling points and sampling times in a bit time.

26.3.6 Error Management Logic (EML)

EML determines the errors of the transport layer module. It receives the error statement from BSP and then informs BSP and interrupt management logic (IML) of error statistics.

26.4 Functional Description

26.4.1 Mode

26.4.1.1 Self-test Mode

You can select the self-test mode by setting the STM bit in the MOD register to ‘1’. In this mode, the RX request command can be used to conduct a full-node test without other active nodes on the bus, and even if no response is received, the ISO11898 controller will perform successful transmission.

26.4.1.2 Silent Mode

You can select the silent mode by setting the LOM bit in the MOD register to “1”. In this mode, the ISO11898 controller will not respond to the ISO11898 bus even if it successfully receives the message, and the error counter will stay at the current value. This mode will force the ISO11898 controller to become a passive error, and no message can be transmitted at this time. This mode can be used in software-driven bit rate detection and hot swap scenarios, and all other functions work normally as the normal mode.

26.4.1.3 Reset Mode

Once the RM bit in the MOD register changes from ‘0’ to ‘1’, it will cause the current TX and RX messages to be terminated and enter the reset mode. When the RM bit changes from ‘1’ to ‘0’, the ISO11898 controller will return to the operating mode.

The meanings of different operations in different modes are as follows.

Table 26.1: The meaning of each register in different modes

ADDRESS OFFSET	OPERATING MODE				RESET MODE	
	READ		WRITE		READ	WRITE
0x00	mode		mode		mode	mode
0x04	(00H)		command		(00H)	command
0x08	status		reserved		status	reserved
0x0C	interrupt		reserved		interrupt	reserved
0x10	interrupt enable		interrupt enable		interrupt enable	interrupt enable
0x14	reserved		reserved		reserved	reserved
0x18	bus timing 0		reserved		bus timing 0	bus timing 0
0x1C	bus timing 1		reserved		bus timing 1	bus timing 1
0x20	reserved		reserved		reserved	reserved
0x24	reserved		reserved		reserved	reserved
0x28	reserved		reserved		reserved	reserved
0x2C	arbitration lost capture		reserved		arbitration lost capture	reserved
0x30	error code capture		reserved		error code capture	reserved
0x34	error warning limit		reserved		error warning limit	error warning limit
0x38	RX error counter		reserved		RX error counter	RX error counter
0x3C	TX error counter		reserved		TX error counter	TX error counter
0x40	SFF RX frame information	EFF RX frame information	SFF TX frame information	EFF TX frame information	acceptance code 0	acceptance code 0

Table 26.1: The meaning of each register in different modes

ADDRESS OFFSET	OPERATING MODE				RESET MODE	
	READ		WRITE		READ	WRITE
0x44	RX identifier 1	RX identifier 1	TX identifier 1	TX identifier 1	acceptance code 1	acceptance code 1
0x48	RX identifier 2	RX identifier 2	TX identifier 2	TX identifier 2	acceptance code 2	acceptance code 2
0x4C	RX data 1	RX identifier 3	TX data 1	TX identifier 3	acceptance code 3	acceptance code 3
0x50	RX data 2	RX identifier 4	TX data 2	TX identifier 4	acceptance mask 0	acceptance mask 0
0x54	RX data 3	RX data 1	TX data 3	TX data 1	acceptance mask 1	acceptance mask 1
0x58	RX data 4	RX data 2	TX data 4	TX data 2	acceptance mask 2	acceptance mask 2
0x5C	RX data 5	RX data 3	TX data 5	TX data 3	acceptance mask 3	acceptance mask 3
0x60	RX data 6	RX data 4	TX data 6	TX data 4	reserved	reserved
0x64	RX data 7	RX data 5	TX data 7	TX data 5	reserved	reserved
0x68	RX data 8	RX data 6	TX data 8	TX data 6	reserved	reserved
0x6C	(FIFO RAM)	RX data 7	reserved	TX data 7	reserved	reserved
0x70	(FIFO RAM)	RX data 8	reserved	TX data 8	reserved	reserved
0x74	RX message counter		reserved		RX message counter	reserved
0x78	RX buffer start address		reserved		RX buffer start address	RX buffer start address
0x7C	clock divider		clock divider		clock divider	clock divider

26.4.2 Sending Process

26.4.2.1 Process

1. Check the TBS bit in the SR register to ensure that TXB is empty.
2. Configure frame information, ID number, and data.
3. Request transmission by setting the TR bit in the CMR register.

26.4.2.2 Termination of Sending

When CPU requests to suspend the previous transmission, you can use this function. For example, you need to send a more urgent message first. Messages that are sending now are not affected by this function and sending will not stop. To check whether the previous message was successfully sent, you should check the TCS bit in the SR register. The application software can use this function by setting the AT bit in the CMR register to ‘1’, which should be executed after the TBS bit in the SR register is set to ‘1’ or the TX interrupt is generated.

It should be noted that even if the message is terminated, a TX interrupt will occur, because the status bit of the TXB has indicated the “released” status.

26.4.2.3 Self-sending and Self-receiving

The application software can realize self-sending and self-receiving by setting the SRR bit in the CMR register. At that time, sending and receiving are synchronized. Other operations are the same as the normal sending process.

26.4.2.4 Precautions

1. If the TR and AT bits of the CMA register are set simultaneously, the message will be sent only once. Even if there is an error event or arbitration lost, it will not be sent again.
2. If the SRR and AT bits of the CMA register are set simultaneously, the message will be sent only once by self-sending and self-receiving. Even if there is an error event or arbitration lost, it will not be sent again.
3. If SRR, TR, and AT bits of the CMA register are set simultaneously, the message will be sent by setting TR and AT bits simultaneously.
4. Once the TX status bit in the status register is set, the internal TX request bit will be cleared automatically.
5. If the TR and SRR bits of the CMA register are set simultaneously, the SRR bit will be ignored.

26.4.3 Receiving Process

26.4.3.1 Process

The received messages are stored in an internal FIFO with a depth of 64 bytes. The FIFO is completely managed by hardware, which saves CPU’s processing load, simplifies software, and ensures data consistency. The application can read the received messages through the FIFO’s output interface. When the RBS bit in the SR register is set, one or more frames of messages can be read in RX FIFO. After the software gets the message, setting the RRB bit in the CMR register can release the RX FIFO occupied by the current message.

26.4.3.2 Number of Messages

The RMC register indicates the number of readable messages in RX FIFO, which increases with each RX event and decreases with each buffer release. The value is 0 after reset.

26.4.3.3 RXB

The RBSA register indicates the address of the first byte of the received message stored in the current internal RAM, which is mapped to the RXB window. The contents of the internal RAM can be interpreted on this basis. This part of the internal RAM can be read and written by CPU (written only in the reset mode).

Example: If the value of RBSA is 18H, the current readable message of the RXB window (offset address: 10H to 12H) is also stored in the RAM address starting from 18H. As the RAM address is directly mapped to the starting position of ISO11898 offset address 20H (corresponding to RAM address 0H), the message can also be read from ISO11898 offset address 38H and the following bytes (ISO11898 address = RBSA + 20H = 18H + 20H = 38H). If the message address exceeds the RAM address 3FH, it will continue from the RAM address 0.

When there is at least one message in FIFO, the command to release the RXB should be issued, and then RBSA will be updated to the starting position of the next message.

When the hardware is reset, the value of RBSA register is initialized to ‘00H’. When the software is reset in the reset mode, the value of this register will not change, but the FIFO will be cleared. This means that the contents of RAM will not change, but the next received (or sent) message will overwrite the visible message in the RXB window.

26.4.4 Identifier Filtering

With the help of ACF, the ISO11898 controller will allow the received message to be delivered to RX FIFO only when the identifier bit of the received message is the same as the predefined bit in the ACF register. ACF consists of the acceptance code registers (ACRn) and acceptance mask registers (AMRn). The values of matching bits in the receivable message are set by the ACRn, and which bits can be masked is set by the AMRn.

There are two different filtering modes (set by the AFM bit in the MOD register):

- Single filter mode (AFM = 1).
- Double filter mode (AFM = 0).

26.4.4.1 Single Filter Configuration

In this configuration, a 4-byte long filter can be defined. The bit correspondence between filter bytes and message bytes depends on the currently received frame format.

Standard frame: If a message in a standard frame format is received, the complete identifier including the RTR bit and the first two data bytes is used to accept filtering. If there is no data byte because RTR bit is set, or there is no data byte or only one data byte because a data length is set, the message can also be received.

As all the filter bits are logically AND, only when all the bits pass through the filter, can a message be received. It should be noted that the low 4 bits of AMR1 and ACR1 are unused, and these bits should be set as mask bits for compatibility with future products. That is, all 3-0 bits of AMR1 are ‘1’.

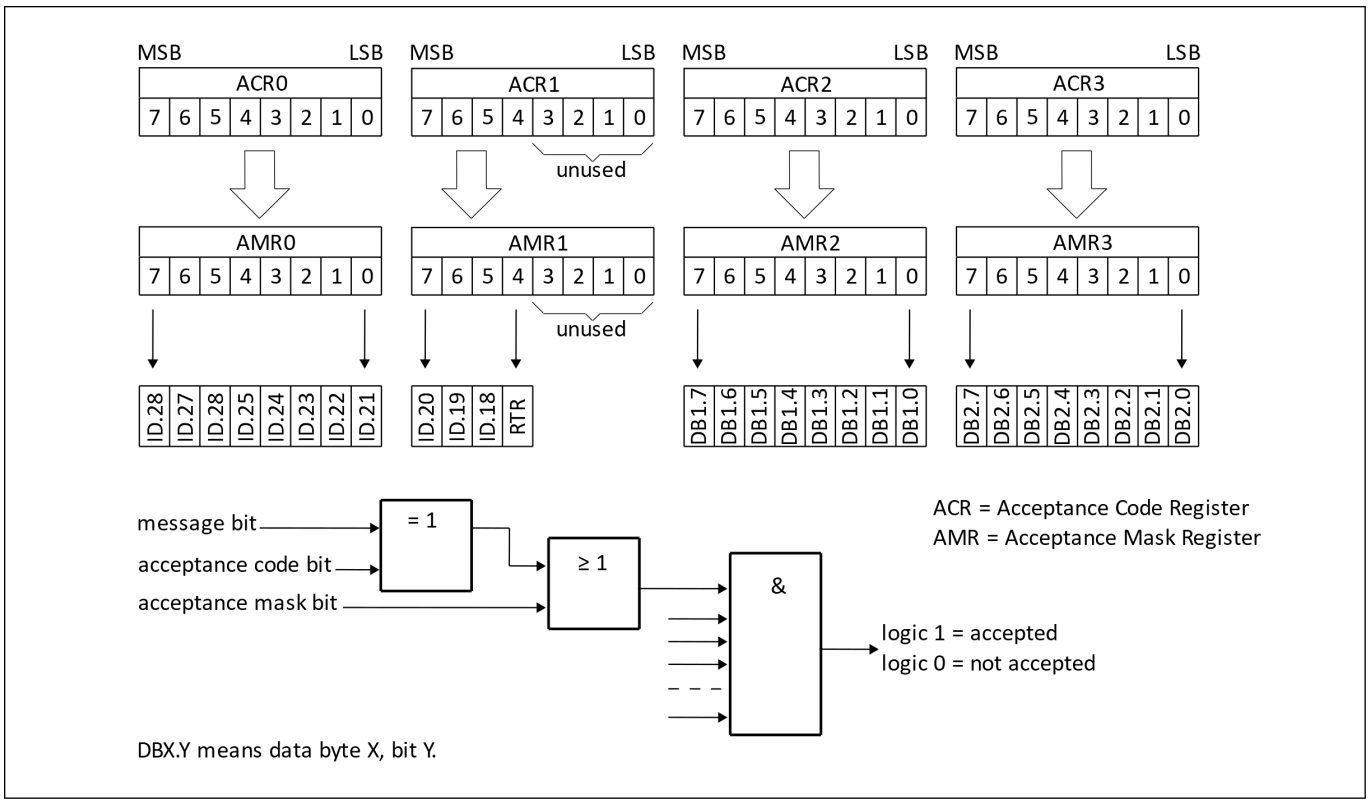


Fig. 26.1: Single filter configuration, receiving standard frame messages

Extended frame: If a message in an extended frame format is received, the complete identifier including the RTR bit is used to accept filtering.

As all the filter bits are logically AND, only when all the bits pass through the filter, can a message be received. It should be noted that the low 2 bits of AMR3 and ACR3 are unused, and these bits should be set as mask bits for compatibility with future products. That is, all 1-0 bits of AMR3 are ‘1’.

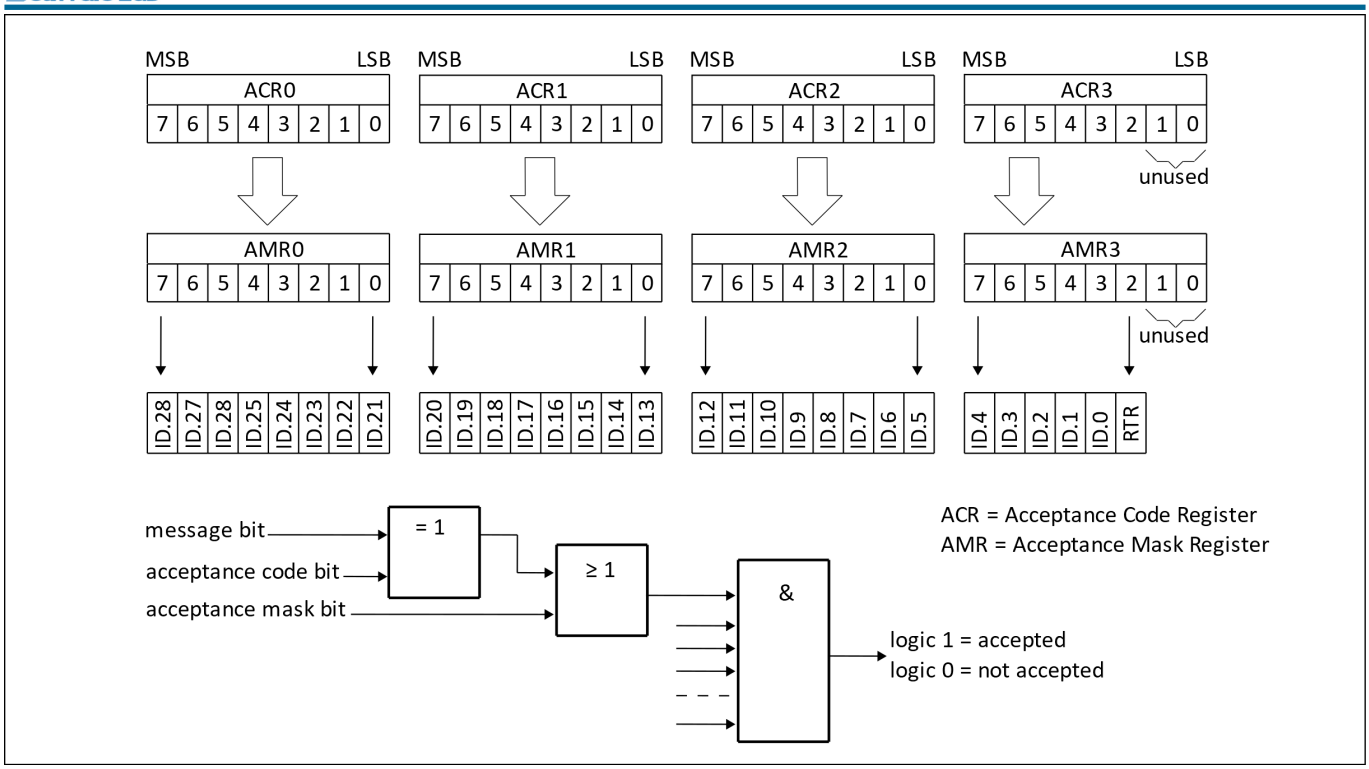


Fig. 26.2: Single filter configuration, receiving extended frame messages

26.4.4.2 Double Filter Configuration

Two short filters can be defined in this configuration, and the received message will be compared with both filters to decide whether to copy the message to the RXB. As long as a filter receives the message, the received message is valid. The bit correspondence between filter bytes and message bytes depends on the currently received frame format.

Standard frame: If a message in a standard frame format is received, the two filters defined look a little different. The first filter compares the complete identifier including RTR and the first data byte, while the second one only compares the standard identifier including RTR.

To successfully receive the message, the comparison result of all single bits in at least one complete filter indicates “accept”. There is no data when RTR is set or the data length is 0. However, if the first part up to the RTR bit indicates “accept”, the message can also pass through the filter 1.

If the first filter does not need to filter data bytes, the low 4 bits of AMR1 and AMR3 must be set to logical ‘1’ (insignificant), and the two filters run identically using standard identifiers including RTR.

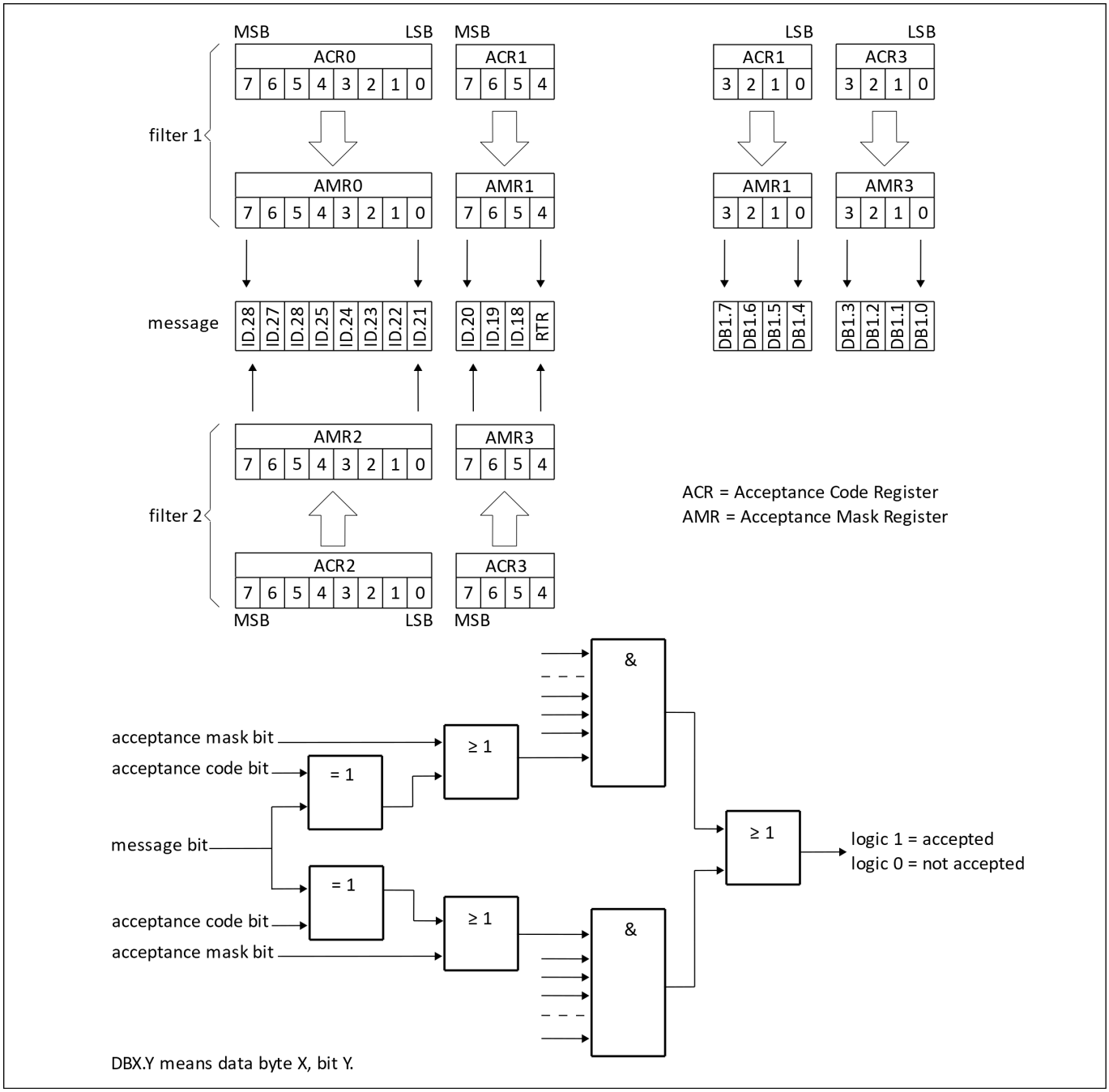


Fig. 26.3: Dual filter configuration, receiving standard frame messages

Extended frame: If a message in an extended frame format is received, the two filters defined look the same. Both filters only compare the first two bytes of the extended identifier.

Only when all single bit comparisons of at least one complete filter indicate acceptance, can the message be successfully received.

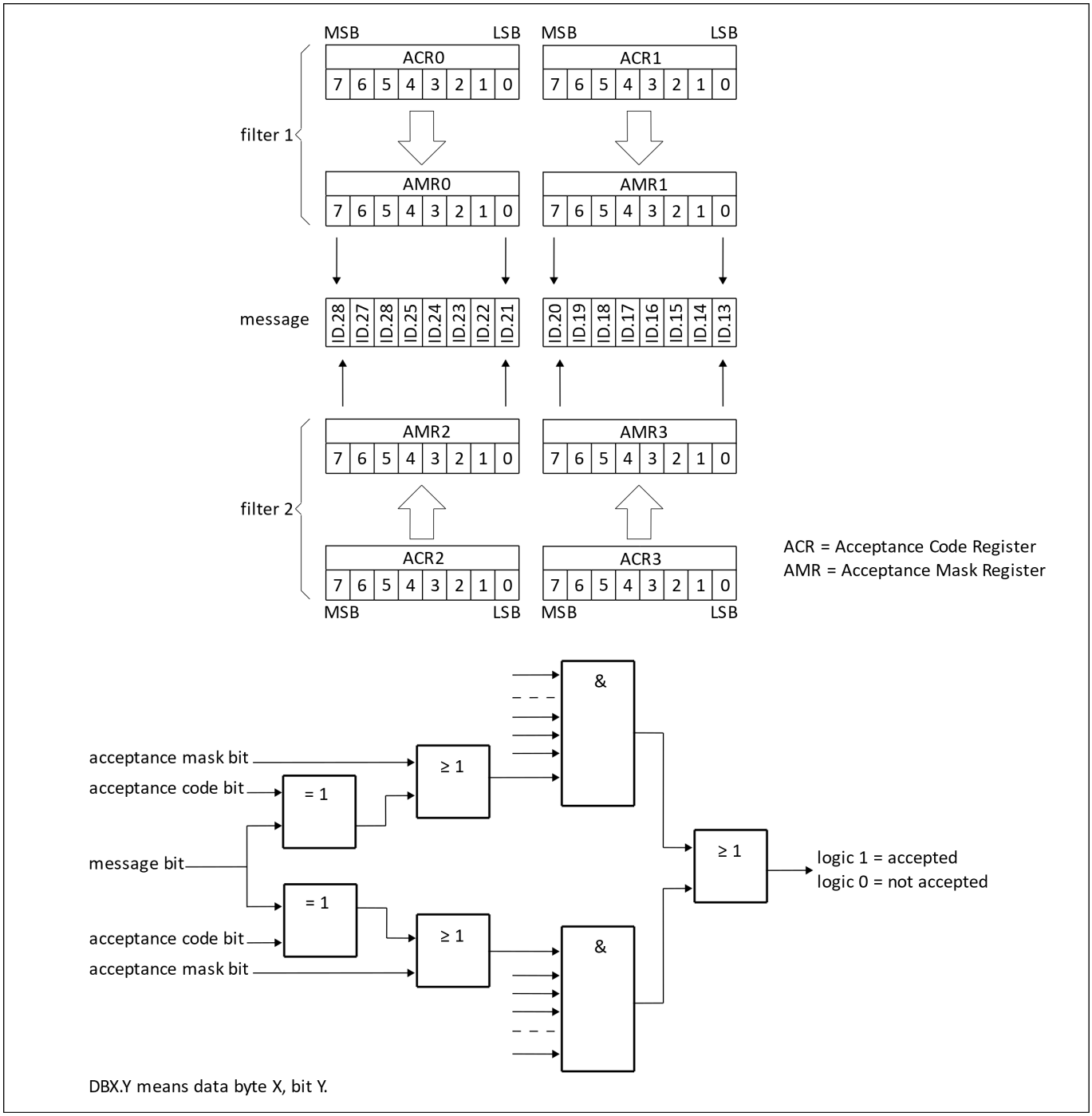


Fig. 26.4: Dual filter configuration, receiving extended frame messages

26.4.5 Error Management

26.4.5.1 Arbitration Lost

The arbitration lost capture (ALC) register contains the position that encounters arbitration lost and can only be read by CPU but not written by it. If the arbitration lost interrupt is enabled, an interrupt will be generated once arbitration loses. The position of the current bit in BSP is captured into the ALC. This register's value is fixed until the user software reads the contents of the ALC. After this value is read, the capture mechanism is activated again. When the interrupt register is read, the corresponding interrupt flag will also be cleared. No arbitration lost interrupt will be generated again before the ALC register is read.

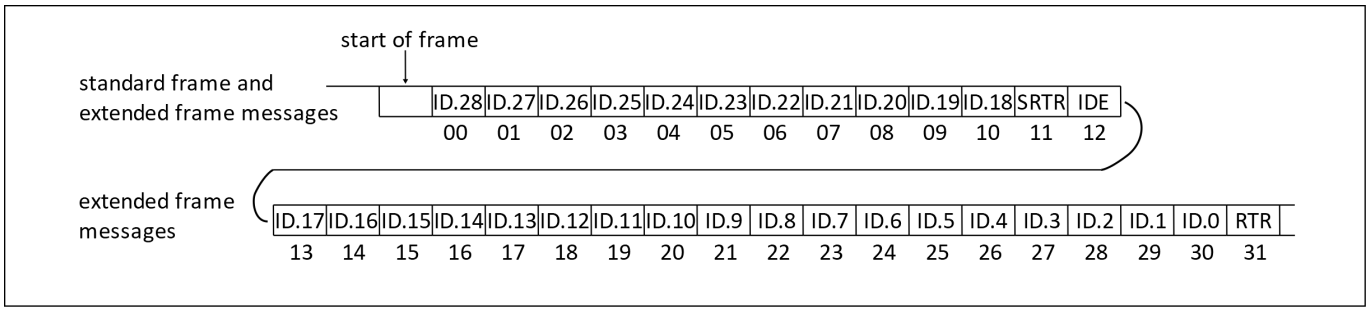


Fig. 26.5: Arbitration lost bit number interpretation

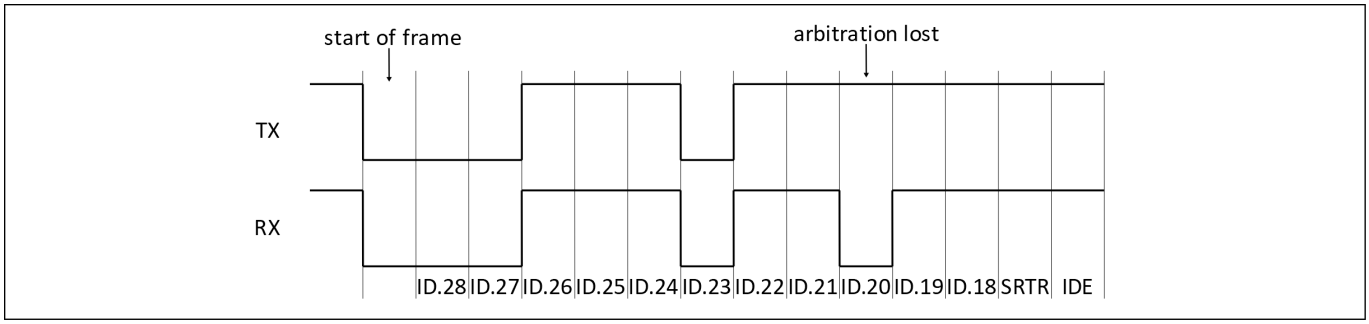


Fig. 26.6: Example of arbitration lost bit number interpretation; result: ALC = 08

Table 26.2: Arbitration loss capture location

BITS					DECIMAL VALUE	FUNCTION
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	0	0	0	00	arbitration lost in bit 1 of identifier
0	0	0	0	1	01	arbitration lost in bit 2 of identifier
0	0	0	1	0	02	arbitration lost in bit 3 of identifier
0	0	0	1	1	03	arbitration lost in bit 4 of identifier
0	0	1	0	0	04	arbitration lost in bit 5 of identifier
0	0	1	0	1	05	arbitration lost in bit 6 of identifier

Table 26.2: Arbitration loss capture location

BITS					DECIMAL VALUE	FUNCTION
ALC.4	ALC.3	ALC.2	ALC.1	ALC.0		
0	0	1	1	0	06	arbitration lost in bit 7 of identifier
0	0	1	1	1	07	arbitration lost in bit 8 of identifier
0	1	0	0	0	08	arbitration lost in bit 9 of identifier
0	1	0	0	1	09	arbitration lost in bit 10 of identifier
0	1	0	1	0	10	arbitration lost in bit 11 of identifier
0	1	0	1	1	11	arbitration lost in bit SRTR
0	1	1	0	0	12	arbitration lost in bit IDE
0	1	1	0	1	13	arbitration lost in bit 12 of identifier
0	1	1	1	0	14	arbitration lost in bit 13 of identifier
0	1	1	1	1	15	arbitration lost in bit 14 of identifier
1	0	0	0	0	16	arbitration lost in bit 15 of identifier
1	0	0	0	1	17	arbitration lost in bit 16 of identifier
1	0	0	1	0	18	arbitration lost in bit 17 of identifier
1	0	0	1	1	19	arbitration lost in bit 18 of identifier
1	0	1	0	0	20	arbitration lost in bit 19 of identifier
1	0	1	0	1	21	arbitration lost in bit 20 of identifier
1	0	1	1	0	22	arbitration lost in bit 21 of identifier
1	0	1	1	1	23	arbitration lost in bit 22 of identifier
1	1	0	0	0	24	arbitration lost in bit 23 of identifier
1	1	0	0	1	25	arbitration lost in bit 24 of identifier
1	1	0	1	0	26	arbitration lost in bit 25 of identifier
1	1	0	1	1	27	arbitration lost in bit 26 of identifier
0	1	1	0	0	28	arbitration lost in bit 27 of identifier
1	1	1	0	1	29	arbitration lost in bit 28 of identifier
1	1	1	1	0	30	arbitration lost in bit 29 of identifier
1	1	1	1	1	31	arbitration lost in bit RTR

26.4.5.2 Error Capture

The error code capture (ECC) register contains the type and location of bus errors and can only be read by CPU but not written by it. If the bus error interrupt is enabled, a bus error interrupt will be generated once a bus error occurs. The position of the current bit in BSP is captured into the ECC. This register's value is fixed until the user software reads the contents of the ECC. After this value is read, the capture mechanism is activated again. Reading the corresponding bit in the interrupt register will clear this bit, and no bus error interrupt will be generated before the ECC register is read.

The error types represented by the values in the ECC register are shown as follows.

Table 26.3: Type of error catch

BIT ECC.7	BIT ECC.6	FUNCTION
0	0	bit error
0	1	form error
1	0	stuff error
1	1	other type of error

Table 26.4: Error catch location

BIT ECC.4	BIT ECC.3	BIT ECC.2	BIT ECC.1	BIT ECC.0	FUNCTION
0	0	0	1	1	start of frame
0	0	0	1	0	ID.28 to ID.21
0	0	1	1	0	ID.20 to ID.18
0	0	1	0	0	bit SRTR
0	0	1	0	1	bit IDE
0	0	1	1	1	ID.17 to ID.13
0	1	1	1	1	ID.12 to ID.5
0	1	1	1	0	ID.4 to ID.0
0	1	1	0	0	bit RTR
0	1	1	0	1	reserved bit 1
0	1	0	0	1	reserved bit 0
0	1	0	1	1	data length code
0	1	0	1	0	data field
0	1	0	0	0	CRC sequence
1	1	0	0	0	CRC delimiter
1	1	0	0	1	acknowledge slot

Table 26.4: Error catch location

BIT ECC.4	BIT ECC.3	BIT ECC.2	BIT ECC.1	BIT ECC.0	FUNCTION
1	1	0	1	1	acknowledge delimiter
1	1	0	1	0	end of frame
1	0	0	1	0	intermission
1	0	0	0	1	active error flag
1	0	1	1	0	passive error flag
1	0	0	1	1	tolerate dominant bits
1	0	1	1	1	error delimiter
1	1	1	0	0	overload flag

26.4.5.3 RX Error Counter Register (RXERR)

The RXERR’s value represents the current number of received errors, and this register is initialized to logical ‘0’ after hardware reset. In the operating mode, this register can only be read by CPU and write in the reset mode. RXERR is set to logical ‘0’ if a bus shutdown event occurs. At this time, the bus is OFF, and the write operation to this register does not work.

It should be noted that CPU can modify the value of RXERR only in the reset mode. In this case, the error state may change, and the error warning interrupt and error passive interrupt will not occur unless the reset mode is exited.

26.4.5.4 TX Error Counter Register (TXERR)

The TXERR’s value represents the current number of send errors. In the operating mode, this register can only be read by CPU and write in the reset mode. This register’s value is initialized to logical ‘0’ after hardware reset. If a bus shutdown event occurs, the TXERR’s value is set to 127, so that the shortest time (128 bus idle signals) defined by the protocol can be calculated. Reading the TXERR’s value during this period can obtain the status information of bus shutdown recovery. If the bus is OFF, a write to TXERR ranging from 0 to 254 will clear the Bus Off flag, and the controller will wait for 11 consecutive recessive bits (Bus Idle) to appear once after clearing the reset mode.

Writing 255 into TXERR by CPU will generate a bus shutdown event. It should be noted that CPU can only change the value of this register by force in the reset mode. In this case, the error or bus state may change, and the error warning interrupt or error passive interrupt will not be affected by the new value unless the reset mode is exited again. After the reset mode is exited, the TXERR’s value still operates as if the bus was shut down due to a bus error, which means that the register will enter the reset mode again, the TXERR’s value is initialized to 127, the RXRERR’s value is initialized to 0, and related statuses and interrupt registers are reset. At this time, exiting the reset mode will execute the bus shutdown recovery process defined by the protocol (waiting for 128 bus idle signals). If the reset

mode is enabled again before the bus is turned off and restored (TXERR>0), the bus will remain OFF and the TXERR's value will be frozen.

26.4.5.5 Error Limit Setting

The error warning limit can be set by the EWLR register, whose default value is 96 (after hardware reset). This register can be read or written by CPU in the reset mode, while it can only be read in the operating mode. When at least one of the two error count values from RXERR and TXERR is greater than or equal to the value set in the EWLR register, the ES bit in the SR register will be set, and otherwise it will be cleared. Then, if the EIE bit in the IER register is set, an error warning interrupt will be generated. It is worth noting that this register can only be operated in the reset mode. The operation of this register may cause the error state to change, and the error warning interrupt will not be generated unless the reset mode is exited again.

26.4.6 Bit Timing

The timing diagram is as follows:

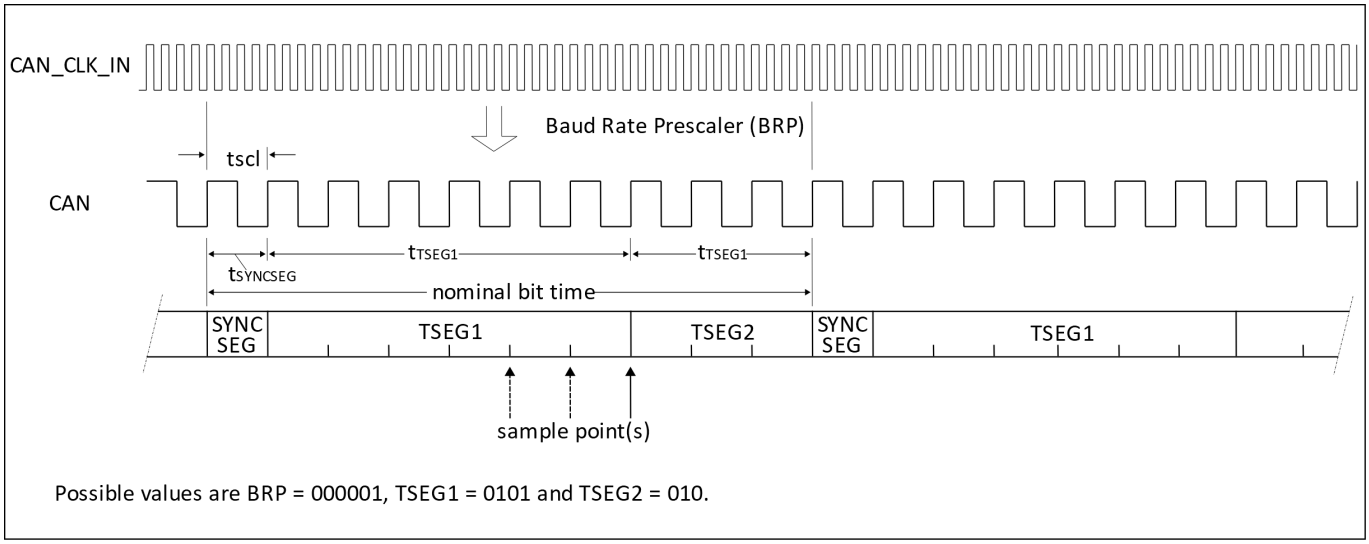


Fig. 26.7: General structure of a bit period

26.4.6.1 Baud Rate Prescaler (BRP)

The cycle of the system clock tscl of the ISO11898 controller can be set, and this determines the timing of each bit. The calculation formula of ISO11898 system clock is as follows:

tscl = 2 * tCLK * (32 * BRP.5 + 16 * BRP.4 + 8 * BRP.3 + 4 * BRP.2 + 2 * BRP.1 + BRP.0 + 1)

26.4.6.2 Synchronization Jump Width (SJW)

To compensate the phase shift between the clock oscillators of different bus controllers, any bus controller must resynchronize at the edge of any related signal currently being transmitted. SJW defines the maximum number of clock cycles that a bit cycle can shorten or extend through one resynchronization:

tSJW = tscl * (2 * SJW.1 + SJW.0 + 1)

26.4.6.3 Sampling (SAM)

When the SAM bit in the BTR1 register is 1, the bus will be sampled three times. This mode suits medium and low speed buses, and this is beneficial to the filter in the bus. If the SAM bit is 0, the bus will only be sampled once. This mode suits the high-speed bus.

26.4.6.4 Time Segment (TSEG)

TSEG, consisting of TSEG1 and TSEG2 in the BTR1 register, determines the number of clocks and sampling point position of each bit, with calculation formula as follows:

tSYNCSEG = 1 * tscl

tTSEG1 = tscl * (8 * TSEG1.3 + 4 * TSEG1.2 + 2 * TSEG1.1 + TSEG1.0 + 1)

tTSEG2 = tscl * (4 * TSEG2.2 + 2 * TSEG2.1 + TSEG2.0 + 1)

27 MJPEG

27.1 Overview

Motion Joint Photographic Experts Group (MJPEG), a video encoding format, can be accurate to frame editing and multi-layer image processing. The motion video sequence is treated as continuous still images, and this compression format compresses each frame separately and completely. Compressing the original data in the YCbCr format can greatly save the memory space occupied by a frame of image.

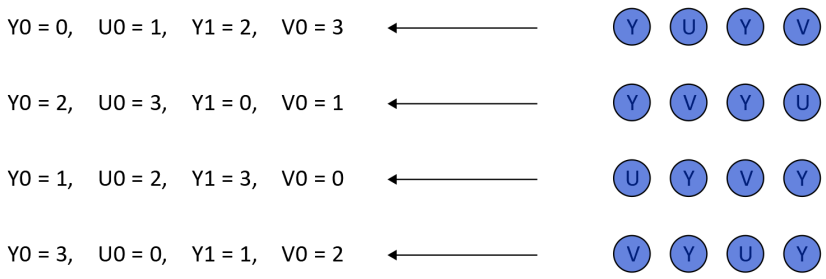
27.2 Features

- Configurable input formats:
  - YCbCr 4:2:2 planar or packed format
  - YCbCr 4:2:0 planar format
  - YCbCr 4:0:0
- Configurable arrangement order of Y, Cb and Cr input data
- Free configuration of quantization coefficient table
- Supports software and linked modes
- Supports swap mode
- Supports kick mode
- Reserves jpg header space and automatically adds jpg tail
- Caches up to 4 images continuously
- Multiple application interrupts are beneficial to flexible use and error prompt

27.3 Functional Description

27.3.1 Input Configuration

You can select the YCbCr format of input data by setting the <REG_YUV_MODE> bit in the register MJPEG_CONTROL_1, including YCbCr 4:2:2 planar or packed format, YCbCr 4:2:0 planar format, and YCbCr 4:0:0 grayscale image. When the YCbCr 4:2:2 packed format is selected, the order of Y, Cb and Cr can be configured by the high 8 bits of the register MJPEG_HEADER_BYTE. When other formats are selected, the order of Cb and Cr can be set by the <REG_ORDER_U_EVEN> bit in the register MJPEG_CONTROL_1. The configuration is shown as follows.



MJPEG YUYV interleave order configuration

27.3.2 Quantization Coefficient Table

The quantization coefficient table can be freely configured by the user. <reg_q_0_00> to <reg_q_0_3F> represents the quantization table of grayscale Y component, while <reg_q_1_00> to <reg_q_1_3F> represents the quantization table of chrominance information Cb and Cr. In the quantization table, the first column is followed by the second column in descending order from the upper left corner. That is, reg_q_0_00 represents the quantized values of the first column on the first row of Y component, reg_q_0_01 represents that of the first column on the second row of the same, reg_q_0_07 represents that of the first column on the eighth row of the same, reg_q_0_08 represents that of the second column on the first row of the same, and so on.

27.3.3 Software Mode and Linked Mode

Setting the <REG_MJPEG_SW_MODE> bit in the register MJPEG_CONTROL_2 to 1 will enable the software mode. In this mode, MJPEG will read the specified frames of original image data from the memory for compression, but the image data must be prepared in advance.

When the <REG_MJPEG_SW_MODE> bit in the register MJPEG_CONTROL_2 is cleared to 0, the linked mode is enabled. In this mode, MJPEG will process the output of CAM as its input by taking 8*8 data blocks as a unit. After CAM writes 8 lines of data into the memory, MJPEG will start to work. The memory space processed by MJPEG will be released to CAM for reuse, so that CAM can link with MJPEG using the storage space that is less than the size of an image.

27.3.4 Swap Mode

In the swap mode, MJPEG’s storage space will be evenly divided into two blocks. When the space of one block is used up, an interrupt will be generated to inform the software to read out the data, and then MJPEG will write data into the other block. Two blocks are used alternatively, so that you can process data with the storage space that is less than the size of a frame of image.

27.3.5 Kick Mode

When the <REG_SW_KICK_MODE> bit in the register MJPEG_CONTROL_2 is set to 1, the kick mode is enabled. In this mode, every time one “1” is written to the <REG_SW_KICK> bit in the register MJPEG_CONTROL_2, one compression will be performed. The number of compressed rows is determined by the <REG_SW_KICK_HBLK> bit in the register MJPEG_YUV_MEM_SW. It should be noted that the kick mode can only be used in the software mode, but not in the linked mode.

27.3.6 Jpg Function

The jpg function will automatically leave some bytes of space at the beginning of each frame of data and fill in zeros. The size of this space is set by the low 12 bits of the register MJPEG_HEADER_BYTE. If automatic padding of jpg tail is enabled, two bytes of data will be added at the end of each frame of data, and the value of the two bytes is 0xFFD9.

27.3.7 Cached Image Information

The module contains four FIFOs to record the image address, image size, and quantization coefficient. Every time this module completely writes a frame into the memory, it will record the start address, image size, and quantization coefficient of this frame of image in FIFO. However, when the memory is insufficient or four FIFOs are full, the module will automatically discard the information of the incoming image. In the part where the image information has been taken out, the oldest image information will be removed by performing the pop operation through APB. Then, FIFO will automatically push data to ensure the timing of the image information in FIFO.

27.3.8 Multiple Interrupts (Separately Configured)

A. Normal interrupt: You can set to generate an interrupt after a fixed number of images are written

B. Camera interrupt: When CAM overflows as MJPEG’s processing speed cannot keep up with the speed of writing in CAM, an interrupt will be generated

C. Memory interrupt: When the memory is rewritten, an interrupt will be generated

D. Frame interrupt: When there are more than 4 unprocessed images, an interrupt will be generated

E. Swap interrupt: When a memory block is full in the swap mode, an interrupt will be generated

28 JDEC

28.1 Overview

JDEC can decode the JPG compressed images in memory into the original images, and then convert them into RGB format for display through the YUV2RGB module.

28.2 Features

- Supports YUV400/YUV420/YUV422 formats.
- Adjustable quantization coefficient: 1–75, 100.
- Supports swap mode
- Supports Skip Mode, where the JPG header is automatically skipped.
- Buffer with up to 16 JPG images.
- Supports flexible interrupt configuration mode, which can set an interrupt at the specified number of images or all complete interrupt.

28.3 Functional Description

28.3.1 Output Configuration

You can select the YCbCr format of output data by setting the <REG_YUV_MODE> bit in the register JDEC_CONTROL_1, including YCbCr4:2:2 planar format, YCbCr4:2:0 planar format, and YCbCr4:0:0 grayscale image. The start address of the output image data is determined by the registers JDEC_YY_FRAME_ADDR and JDEC_UV_FRAME_ADDR. The resolution of the output image data is determined by the register JDEC_FRAME_SIZE, in which the <REG_FRAME_HBLK> bit represents the number of 8×8 blocks in the vertical direction and the <REG_FRAME_WBLK> bit represents the number of 8×8 blocks in the horizontal direction.

28.3.2 Quantization Coefficient

The quantization coefficient can be set by the <REG_Q_MODE> bit in the register JDEC_CONTROL_1. When the set value does not exceed 75, the quantization coefficient is the set value. When that exceeds 75, the quantization coefficient is 100.

28.3.3 Swap Mode

In Swap Mode, a space with a size of two frames of output images is used as a buffer. When one frame of space is full, the decompressed image data continues to be stored in another frame of space buffer. The two frames of space are used alternately back and forth. This mode can be linked with the Display Module to improve efficiency.

28.3.4 Skip Mode

When the <REG_HDER_SKIP> bit in the register JDEC_HEADER_SKIP is set to 1, the Skip Mode is enabled. Then, the JDEC Module takes the value that equals to the start address of JPG image set by the <REG_JP_ADDR> bit in the register JDEC_FRAM_PUSH plus the number of byte offset set by the <REG_HDER_SKIP_BYTE> bit in the register JDEC_HEADER_SKIP as the final actual start address for decompressing JPG images. This applies to the scenario of skipping JPG header information.

28.3.5 Buffer

The JDEC input buffer can store the information of up to 16 JPG images. Every time one "1" is written to the <REG_JP_-PUSH> bit in the register JDEC_FRAM_PUSH, the free space of the buffer decreases. Every time JDEC decompresses a JPG image, this free space increases. The number of JPG images to be decompressed in the current buffer can be obtained by reading the <JP_FRAME_CNT> bit in the register JDEC_FRAM_STS. The <FRAME_VALID_CNT> bit in the register JDEC_CONTROL_3 indicates the number of decompressed images yet not processed.

28.3.6 Operating Process

- Disable the JDEC Module and clear the REG_MJ_DEC_ENABLE bit in the register JDEC_CONTROL_1.
- Set the formats of output image data as YUV422/YUV420/YUV400/…
- Choose whether to enable Swap Mode according to the application scenario.
- Set the quantization coefficient of images.
- Set the start addresses of Y and UV components of the output image in memory respectively.
- Set the numbers of horizontal and vertical 8×8 blocks according to the resolution of image data.
- If you need to enable Skip Mode, set the number of bytes to skip.
- Enable the JDEC Module and set the REG_MJ_DEC_ENABLE bit in the register JDEC_CONTROL_1 to "1".
- Write the start address of the input JPG images and the command to cache in the buffer synchronously, that is,

writing the start address of JPG images into the register JDEC_FRAM_PUSH and setting its <REG_JP_PUSH> bit to “1”.

28.4 Precautions

- The start address of JPG image in memory shall be 8-byte aligned.
- The resolution of JPG images must be an integer multiple of 8.
- Currently, only the standard Huffman coding is supported.

29 VENC

29.1 Overview

VENC follows the H264 video coding standard, which mainly compresses objects by prediction and motion compensation, and improves image quality by a loop filter, while meeting the requirements of stream transmission and image quality.

29.2 Features

- 1920x1080p@30fps + 640x480@30fps, BP/MP
- Input: Semi-Planar YCbCr 4:2:0
- Output: NALU(Network Abstract Layer Uint) in byte stream format
- CBR/VBR mode
- Up to 8 ROI
- Up to 16 OSD coding areas
- Supports software and linked modes
- Dynamically configurable max/min quantization parameters
- Dynamically configurable I/P-frame target bit
- Dynamically configurable I-frame distance

29.3 Functional Description

29.3.1 Software Mode and Linked Mode

Software Mode: The hardware interacts with the software in the unit of frame. The software sets that the hardware compresses one frame only after one frame is started, and a frame interrupt is returned after compression is completed. The input frame memory shall contain at least one frame.

Linked Mode: The software sets the <CFG_S_ENC_SEQ_EN> or <CFG_ENC_SEQ_EN> bit in H264_ENCODER_CTRL to enable sequence encoding. The hardware automatically detects the state of the input frame and compresses it. Until software pulls down Sequence Enable, the hardware stops compression (ending at a complete frame) and the sequence interrupt is returned. When each frame in the sequence is finished, a frame interrupt is returned.

29.3.2 CBR/VBR Mode

If <NUM_IMB_BITS> or <S_NUM_IMB_BITS> in CORE_REG23 is set to 0, the stream is in VBR Mode. If it is not 0, it is in CBR Mode. In VBR Mode, I/P-frame fixed quantization parameters (CORE_REG3/CORE_REG5) shall be set. It is not recommended to switch modes when hardware is working.

29.3.3 Dual-Stream

The frame enable, frame interrupt, sequence enable, and sequence interrupt of the two streams are configured separately. Essentially, the frame rate of two streams shall be the same, but the resolution and other compression parameters can be different.

29.3.4 Region of Interest (ROI)

A single stream supports up to 8 ROI areas, and each area has its own enabling and start/end macroblock position settings. It is necessary to set the <CFG_ROI_UPD> bit in H264_ROI_MODE or the <CFG_S_ROI_UPD> bit in H264_S_ROI_MODE to “1”, to make the new settings take effect. The hardware synchronizes ROI settings for each frame.

29.3.5 OSD Coding Area

The hardware supports up to 16 OSD areas, and H264_OSD_EN decides the OSD area of each stream as configured in the software. The OSD area is represented by the start/end macroblock position.

29.3.6 Dynamically Configurable Max/Min Quantization Parameters

To provide a better control stream, after the software sets CORE_REG31, it is necessary to set H264_ENCODER_CTRL (the <CFG_QR_UPD> and <CFG_S_QR_UPD> bits set two streams respectively) to make the new setting value take effect. The hardware will synchronize the new setting value in the unit of GOP to adjust the quantization parameters dynamically.

29.3.7 Dynamically Configurable I/P-Frame Target Bit

The software can set CORE_REG23/CORE_REG24 to adjust the target bit of I/P-frame. After CORE_REG23/CORE_REG24 is set, it is necessary to set H264_ENCODER_CTRL (the <CFG_QR_UPD> and <CFG_S_QR_UPD> bits set two streams respectively) to make the new setting value take effect. The hardware will synchronize the new setting value in the unit of GOP (CBR Mode).

29.3.8 Dynamically Configurable I-Frame Distance

After the software sets CORE_REG8, it is necessary to set H264_ENCODER_CTRL (the <CFG_QR_UPD> and <CFG_S_QR_UPD> bits set two streams respectively) to make the new setting value take effect. The hardware will synchronize the new setting value in the unit of GOP (CBR/VBR available).

29.3.9 Video Sequence Interrupt

In Linked Mode, when software pulls down Sequence Enable, the hardware generates sequence interrupt after processing the current frame (<S_SEQ_DONE_INT> or <SEQ_DONE_INT> bit in VDO_INT), and this interrupt requires the software to set the interrupt clear (<S_SEQ_DONE_INT_CLR> or <SEQ_DONE_INT_CLR> bit in VDO_INT_CLR). This interrupt can be ignored by masking (<S_SEQ_DONE_INT_MASK> or <SEQ_DONE_INT_MASK> bit in VDO_INT_MASK).

29.3.10 Frame Interrupt

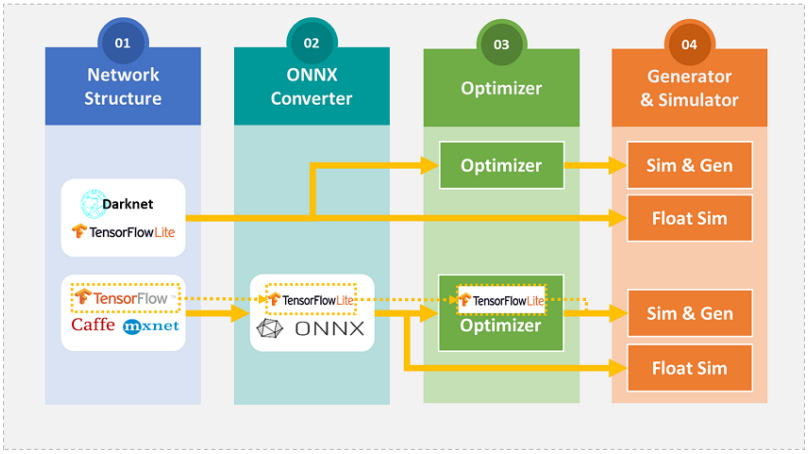
In Software Mode and Linked Mode, the frame interrupt (<S_FRM_DONE_INT> or <FRM_DONE_INT> in VDO_INT) will be generated as soon as the hardware compresses one frame. This interrupt requires the software to set the interrupt clear (<S_FRM_DONE_INT_CLR> or <FRM_DONE_INT_CLR> bit in VDO_INT_CLR). This interrupt can be ignored by masking (<S_FRM_DONE_INT_MASK> or <FRM_DONE_INT_MASK> bit in VDO_INT_MASK).

29.3.11 Input Cache Overflow Alert

If the video compression speed is much lower than the camera pixel, buffer overflow will occur at output. The software can detect that through the status register VDO_SRC_R_DBG (<SRC_WR_OV_RD> bit) or VDO_S_SRC_R_DBG (<S_SRC_WR_OV_RD> bit).

30 NPU toolchain

30.1 GUI Flow chart

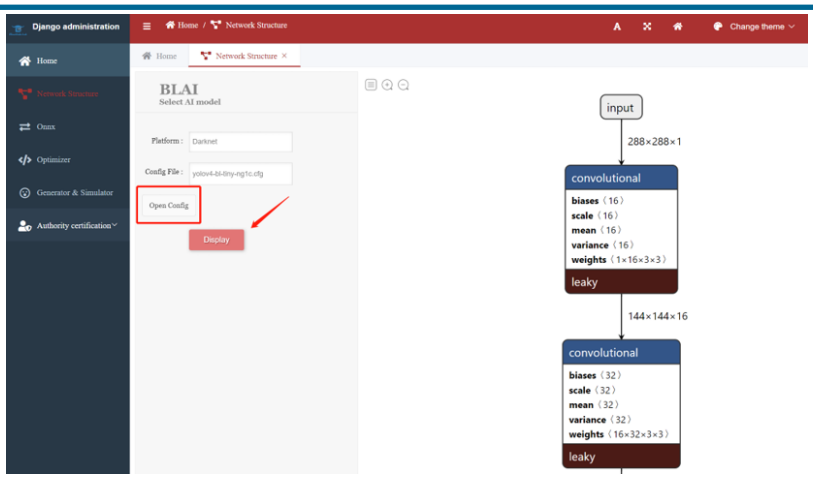


30.2 Network Structure

To display model structure via Netron on GUI, you have to choose a platform and a config file that you want to display.

Platform:

Darknet, Tensorflow, Caffe, Mxnet, ONNX



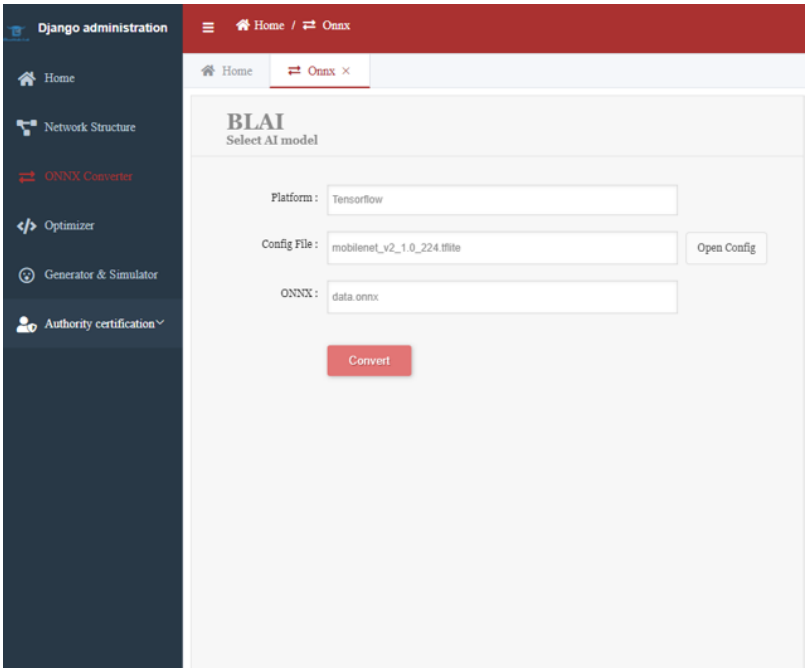
30.3 ONNX Converter

To convert your platform into ONNX, you have to choose a platform and name the ONNX file.

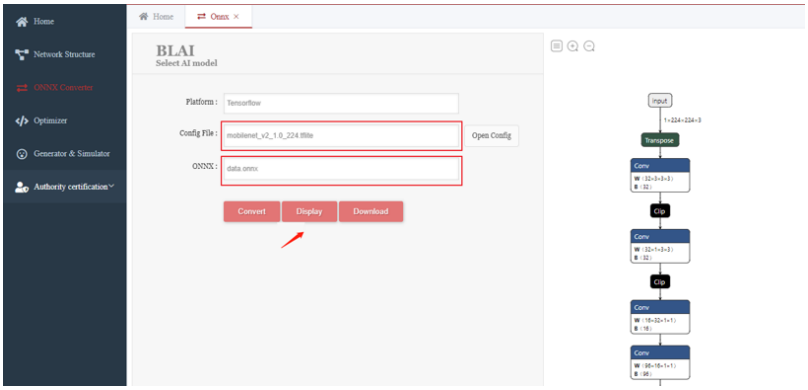
Platform:

Tensorflow, Caffe, Mxnet

(DarkNet, Tensorflow Lite is supported directly.)



After you press “convert” on ONNX Converter, the screen will show two more buttons which you can choose to “display” it on GUI or “download” it.



30.4 Optimizer

The optimizer section allows you to quantize float to fixed points.

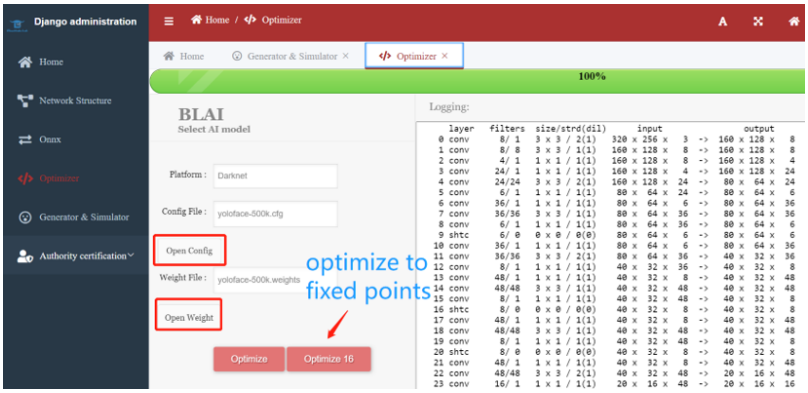
“**Optimize**”：spend more time, high accuracy

“**Fast Optimize**”：spend less time, more accuracy loss

Platform:

Darknet, Tensorflow, ONNX

(BLAI NPU also support optimizer format on Tensorflow Lite)



30.5 Generator & Simulator

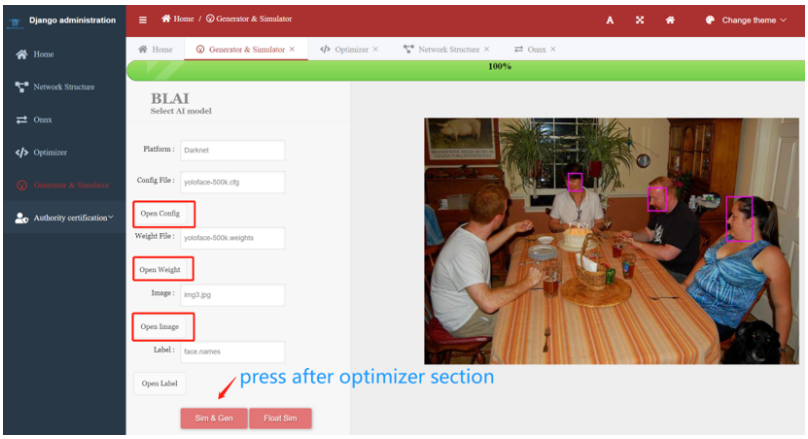
“**Sim&Gen**” will predict the image with 8 bits quantized weights after you have done the Optimizer section, and generate BLAI NPU instruction

“**Float Sim**” will predict the image with 32 bits unquantized weights.

Platform:

Darknet, Tensorflow, ONNX

(BLAI NPU also support optimizer format on Tensorflow Lite)



31 NPU

31.1 Overview

The Neural Processing Unit (NPU), also called AI accelerator, is an electronic circuit dedicated to deep learning algorithms. It can implement all the control and arithmetic logic necessary to execute machine learning algorithms, usually with a separate data memory and special instruction set architecture. NPU aims to provide higher efficiency and performance for deep learning algorithms than the general central processing unit (CPU). NPU uses a large number of computing modules to utilize high data-level parallelism, a relatively large buffer/memory to apply the data reuse patterns, and a finite data width operator for deep learning fault tolerance.

31.2 Features

- AI computing power: strong:*0.1 TOPS*
- Supports 8-bit operation
- Compatible with TensorFlowlite/ONNX/Caffe/Mxnet/Darknet/Pytorch model, and multiple frameworks
- Provide PC-side easy-to-use development tools, including model conversion, model quantification, performance prediction, and accuracy verification
- Supports single-layer instruction mode and multi-layer instruction mode
- Supports the maximum feature graph 4096 × 4096 × 4096 (width × height × depth)

31.3 Feature List

Type	Operators	Applicable Subset Spec.	Processor
Convolution	Conv	1x1 up to 7x7	**NPU**
	Depthwise Conv	1x1 up to 7x7	**NPU**
	Pad	same	**NPU**
	Deconvolution	use Upsample + Conv	**NPU**

Type	Operators	Applicable Subset Spec.	Processor
Pooling	MaxPool(2x2)	Stride 2	**NPU**
	MaxPool(3x3)	3x3	**NPU**
	AveragePool	Stride 1, 2	DSP
	GlobalAveragePool		DSP
	GlobalMaxPool		DSP
Activation	Relu		**NPU**
	LeakyRelu		**NPU**
	Relu-n	n > 0 (Relu6)	**NPU**
	Mish	Look up table	**NPU**
	ELU	Look up table	**NPU**
	PRelu		DSP
	Sigmoid		DSP
Other processing	BatchNormalization		DSP
	Add (shortcut)		**NPU**
	Concat (route)		**NPU**
	Fully Connected		**NPU**
	Mul		**NPU**
	Slice		DSP

31.4 API Reference

void gen_npu_inst_layer(npu_layer* l, bool use_tflite, bool unsgn_input, bool img_in)

```
/**
* function     Generate NPU instructions
* @param[in]   l: NPU layer data structure indicators
* @param[in]   use_tflite: whether to use tflite data format
* @param[in]   unsgn_input: whether the input data is uint8
* @param[in]   img_in: whether the input data is YUV400
* @return      Null
**/
```

bool check_BLAI_NPU_RUN(int type, int size, int stride, int dilation)

```
/**
* function     Confirm whether the layer is eligible for NPU acceleration
* @param[in]   type: The layer type (enum LAYER_TYPE)
* @param[in]   size: convolution kernel size
```

(continues on next page)

(continued from previous page)

```
* @param[in]   stride: step size
* @param[in]   dilation: Atrous convolution kernel size
* @return      true: NPU acceleration conditions are met, false: NPU acceleration conditions are not met
**/
```

bool BLAI_MEM_alloc(npu_layer l, PSRAM_ctrl *ctrl)

```
/**
* function     Automatically generate NPU related memory configuration
* @param[in]   l: NPU layer data structure indicators
* @param[in]   ctrl: memory configuration parameter structure
* @return      true: memory configuration succeeded, false: memory configuration failed
**/
```

void fetch_BLAI_data_general(npu_layer *l, PSRAM_ctrl* ctrl, bool use_tflite, bool img_in)

```
/**
* function     Automatically generate NPU instructions (the maximum number of input layers is 2)
* @param[in]   l: NPU layer data structure indicators
* @param[in]   ctrl: memory configuration parameter structure
* @param[in]   use_tflite: whether to use tflite data format
* @param[in]   img_in: whether the input data is YUV400
* @return      null
**/
```

void fetch_BLAI_data_route(npu_layer *l, PSRAM_ctrl* ctrl, bool use_tflite, bool img_in)

```
/**
* function     For ROUTE layers with more than two input layers, NPU commands are automatically generated
* @param[in]   l: NPU layer data structure indicators
* @param[in]   ctrl: memory configuration parameter structure
* @param[in]   use_tflite: whether to use tflite data format
* @param[in]   img_in: whether the input data is YUV400
* @return      null
**/
```

bool BLAI_encode(npu_layer *l, PSRAM_ctrl* ctrl, int use_tflite, bool img_in)

```
/**
* function     Automatically generate NPU instructions, NPU-related memory configuration
* @param[in]   l: NPU layer data structure indicators
* @param[in]   ctrl: memory configuration parameter structure
* @param[in]   use_tflite: whether to use tflite data format
```

(continues on next page)

(continued from previous page)

```
* @param[in]   img_in: whether the input data is YUV400
* @return      true: NPU acceleration conditions are met and memory configuration is successful, false: NPU␣
 ↪acceleration conditions are not met or memory configuration fails
**/
```

void Load_NPU_weights(npu_layer l, int8_t* WEI_buf, int* BIAS_buf, bool use_tflite)

```
/**
* function     The convolution parameters of this layer are stored in the NPU-specific parameter memory in␣
 ↪the way specified by the NPU.
* @param[in]   l: NPU layer data structure indicators
* @param[in]   WEI_buf: NPU dedicated parameter memory
* @param[in]   BIAS_buf: NPU dedicated parameter memory
* @param[in]   use_tflite: whether to use tflite data format
* @return      null
**/
```

void Store_tensor_data_to_NPU(npu_layer l, fixed_point_t* DATA_buf)

```
/**
* function     Store the tensor data in the operation graph into the dedicated data memory of the NPU
* @param[in]   l: NPU layer data structure indicators
* @param[in]   DATA_buf: NPU dedicated data memory
* @return      null
**/
```

void Load_NPU_data_to_tensor(npu_layer l, fixed_point_t* DATA_buf)

```
/**
* function     Read tensor data from NPU dedicated data memory
* @param[in]   l: NPU layer data structure indicators
* @param[in]   DATA_buf: NPU dedicated data memory
* @return      null
**/
```

void forward_NPU(npu_layer l, int8_t* DATA_buf, bool use_tflite)

```
/**
* function     Use the command to run the NPU (you need to use gen_npu_inst_layer to generate the command␣
 ↪first)
* @param[in]   l: NPU layer data structure indicators
* @param[in]   DATA_buf: NPU dedicated data memory
* @param[in]   use_tflite: whether to use tflite data format
```

(continues on next page)

(continued from previous page)

```
* @return      null
**/
```

31.5 Data Structure Reference

npu_layer data structure:

```
/** NPU layer information of dynamic fixed point format(CMSIS/NMSIS) */

struct npu_layer {

    ////////////////////////////////////////////
    /////    Begin of user define region      /////
    ////////////////////////////////////////////

    /** operation type (enum LAYER_TYPE)*/
    uint8_t type;

    /** activation type (enum ACTIVATION)*/
    uint8_t activation;

    /** layer width */
    uint16_t w;

    /** layer height */
    uint16_t h;

    /** layer channel (1) */
    uint16_t c;

    /** extra layer channel (2-8) */
    uint16_t cn[7];

    /** layer output width */
    uint16_t out_w;

    /** layer output height */
    uint16_t out_h;

    /** layer output channel */
    uint16_t out_c;
```

(continues on next page)

(continued from previous page)

```
    /** number of input layers*/
    uint8_t input_num;

    /** CONV kernel size */
    uint8_t size;

    /** CONV groups size */
    uint16_t groups;

    /** CONV dilation size */
    uint8_t dilation;

    /** stride size */
    uint8_t stride;

    /** input tensor type*/
    uint8_t input_type;

    /** True: combined layer need to keep output data */
    bool mid_out;

    /** True: input image is 1-channel */
    bool img_in;

    /** dynamic fixed point format(CMSIS/NMSIS) for input data */
    int8_t fdata;

    /** dynamic fixed point format(CMSIS/NMSIS) for weight */
    int8_t fweight;

    /** dynamic fixed point format(CMSIS/NMSIS) for bias */
    int8_t fbias;

    /** dynamic fixed point format(CMSIS/NMSIS) for output data */
    int8_t fout;

    /** dynamic fixed point format(CMSIS/NMSIS) for route input data (1) */
    int8_t froute1;

    /** dynamic fixed point format(CMSIS/NMSIS) for route input data (2) */
    int8_t froute2;

    /** dynamic fixed point format(CMSIS/NMSIS) for route input data (3-8) */
```

(continues on next page)

(continued from previous page)

```
    int8_t frouten[6];

    /** Tensorflow-Lite input offset (1) */
    uint8_t tf_input1_offset;

    /** Tensorflow-Lite input offset (2) */
    uint8_t tf_input2_offset;

    /** Tensorflow-Lite input offset (3-8) */
    uint8_t tf_input_offset_extra[6];

    /** Tensorflow-Lite output offset */
    uint8_t tf_output_offset;

    /** Tensorflow-Lite input shift (1) */
    int8_t tf_input1_shift;

    /** Tensorflow-Lite input shift (2) */
    int8_t tf_input2_shift;

    /** Tensorflow-Lite input shift (3-8) */
    int8_t tf_input_shift_extra[6];

    /** Tensorflow-Lite output shift */
    int8_t tf_output_shift;

    /** Tensorflow-Lite quantized_activation_min */
    int16_t quantized_activation_min;

    /** Tensorflow-Lite quantized_activation_max */
    int16_t quantized_activation_max;

    /** Tensorflow-Lite input multiplier (1) */
    int tf_input1_multiplier;

    /** Tensorflow-Lite input multiplier (2) */
    int tf_input2_multiplier;

    /** Tensorflow-Lite input multiplier (3-8) */
    int tf_input_multiplier_extra[6];

    /** Tensorflow-Lite output multiplier */
    int tf_output_multiplier;
```

(continues on next page)

(continued from previous page)

```
/** Tensorflow-Lite route input multiplier */
int tf_route_input_multiplier;

/** Tensorflow-Lite route input multiplier */
int tf_route_input_shift;

/** Tensorflow-Lite left shift */
int8_t tf_left_shift;

/** pointer of input data buffers */
int8_t* input_i8[8];

/** pointer of output data buffer */
int8_t* output_i8;

/** pointer of combined layer output data buffer */
int8_t* mid_output_i8;

/** pointer of weight buffer */
int* weights;

/** pointer of bias buffer */
int* biases;

////////////////////////////////////////////
/////     End of user define region       /////
////////////////////////////////////////////

/** pointer of NPU instruction buffer */
uint8_t* NPU_inst;

/** flag of NPU processor */
bool NPU_on;

/** memory patch size*/
uint32_t DRAM_patch_size;

/** memory patch location for input data */
uint16_t DRAM_in[8];

/** memory patch location for output data */
uint16_t DRAM_out[8];
```

(continues on next page)

(continued from previous page)

```
/** memory patch location for mid output data */
uint16_t DRAM_mid_out;

/** memory patch location for weight data */
uint16_t DRAM_weight;

/** size of weight data */
int DRAM_nweight;

/** memory patch location for bias data */
uint16_t DRAM_bias;

/** size of bias data */
int DRAM_nbias;

/** uint8_t input data */
bool unsgn_input;

/** number of instruction */
uint8_t inst_cnt;
};
```

32 IPC

32.1 Overview

The Interprocessor Communication (IPC) is a communication mechanism between multi-core processors. BL808 has three heterogeneous cores, and any two cores can communicate with each other through IPC.

32.2 Features

- Each core has an independent 32-bit IPC channel
- Any two cores can communicate with each other

32.3 Functional Description

32.3.1 Basic Principle

Each core has six IPC registers: IPCx_TRI, IPCx_STS, IPCx_ACK, IPCx_IEN, IPCx_IDIS and IPCx_ISTS. The length of each register is 32-bit, and each bit corresponds to one channel of IPC. The cores M0, LP and D0 correspond to IPC0, IPC1 and IPC2 respectively. When one core needs to issue a notification to another, it only needs to write “1” to the corresponding channel of IPCx_TRI of the receiving core. At this time, the corresponding channel of IPCx_STS of the receiving core is set to “1”. If the interrupt of the corresponding IPC channel of the receiving core is also enabled, an interrupt will be received, indicating that the receiving core receives the notification from the other core.

32.3.2 Operating Process

For example, the core x sends a notification to the core Y, and the channel z is used. The operation flow is as follows:

- The core y enables the interrupt of channel z. That is, the core y performs operation IPCy_IEN = 1«z
- The core x writes 1 to the bit z in the register IPCy_TRI. That is, core x performs operation IPCy_TRI = 1«z
- Upon receiving the interrupt, the core y responds to the core x. That is, the core y performs operation IPCy_ACK = 1«z

- The core y performs the notified application layer related operation

The operation flow is as follows:

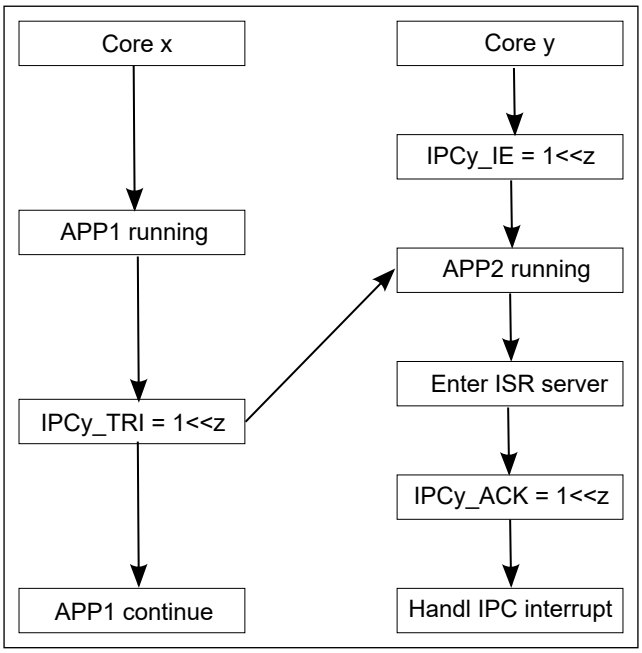


Fig. 32.1: IPC Flow

33 LowPower

33.1 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.

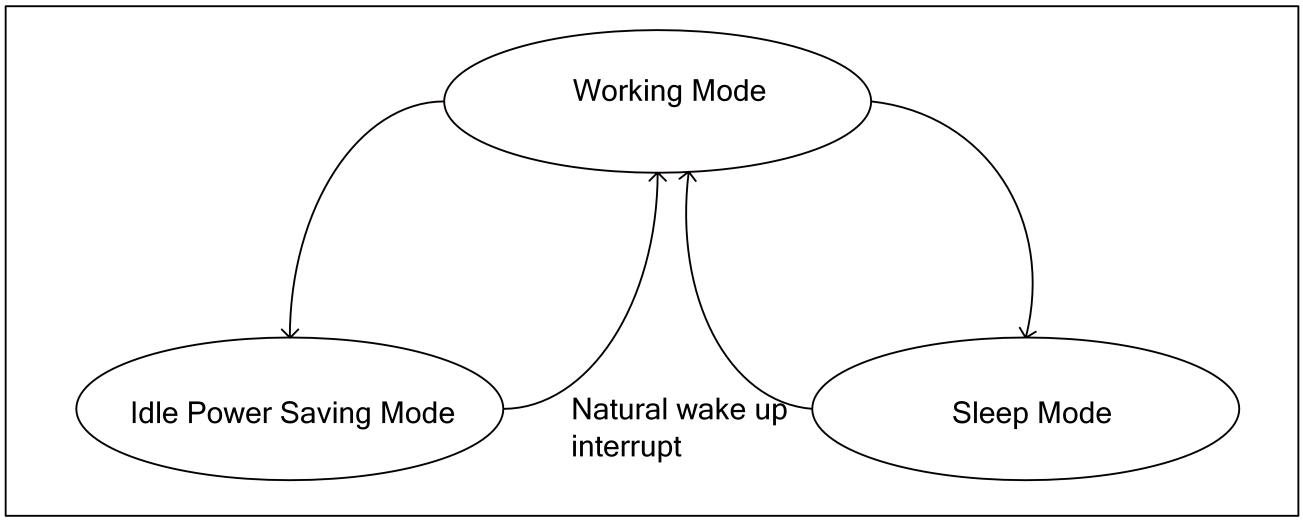


Fig. 33.1: Low power modes

33.2 Features

- Clock control: clock control of peripherals in GLB, small-scale power saving, and fast response speed
- Sleep control (PDS): 4 levels (PDS 1/2/3/7), 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

33.3 Functional Description

33.3.1 Power Domain

There are 7 power domains in the BL808 chip, with main functions as follows:

- PD_AON_RTC
  - RC32K/XTAL32K control register is reserved
  - RTC Counter clock source selection function
  - RTC can be used for wake-up or LED flashing
- PD_AON
  - HBN state machine controls power supply/isolation/reset/clock
  - Maintain internal voltage output selection
  - AON_PIN (GPIO9/10/11/12/13/40/41) pin wake-up control
  - HBN_OUT0_PIR (Acomp0/Acomp1/bor/pir) wake-up mask, enable register, and interrupt status register
  - BOR (Brown Out Reset) function
- PD_AON_HBNCORE
  - Partial power control register
  - 4 KB HBN_RAM, which is used to save program data before the PDS/HBN mode is enabled, so that data will not disappear in these modes
  - PIR digital control: PIR is a Passive Infra-Red (PIR) sensor, a peripheral in HBN area, which can be used as HBN wake-up source
  - AON_PIN control and IO retention function in HBN/PDS mode
  - Acomp0 and Acomp1 configuration, GPADC clock source selection and configuration
- PD_CORE
  - HBN state machine controls power supply/isolation/reset/clock
  - Reserved 64 KB RAM
  - WIFI/BLE timer control
  - 160KB WRAM/EM
- PD_CORE_MISC_DIG
  - M0 CPU and its peripherals

  - Chip’s global register

- PD_USB
  - USB digital controller
- PD_MM
  - D0 CPU and its peripherals, PSRAM
  - Codec
  - VRAM 32~96KB
  - BLAI

Each power domain is controlled by 9 different power modes as shown below:

Table 33.1: Power mode

NO.	Scenario	Power Domain						
		PD_AON_RTC	PD_AON	PD_AON_HBN_CORE	PD_CORE	PD_CORE_MISC_DIG	PD_USB	PD_MM
1	Normal	ON	ON	ON	ON	ON	ON	ON
2	PDS1	ON	ON	ON	ON	ON	ON	OFF
3	PDS2	ON	ON	ON	ON	ON	OFF	ON
4	PDS3	ON	ON	ON	ON	ON	OFF	OFF
5	PDS7	ON	ON	ON	ON	OFF	OFF	OFF
6	HBN0	ON	ON	ON	OFF	OFF	OFF	OFF
7	HBN1	ON	ON	OFF	OFF	OFF	OFF	OFF
8	HBN2	ON	OFF	OFF	OFF	OFF	OFF	OFF
9	HBN3	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Note:

- To enter the HBN2 mode, the chip needs to enter the HBN1 mode first, and then turn VDDIO2 off
- In the HBN2 mode, only RTC_Timer is kept, but no wake-up is provided. To quit the HBN2 mode, you only need to resupply power to VDDIO2.
- VDD33_RTC/VDDIO2(AON) share pin encapsulation (QFN88 Type2, QFN68), without HBN2 mode
- The pu_chip must be powered off before entering the HBN3 mode. In this mode, most devices will be powered off, but you can set whether to power RTC off through rtc_pu_chip_sel.

If rtc_pu_chip_sel is 1, the RTC power supply is always maintained. If that is 0, it is controlled by pu_chip. The value of rtc_pu_chip_sel is determined at encapsulation.

33.3.2 Wake-up Source

The chip supports multiple wake-up sources, which can wake up the chip from different power modes.

PDS 1/2/3 can be wake up in the following ways:

- HBN wake-up source
- All GPIO wake-up
- Infrared receiver
- BLE wake-up event
- WIFI wake-up event
- PDS timer

The wake-up sources of other power modes are shown as follows:

Table 33.2: Wakeup source

Power Mode	Wake-up Source
PDS7	PDS timer/PSD_PIN/RTC/AON_PIN/BOR/Pir/Acomp0/Acomp1
HBN0	RTC/AON_PIN/BOR/Pir/Acomp0/Acomp1
HBN1	RTC/AON_PIN
HBN2	Resupply power to VDDIO2
HBN3	Resupply power to pu_chip

33.3.3 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 the HBN0 mode. 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.

33.3.4 IO Retention

IO retention includes AON_IO retention and PDS_IO retention. In the PDS 0/1/2/3 mode, for the chip’s MISC domain is still powered, GPIO can be controlled by the glb register. After the glb register is powered off, AON_CTRL and PDS can control the IE/PD/PU of AON_IO and PDS_IO.

AON_IO

AON_IO refers to GPIO9/10/11/12/13/14/15/40/41. GPIO40/41 is used as the input and output of XTAL32K by default (it can only be used as AON_IO if it is changed to other pinmux purposes). When the IE/OE of GPIO40/41 is all set to 0, GPIO40/41 multiplexes the analog function XTAL32K_INXTAL32K_OUT. Contrarily, when that is not all set to 0, GPIO40/41 is used as a normal IO function. For example, when GPIO40 is used as a general IO function, reg_en_aon_ctrl_gpio[7] and reg_aon_pad_oe[7] are set to 0, and reg_aon_pad_ie_smt[7] is set to 1.

1. Hardware’s IO retention: HBN can control the IE/PD/PU/OE/O of AON_IO, thus achieving IO retention. When reg_en_aon_ctrl_gpio is 1, reg_aon_gpio_pu controls the pull-up enable of AON_IO and reg_aon_gpio_pd controls the pull-down enable. The reg_aon_gpio_oe controls OE, and aon_led_out[0], [1], and [2] control AON PAD O, respectively. Whereas, IE/SMT can be controlled by reg_aon_gpio_ie_smt no matter the value of reg_en_-

aon_ctrl_gpio is 0 or 1. For example, when reg_en_aon_ctrl_gpio is 0, even if reg_aon_pad_pu is 1, it cannot enable pull-up. But setting reg_aon_gpio_ie_smt to 1 can enable the IE function.

2. Software IO retention: After reg_aon_gpio_iso_mode is set to 1, AON PAD can retain OEO, but PUPD cannot be retained when entering the HBN mode. After HBN wake-up, the AON PAD state will still be retained, and you need to clear reg_aon_gpio_iso_mode before exiting IO retention state. For example, GPIO40 keeps high level in the HBN mode, so you need to configure it as a normal IO function first, then configure it to output high level through the glb register, and finally set reg_aon_gpio_iso_mode to 1 before entering the HBN mode.

PDS_IO

PDS_IO refers to other 32 GPIOs except AON_IO, and they are divided into 3 groups by the physical location of PAD:

- Left: GPIO0-8
- Right: GPIO16-23
- Top: GPIO24-39

1. Hardware IO retention: The IE/PD/PU of PDS_IO can be controlled by the register pds_gpio_i_set, and the same group of GPIOs must hold the same level. For example, if GPIO0 is configured to pull-up, GPIO8 is also configured to pull-up.
2. Software IO retention: When cr_pds_gpio_iso_mode is 1, if cr_pds_gpio_kee_en[0], [1], and [2] is 1 in the PDS7 mode, GPIO0-8, GPIO16-23, and GPIO24-38 enter the GPIO retention state respectively. After PDS7 is woke up, the PDS_IO state will still be retained, so you need to clear cr_pds_gpio_iso_mode to 0 before existing IO retention. The advantage of this IO retention is that the same group of GPIOs can hold different levels.

33.4 Register description

Name	Description
HBN_TIME_L	
HBN_TIME_H	

33.4.1 HBN_TIME_L

Address：0x2000f004

hbn_time_l

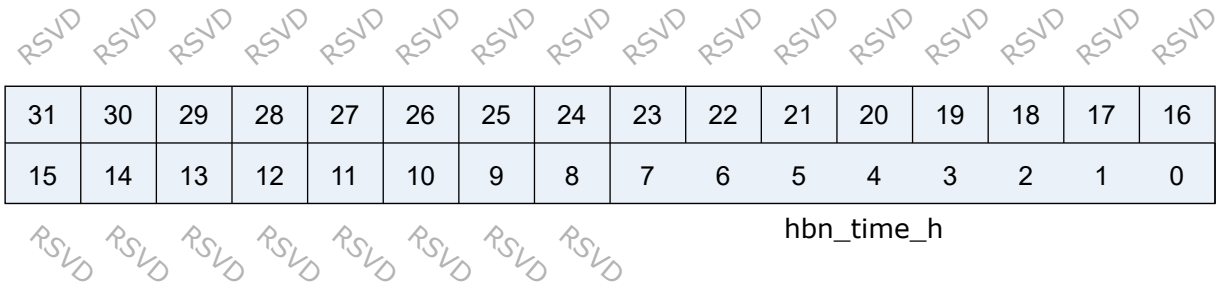
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

hbn_time_l

Bits	Name	Type	Reset	Description
31:0	hbn_time_l	r/w	32'h0	RTC timer compare bit 31:0

33.4.2 HBN_TIME_H

Address：0x2000f008



Bits	Name	Type	Reset	Description
31:8	RSVD			
7:0	hbn_time_h	r/w	8'h0	RTC timer compare bit 39:32

34 SEC ENG

34.1 Overview

34.1.1 AES

Advanced Encryption Standard (AES) is the most common type of symmetric encryption algorithm. Also known as Rijndael encryption algorithm in cryptography, it is a block encryption standard adopted by the U.S. federal government.

34.1.2 MD5

Message Digest Algorithm 5 (MD5), a widely used cryptographic hash function, can generate a 128-bit (16-byte) hash value, to ensure the integrity and consistency of information transmission.

34.1.3 SHA

SHA256 is a variant of Secure Hash Algorithm 2 (SHA2). SHA2, the successor of SHA1, is a cryptographic hash function algorithm standard designed by the United States National Security Agency. SHA2 has six different variants: SHA224, SHA256, SHA384, SHA512, SHA512/224, and SHA512/256.SHA-512/224 means that the result takes the first 224 bits of SHA-512, and SHA-512/256 means that the result takes the first 256 bits of SHA-512.

34.1.4 CRC

Cyclic Redundancy Check (CRC) is a channel coding technique that generates short fixed-digit check codes based on data including network packets or computer files. It is mainly used to detect or check possible errors after data transfer or storage. It detects errors based on the principle of division and remainder.

34.1.5 GMAC

GMAC is to use Galois Field (GF, finite field) multiplication to calculate the MAC value of a message.

34.2 Features

- Encryption and decryption of aes128, aes192, and aes256
- Crc16, crc32, md5, sha256, and sha512
- Trng
- Gmac

34.3 Principle

AES means that the same key is used for encryption and decryption, with encryption process as follows:

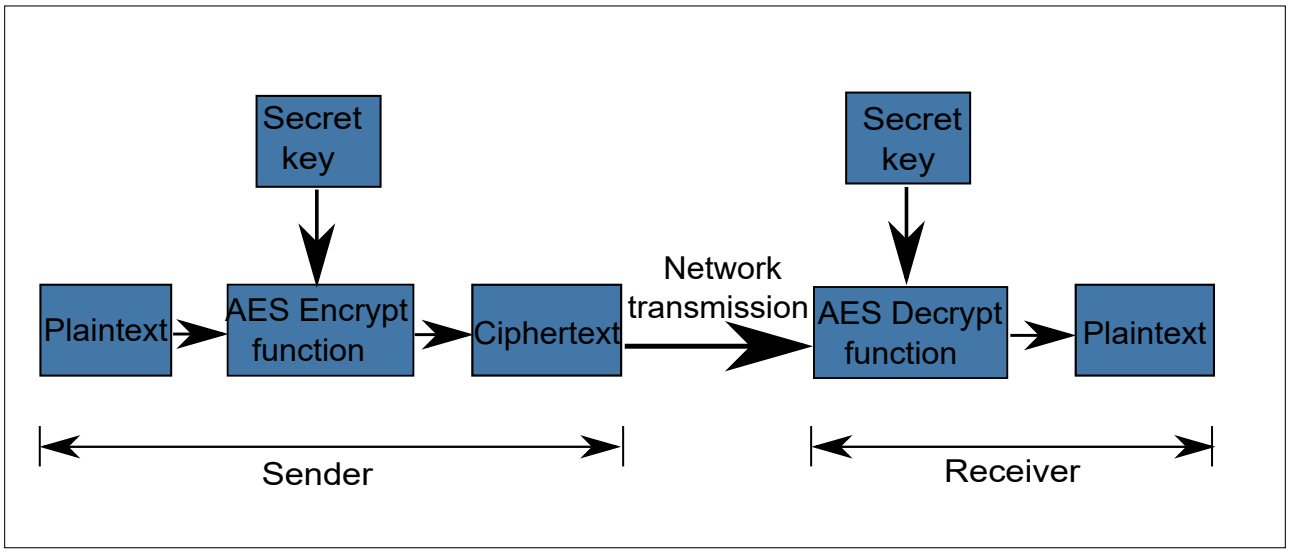


Fig. 34.1: AES Encryption Process

The following briefs the functions and significance of each part:

- Plaintext (P): unencrypted data.
- Key (K): The key used to encrypt plaintext. In symmetric encryption algorithm, encryption and decryption share one key. The key is generated by negotiation between the receiver and the sender, but it cannot be directly transmitted over network. Otherwise it will leak the key. Usually, the key is encrypted by an asymmetric encryption algorithm, and then transmitted to the other side over network, or the key is discussed face to face. The key must not be leaked. Otherwise the attacker will restore the ciphertext and steal confidential data.
- AES encryption function: Let this function be E, then C = E(K, P), where P, K and C denote plaintext, key, and ciphertext, respectively. That is, if plaintext (P) and key (K) are input as the parameters of function, this function

will output ciphertext (C).

- Ciphertext (C): data processed by encryption function.
- AES decryption function: Let this function be D, then P = D(K, C), where C, K, and P denote ciphertext, key, and plaintext, respectively. That is, if ciphertext (C) and key (K) are input as the parameters of function, this function will output plaintext (P).

34.3.1 Implementation of MD5

MD5 can be regarded as a block-based algorithm, as it requires that the length of data processed each time is 512 bits. However, the length of plaintext to be actually processed is not always an integer multiple of 512, so data padding is required. Assuming that the length of the original plaintext message is K, the padding of MD5 is further divided into 2 steps:

1. Append padding bits: 100... are padded after the K bits of the original plaintext message until 512-64 = 448 bits. The padding rule is that only the first bit is 1, followed by all zeros.
2. Append length: The length of the original message is filled after the result of the first step, and the available storage length is 64 bits. The whole padding process is as follows:

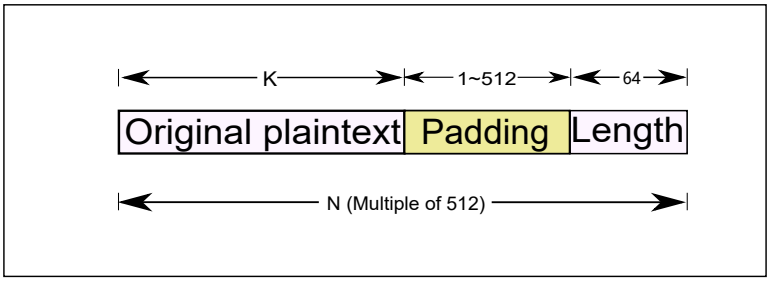


Fig. 34.2: MD5 Padding

Data to be used:

- Four constants: A = 0x67452301, B = 0x0EFCDAB89, C = 0x98BADCFE, D = 0x10325476;
- Four functions: F(X,Y,Z)=(X & Y) | ((~X) & Z); G(X,Y,Z)=(X & Z) | (Y & (~Z)); H(X,Y,Z)=X ^ Y ^ Z; I(X,Y,Z)=Y ^ (X | (~Z));

The message is divided into 512-bit groups for processing. Each group undergoes 4 rounds of transformation. The above 4 constants are used as the initial variables for calculation, to re-output 4 variables, based on which the next group is calculated. For the last group, the 4 variables are the final result, namely MD5 value.

34.3.2 Implementation of SHA256

It is implemented in a way similar to that of MD5. Its rule of data padding is the same as that of MD5. SHA256 uses 8 initial hash values and 64 hash constants.

The 8 initial hash values are as follows:

- h0 = 0x6a09e667
- h1 = 0xbb67ae85
- h2 = 0x3c6ef372
- h3 = 0xa54ff53a
- h4 = 0x510e527f
- h5 = 0x9b05688c
- h6 = 0x1f83d9ab
- h7 = 0x5be0cd19

These initial values are obtained by taking the first 32 bits from the decimal part of the square root of the first 8 prime numbers (2, 3, 5, 7, 11, 13, 17, and 19) in natural numbers.

The 64 constants are as follows:

- 428a2f98 71374491 b5c0fbcf e9b5dba5
- 3956c25b 59f111f1 923f82a4 ab1c5ed5
- d807aa98 12835b01 243185be 550c7dc3
- 72be5d74 80deb1fe 9bdc06a7 c19bf174
- e49b69c1 efbe4786 0fc19dc6 240ca1cc
- 2de92c6f 4a7484aa 5cb0a9dc 76f988da
- 983e5152 a831c66d b00327c8 bf597fc7
- c6e00bf3 d5a79147 06ca6351 14292967
- 27b70a85 2e1b2138 4d2c6dfc 53380d13
- 650a7354 766a0abb 81c2c92e 92722c85
- a2bfe8a1 a81a664b c24b8b70 c76c51a3
- d192e819 d6990624 f40e3585 106aa070
- 19a4c116 1e376c08 2748774c 34b0bcb5

- 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
- 748f82ee 78a5636f 84c87814 8cc70208
- 90befffa a4506ceb bef9a3f7 c67178f2

Similarly, these constants are obtained by taking the first 32 bits from the decimal part of the cube root of the first 64 prime numbers (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97…) in natural numbers.

SHA256 hash function:

- Ch(x,y,z)=(x□y)□(¬x□z)
- Ma(x,y,z)=(x□y)□(x□z)□(y□z)
- Σ0(x)=S ^{2} (x)□S ^{13} (x)□S ^{22} (x)
- Σ1(x)=S ^{6} (x)□S ^{11} (x)□S ^{25} (x)
- σ0(x)=S ^{7} (x)□S ^{18} (x)□R ^{3} (x)
- σ1(x)=S ^{17} (x)□S ^{19} (x)□R ^{10} (x)

Where:

- □: bitwise “AND”
- ¬: bitwise “Padding”
- □: bitwise “Exclusive OR”
- S ^{n} : Circularly shift right by n bits
- R ^{n} : Shift right by n bits

34.3.3 Principle of GMAC

Authentication is actually a redundant message generated against the message itself, that is, the message authentication code (MAC). MAC is a technique for authenticating the integrity of a message. In cryptography, MAC refers to a verification mechanism used by both communication entities and a tool to ensure the integrity of message data. MAC is a hash function with a key. But why does it need a key? The reason is that the message can be tampered with during transmission, so can the hash value. Therefore, to ensure a valid hash value, the hash value is protected by encryption, so that the receiver can judge the integrity of the whole message through the hash value upon reception, thus completing information transfer.

The MAC process is shown as follows:

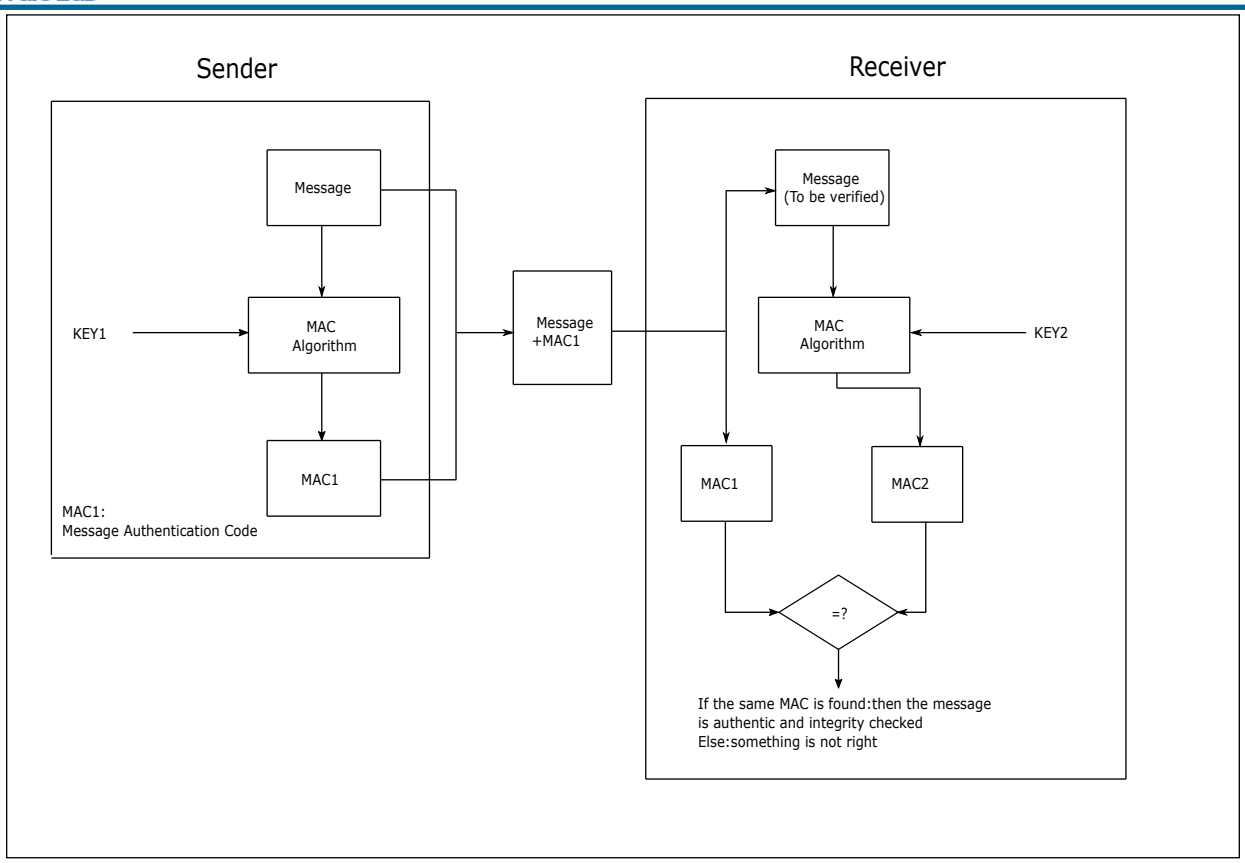


Fig. 34.3: MAC Flow Chart

1. The sender and the receiver share the key K in advance (keep KEY1 and KEY2 values in the above figure consistent).
2. The sender calculates the MAC value based on the message (KEY1 is used to calculate the MAC1 for the original message).
3. The sender sends the original message and MAC1 to the receiver.
4. The receiver calculates MAC2 using KEY2 based on the received original message .
5. The receiver compares the calculated MAC2 with the MAC1 received from the sender.
6. If the MAC is consistent, the receiver can judge that the message is indeed from the sender (authentication succeeded) and has not been tampered with or there is a transmission error. If not, the message is not from the sender (authentication failed).

Note: It is recommended that the sender and the receiver store the KEY in the hardware security module, and the MAC value also shall be calculated in that module, to ensure the security of KEY, for example, in the encrypted chip.

GMAC is to use Galois Field (GF, finite field) multiplication to calculate the MAC value of a message.

34.4 Functional Description

34.4.1 AES Accelerator

1. AES accelerator supports AES128/192/256 encryption and decryption.

- Configure se_aes_0_mode and se_aes_0_dec_en in the register se_aes_0_ctrlL, as shown below:

amode	adec en	运算
0	0	AES-128 加密
1	0	AES-256 加密
2	0	AES-192 加密
0	1	AES-128 解密
1	1	AES-256 解密
2	1	AES-192 解密

Fig. 34.4: AES Operation Modes

Configure se_aes_0_block_mode in the register se_aes_0_ctrlL to select among different encryption modes including ECB, CTR, CBC, and XTS.

2. Key, plaintext, ciphertext, and initialization vector

- The register se_aes_0_msa stores the address of plaintext or ciphertext.
- The register se_aes_0_msa stores the address of ciphertext or plaintext.
- The register se_aes_0_iv_0~se_aes_0_iv_3 stores IV.
- The register se_aes_0_key_0~se_aes_0_key_7 stores the key.

3. Software and hardware encryption process

- Configure the 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, and se_aes_0_twk_endian

  Note: 0: littleendian 1: bigendian

- Configure se_aes_0_block_mode in the register se_aes_0_ctrlL
- Configure se_aes_0_mode in the register se_aes_0_ctrlL
- Configure se_aes_0_dec_en in the register se_aes_0_ctrlL
- Configure se_aes_0_dec_key_sel 0:new key 1:same key as last one in the register se_aes_0_ctrlL

- Configure se_aes_0_iv_sel 0:new iv 1:same iv as last one in the register se_aes_0_ctrlL
- Configure se_aes_0_en enable aes in the register se_aes_0_ctrlL
- Configure the register se_aes_0_iv_0~se_aes_0_iv_3 set IV.

  Sequence of filling: MSB: se_aes_0_iv_0~se_aes_0_iv_3; LSB: se_aes_0_iv_3~se_aes_0_iv_0
- Configure the register se_aes_0_key_0~se_aes_0_key_7 set key.

  Sequence of filling: MSB: se_aes_0_key_0~se_aes_0_key_7; LSB: se_aes_0_key_7~se_aes_0_key_0.

  Bits for filling: first 4 bits for AES128; first 6 bits for AES196, and first 8 bits for AES256.
- Configure the register se_aes_0_msa set msa addr
- Configure the register se_aes_0_msa set mda addr
- Configure se_aes_0_msg_len set msg len in the register se_aes_0_ctrlL
- Configure se_aes_0_trig_1t Trigger AES Engine in the register se_aes_0_ctrlL
- The output result is stored in the address corresponding to the register se_aes_0_mda

34.4.2 SHA Accelerator

1. SHA accelerator supports 7 standard operations:

- SHA-1、SHA-224、SHA-256、SHA-512、SHA-384、SHA-512/224、SHA-512/256，同时还支持 MD5、CRC16、CRC32。

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

Configure se_sha_0_mode in the register se_sha_0_ctrl to select among different SHA operations, and configure se_sha_0_mode_ext in the register se_sha_0_ctrl to select among MD5, CRC16, and CRC32

- The se_sha_0_mode is valid when se_sha_0_mode_ext is 0.
- The se_sha_0_mode is invalid when se_sha_0_mode_ext is not 0.

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.

Sequence: 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

3. Operation flow

- Configure se_sha_0_mode set SHA operation type in the register se_sha_0_ctrl
- Configure se_sha_0_en enable sha in the register se_sha_0_ctrl
- Configure se_sha_0_hash_sel 0:new hash 1:accumulate last hash in the register se_sha_0_ctrl
- Configure the register se_sha_0_msa set mda addr
- Configure se_sha_0_msg_len set msg len in the register se_sha_0_ctrl
- Configure se_sha_0_trig_1t Trigger SHA Engine in the register se_sha_0_ctrl
- **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

34.4.3 Random Number Generator (RNG)

1. 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 can think 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.

2. Output

- The register SE_TRNG_0_DOUT_0~SE_TRNG_0_DOUT_7 stores the output random number.

3. Usage process

- Configure se_trng_0_en enable trng in the register se_trng_0_ctrl_0
- Configure se_trng_0_trig_1t trigger trng engine in the register se_trng_0_ctrl_0
- The output result is stored in se_trng_0_dout_0~se_trng_0_dout_7

34.4.4 GMAC(link Mode)

1. Definition of GMAC_link_Table Structure

- Word0:
  - [9]:se_gmac_0_int_clr_1t
  - [10]:se_gmac_0_int_set_1t
  - [31:16]:se_gmac_0_msg_len
- Word1:se_gmac_0_msa
- Word2、Word3、Word4、Word5：se_gmac_0_h
- Word6、Word7、Word8、Word9：se_gmac_0_tag

2. Usage process

- Configure se_gmac_0_x_endian, se_gmac_0_h_endian, and se_gmac_0_t_endian 0:littleendian 1:bigendian in the register se_gmac_0_ctrl_0
- Write the start address of the GMAC_link_Table structure into the register se_gmac_0_lca
- Assign the original message address to se_gmac_0_msa in Word1 in GMAC_link_Table structure
- Assign the original message length to se_gmac_0_msg_len in Word0 in GMAC_link_Table structure, where the 128-bit message length corresponds to 1 in se_gmac_0_msg_len
- Configure se_gmac_0_trig_1t trigger gmac engine in the register se_gmac_0_ctrl_0
- The output result is stored in Word6~Word9 in the GMAC_link_Table structure

34.5 Register description

Name	Description
se_sha_0_ctrl	
se_sha_0_msa	
se_sha_0_status	
se_sha_0_endian	
se_sha_0_hash_l_0	
se_sha_0_hash_l_1	
se_sha_0_hash_l_2	
se_sha_0_hash_l_3	

Name	Description
se_sha_0_hash_l_4	
se_sha_0_hash_l_5	
se_sha_0_hash_l_6	
se_sha_0_hash_l_7	
se_sha_0_hash_h_0	
se_sha_0_hash_h_1	
se_sha_0_hash_h_2	
se_sha_0_hash_h_3	
se_sha_0_hash_h_4	
se_sha_0_hash_h_5	
se_sha_0_hash_h_6	
se_sha_0_hash_h_7	
se_sha_0_link	
se_sha_0_ctrl_prot	
se_aes_0_ctrl	
se_aes_0_msa	
se_aes_0_mda	
se_aes_0_status	
se_aes_0_iv_0	
se_aes_0_iv_1	
se_aes_0_iv_2	
se_aes_0_iv_3	
se_aes_0_key_0	
se_aes_0_key_1	
se_aes_0_key_2	
se_aes_0_key_3	
se_aes_0_key_4	
se_aes_0_key_5	
se_aes_0_key_6	

Name	Description
se_aes_0_key_7	
se_aes_0_key_sel	
se_aes_1_key_sel	
se_aes_0_endian	
se_aes_sboot	
se_aes_0_link	
se_aes_0_ctrl_prot	
se_trng_0_ctrl_0	
se_trng_0_status	
se_trng_0_dout_0	
se_trng_0_dout_1	
se_trng_0_dout_2	
se_trng_0_dout_3	
se_trng_0_dout_4	
se_trng_0_dout_5	
se_trng_0_dout_6	
se_trng_0_dout_7	
se_trng_0_test	
se_trng_0_ctrl_1	
se_trng_0_ctrl_2	
se_trng_0_ctrl_3	
se_trng_0_test_out_0	
se_trng_0_test_out_1	
se_trng_0_test_out_2	
se_trng_0_test_out_3	
se_trng_0_ctrl_prot	
se_pka_0_ctrl_0	
se_pka_0_seed	
se_pka_0_ctrl_1	

Name	Description
se_pka_0_rw	
se_pka_0_rw_burst	
se_pka_0_ctrl_prot	
se_cdet_0_ctrl_0	
se_cdet_0_ctrl_1	
se_cdet_0_ctrl_prot	
se_gmac_0_ctrl_0	
se_gmac_0_lca	
se_gmac_0_status	
se_gmac_0_ctrl_prot	
se_ctrl_prot_rd	
se_ctrl_reserved_0	
se_ctrl_reserved_1	
se_ctrl_reserved_2	

34.5.1 se_sha_0_ctrl

Address：0x20004000

se_sha_0_msg_len

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
se_sha_0_link_mode	RSVD	se_sha_0_mode_ext		se_sha_0_int_mask		se_sha_0_int_set_1t		se_sha_0_int_clr_1t		se_sha_0_int	RSVD	se_sha_0_hash_sel		se_sha_0_en	se_sha_0_mode		se_sha_0_trig_1t	se_sha_0_busy	

Bits	Name	Type	Reset	Description
31:16	se_sha_0_msg_len	r/w	0	number of 512-bit block
15	se_sha_0_link_mode	r/w	0	1:enable sha link mode

Bits	Name	Type	Reset	Description
14	RSVD			
13:12	se_sha_0_mode_ext	r/w	0	hash mode extention; 0:SHA 1:MD5 2:CRC-16 3:CRC-32
11	se_sha_0_int_mask	r/w	0	
10	se_sha_0_int_set_1t	w1p	0	1:set interrupt
9	se_sha_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_sha_0_int	r	0	interrupt value
7	RSVD			
6	se_sha_0_hash_sel	r/w	0	0:new hash 1:accumulate last hash
5	se_sha_0_en	r/w	0	1:enable sha engine
4:2	se_sha_0_mode	r/w	0	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
1	se_sha_0_trig_1t	w1p	0	1:trigger sha engine
0	se_sha_0_busy	r	0	1:sha engine busy

34.5.2 se_sha_0_msa

Address：0x20004004

se_sha_0_msa

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

se_sha_0_msa

Bits	Name	Type	Reset	Description
31:0	se_sha_0_msa	r/w	0	message source address

34.5.3 se_sha_0_status

Address：0x20004008

se_sha_0_status

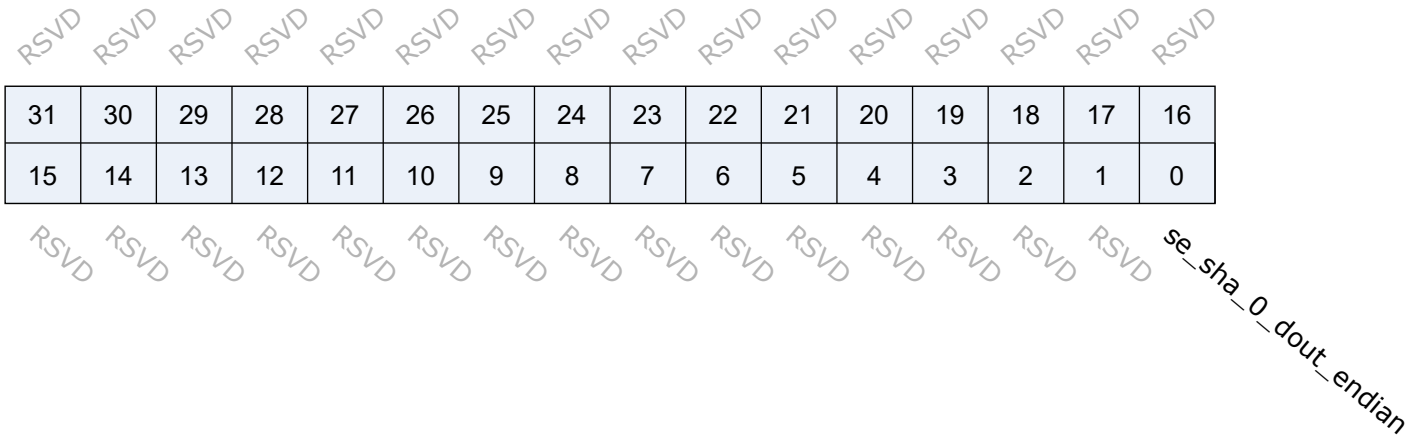
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se_sha_0_status

Bits	Name	Type	Reset	Description
31:0	se_sha_0_status	r	32'h41	

34.5.4 se_sha_0_endian

Address：0x2000400c



Bits	Name	Type	Reset	Description
31:1	RSVD			
0	se_sha_0_dout_endian	r/w	1	0:little-endian 1:big-endian

34.5.5 se_sha_0_hash_l_0

Address：0x20004010

se_sha_0_hash_l_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

se_sha_0_hash_l_0

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_0	r	0	big-endian hash 0 (MSB)

34.5.6 se_sha_0_hash_l_1

Address：0x20004014

se_sha_0_hash_l_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

se_sha_0_hash_l_1

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_1	r	0	big-endian hash 1

34.5.7 se_sha_0_hash_l_2

Address：0x20004018

se_sha_0_hash_l_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

se_sha_0_hash_l_2

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_2	r	0	big-endian hash 2

34.5.8 se_sha_0_hash_l_3

Address：0x2000401c

se_sha_0_hash_l_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

se_sha_0_hash_l_3

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_3	r	0	big-endian hash 3

34.5.9 se_sha_0_hash_l_4

Address：0x20004020

se_sha_0_hash_l_4

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

se_sha_0_hash_l_4

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_4	r	0	big-endian hash 4

34.5.10 se_sha_0_hash_l_5

Address：0x20004024

se_sha_0_hash_l_5

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

se_sha_0_hash_l_5

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_5	r	0	big-endian hash 5

34.5.11 se_sha_0_hash_l_6

Address：0x20004028

se_sha_0_hash_l_6

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

se_sha_0_hash_l_6

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_6	r	0	big-endian hash 6

34.5.12 se_sha_0_hash_l_7

Address：0x2000402c

se_sha_0_hash_l_7

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

se_sha_0_hash_l_7

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_l_7	r	0	big-endian hash 7 (LSB)

34.5.13 se_sha_0_hash_h_0

Address：0x20004030

se_sha_0_hash_h_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

se_sha_0_hash_h_0

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_0	r	0	big-endian hash 0 (MSB)

34.5.14 se_sha_0_hash_h_1

Address：0x20004034

se_sha_0_hash_h_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

se_sha_0_hash_h_1

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_1	r	0	big-endian hash 1

34.5.15 se_sha_0_hash_h_2

Address：0x20004038

se_sha_0_hash_h_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

se_sha_0_hash_h_2

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_2	r	0	big-endian hash 2

34.5.16 se_sha_0_hash_h_3

Address：0x2000403c

se_sha_0_hash_h_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

se_sha_0_hash_h_3

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_3	r	0	big-endian hash 3

34.5.17 se_sha_0_hash_h_4

Address：0x20004040

se_sha_0_hash_h_4

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

se_sha_0_hash_h_4

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_4	r	0	big-endian hash 4

34.5.18 se_sha_0_hash_h_5

Address：0x20004044

se_sha_0_hash_h_5

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

se_sha_0_hash_h_5

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_5	r	0	big-endian hash 5

34.5.19 se_sha_0_hash_h_6

Address：0x20004048

se_sha_0_hash_h_6

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

se_sha_0_hash_h_6

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_6	r	0	big-endian hash 6

34.5.20 se_sha_0_hash_h_7

Address：0x2000404c

se_sha_0_hash_h_7

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

se_sha_0_hash_h_7

Bits	Name	Type	Reset	Description
31:0	se_sha_0_hash_h_7	r	0	big-endian hash 7 (LSB)

34.5.21 se_sha_0_link

Address：0x20004050

se_sha_0_lca

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

se_sha_0_lca

Bits	Name	Type	Reset	Description
31:0	se_sha_0_lca	r/w	0	aes link config address(word align)

34.5.22 se_sha_0_ctrl_prot

Address：0x200040fc

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	se_sha_id1_en	se_sha_id0_en	RSVD

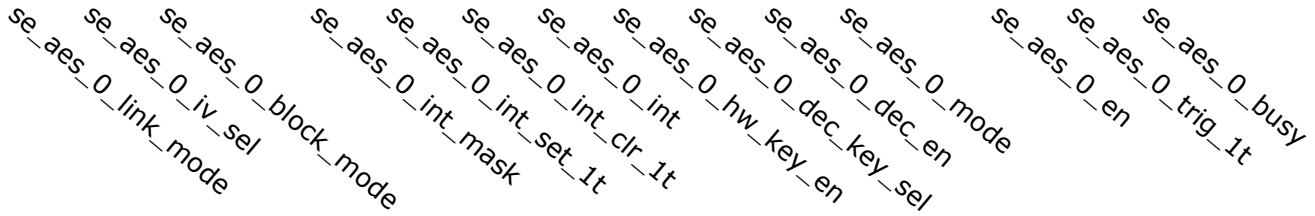
Bits	Name	Type	Reset	Description
31:3	RSVD			
2	se_sha_id1_en	r/w	1	id1 access right
1	se_sha_id0_en	r/w	1	id0 access right
0	RSVD			

34.5.23 se_aes_0_ctrl

Address：0x20004100

se_aes_0_msg_len

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



Bits	Name	Type	Reset	Description
31:16	se_aes_0_msg_len	r/w	0	number of 128-bit block
15	se_aes_0_link_mode	r/w	0	1:enable aes link mode
14	se_aes_0_iv_sel	r/w	0	0:new iv 1:same iv as last one
13:12	se_aes_0_block_mode	r/w	0	0:ECB mode 1:CTR mode 2:CBC mode 3:XTS mode
11	se_aes_0_int_mask	r/w	0	
10	se_aes_0_int_set_1t	w1p	0	1:set interrupt
9	se_aes_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_aes_0_int	r	0	interrupt value
7	se_aes_0_hw_key_en	r/w	0	0:sw key 1:hw key
6	se_aes_0_dec_key_sel	r/w	0	0:new key 1:same key as last one
5	se_aes_0_dec_en	r/w	0	0:encode 1:decode
4:3	se_aes_0_mode	r/w	0	0:128-bit mode 1:256-bit mode 2:192-bit mode 3:128-bit double key mode
2	se_aes_0_en	r/w	0	0:disable 1:enable aes
1	se_aes_0_trig_1t	w1p	0	1:trigger aes engine
0	se_aes_0_busy	r	0	1:aes engine busy

34.5.24 se_aes_0_msa

Address：0x20004104

se_aes_0_msa

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

se_aes_0_msa

Bits	Name	Type	Reset	Description
31:0	se_aes_0_msa	r/w	0	message source address

34.5.25 se_aes_0_mda

Address：0x20004108

se_aes_0_mda

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

se_aes_0_mda

Bits	Name	Type	Reset	Description
31:0	se_aes_0_mda	r/w	0	message destination address

34.5.26 se_aes_0_status

Address：0x2000410c

se_aes_0_status

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

se_aes_0_status

Bits	Name	Type	Reset	Description
31:0	se_aes_0_status	r	32'h4100	

34.5.27 se_aes_0_iv_0

Address：0x20004110

se_aes_0_iv_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

se_aes_0_iv_0

Bits	Name	Type	Reset	Description
31:0	se_aes_0_iv_0	r/w	0	big endian initial vector (MSB)

34.5.28 se_aes_0_iv_1

Address：0x20004114

se_aes_0_iv_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

se_aes_0_iv_1

Bits	Name	Type	Reset	Description
31:0	se_aes_0_iv_1	r/w	0	big endian initial vector

34.5.29 se_aes_0_iv_2

Address：0x20004118

se_aes_0_iv_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

se_aes_0_iv_2

Bits	Name	Type	Reset	Description
31:0	se_aes_0_iv_2	r/w	0	big endian initial vector

34.5.30 se_aes_0_iv_3

Address：0x2000411c

se_aes_0_iv_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

se_aes_0_iv_3

Bits	Name	Type	Reset	Description
31:0	se_aes_0_iv_3	r/w	0	big endian initial vector (LSB) (CTR mode: 32-bit counter initial value)

34.5.31 se_aes_0_key_0

Address：0x20004120

se_aes_0_key_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

se_aes_0_key_0

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_0	r/w	0	big endian aes key (aes-128/256 key MSB)

34.5.32 se_aes_0_key_1

Address：0x20004124

se_aes_0_key_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

se_aes_0_key_1

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_1	r/w	0	big endian aes key

34.5.33 se_aes_0_key_2

Address：0x20004128

se_aes_0_key_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

se_aes_0_key_2

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_2	r/w	0	big endian aes key

34.5.34 se_aes_0_key_3

Address：0x2000412c

se_aes_0_key_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

se_aes_0_key_3

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_3	r/w	0	big endian aes key (aes-128 key LSB)

34.5.35 se_aes_0_key_4

Address：0x20004130

se_aes_0_key_4

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

se_aes_0_key_4

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_4	r/w	0	big endian aes key

34.5.36 se_aes_0_key_5

Address：0x20004134

se_aes_0_key_5

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

se_aes_0_key_5

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_5	r/w	0	big endian aes key

34.5.37 se_aes_0_key_6

Address：0x20004138

se_aes_0_key_6

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

se_aes_0_key_6

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_6	r/w	0	big endian aes key

34.5.38 se_aes_0_key_7

Address：0x2000413c

se_aes_0_key_7

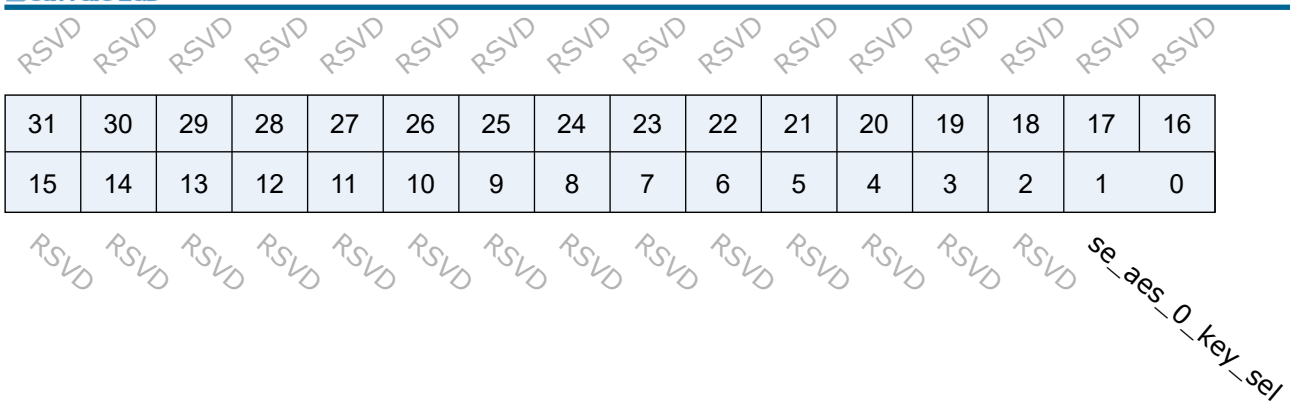
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se_aes_0_key_7

Bits	Name	Type	Reset	Description
31:0	se_aes_0_key_7	r/w	0	big endian aes key (aes-256 key LSB)

34.5.39 se_aes_0_key_sel

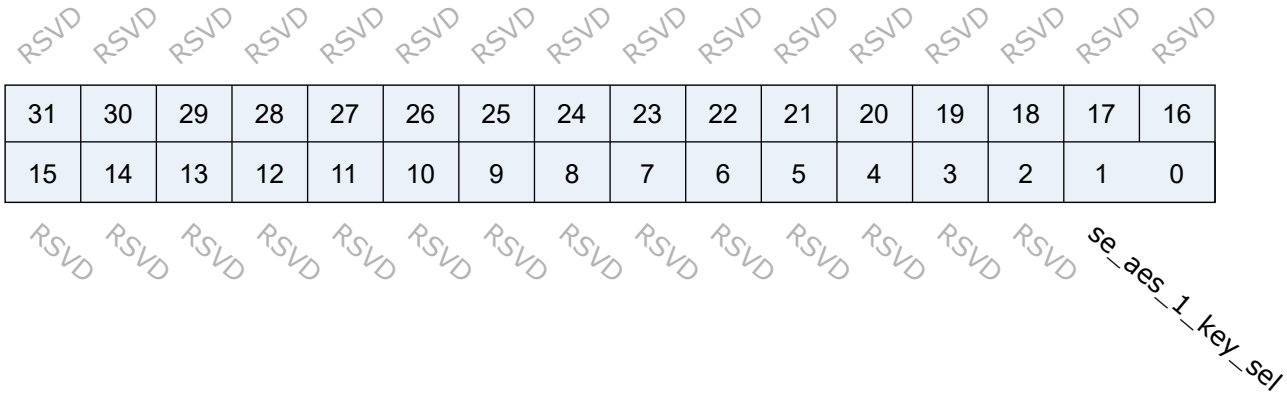
Address：0x20004140



Bits	Name	Type	Reset	Description
31:2	RSVD			
1:0	se_aes_0_key_sel	r/w	0	

34.5.40 se_aes_1_key_sel

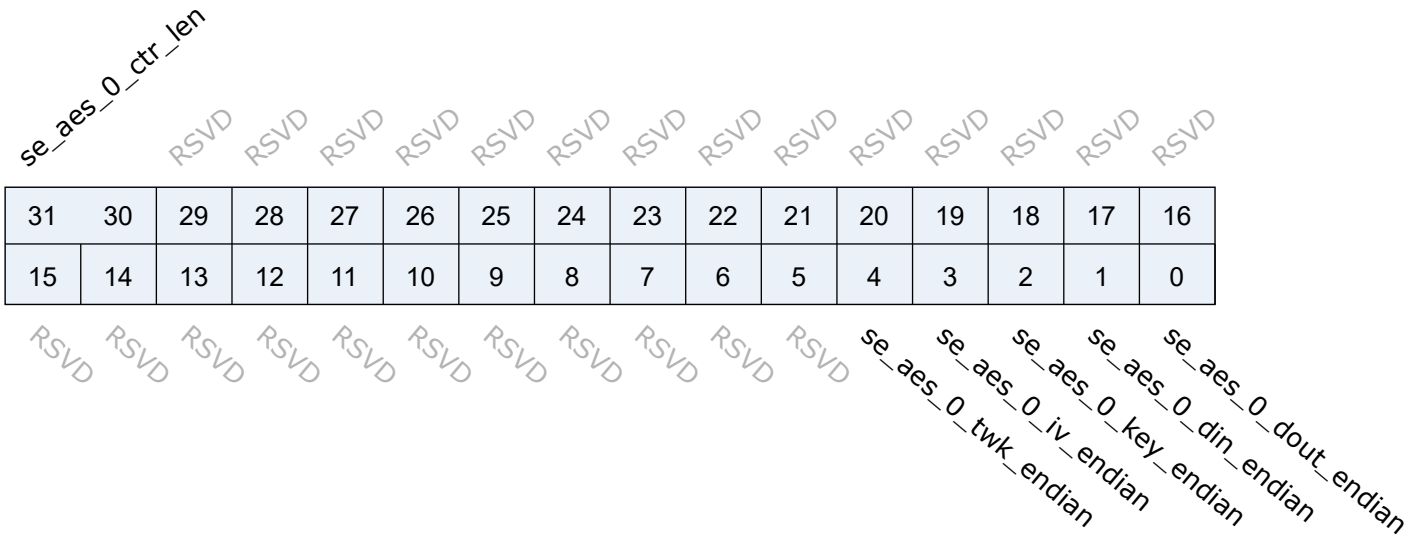
Address：0x20004144



Bits	Name	Type	Reset	Description
31:2	RSVD			
1:0	se_aes_1_key_sel	r/w	0	

34.5.41 se_aes_0_endian

Address：0x20004148

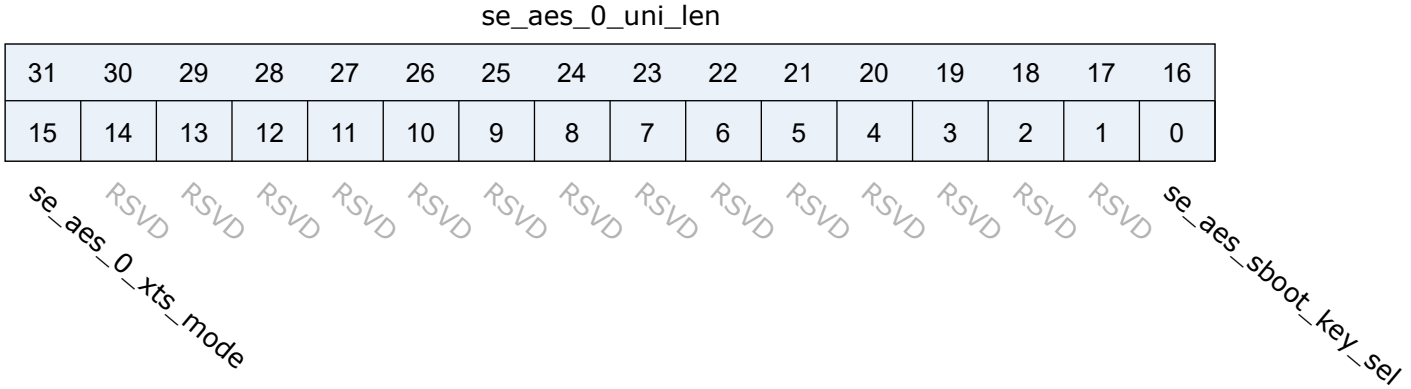


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Reset	Description
31:30	se_aes_0_ctr_len	r/w	0	2'd0:4-byte counter, 2'd1:1-byte counter, 2'd2:2-byte counter, 2'd3:3-byte counter
29:5	RSVD			
4	se_aes_0_twk_endian	r/w	1	0:little-endian 1:big-endian, default 1 for XTS
3	se_aes_0_iv_endian	r/w	1	0:little-endian 1:big-endian
2	se_aes_0_key_endian	r/w	1	0:little-endian 1:big-endian
1	se_aes_0_din_endian	r/w	1	0:little-endian 1:big-endian
0	se_aes_0_dout_endian	r/w	1	0:little-endian 1:big-endian

34.5.42 se_aes_sboot

Address：0x2000414c



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

Bits	Name	Type	Reset	Description
31:16	se_aes_0_uni_len	r/w	16'd2	XTS data unit length: number of 128-bit blocks in a data unit, msg_len = N*uni_len
15	se_aes_0_xts_mode	r/w	0	0: normal XTS, 1: XTS with only one data unit
14:1	RSVD			
0	se_aes_sboot_key_sel	r/w	0	

34.5.43 se_aes_0_link

Address：0x20004150

se_aes_0_lca

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

se_aes_0_lca

Bits	Name	Type	Reset	Description
31:0	se_aes_0_lca	r/w	0	aes link config address(word align)

34.5.44 se_aes_0_ctrl_prot

Address：0x200041fc

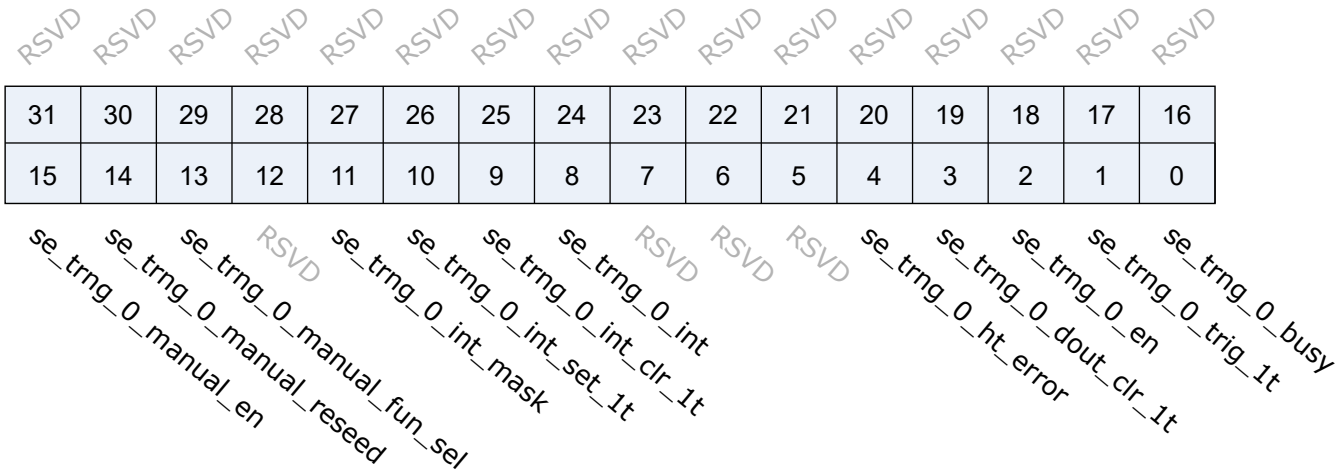
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	se_aes_id1_en	se_aes_id0_en	RSVD

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	se_aes_id1_en	r/w	1	id1 access right
1	se_aes_id0_en	r/w	1	id0 access right

Bits	Name	Type	Reset	Description
0	RSVD			

34.5.45 se_trng_0_ctrl_0

Address：0x20004200



Bits	Name	Type	Reset	Description
31:16	RSVD			
15	se_trng_0_manual_en	r/w	0	1:enable manual mode
14	se_trng_0_manual_reseed	r/w	0	1:clear reseed counter to zero and get new entropy
13	se_trng_0_manual_fun_sel	r/w	0	0:go to instantiate state 1:go to generate state
12	RSVD			
11	se_trng_0_int_mask	r/w	0	
10	se_trng_0_int_set_1t	w1p	0	1:set interrupt
9	se_trng_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_trng_0_int	r	0	interrupt value
7:5	RSVD			
4	se_trng_0_ht_error	r	0	1:health test error
3	se_trng_0_dout_clr_1t	w1p	0	1:clear trng dout to zero
2	se_trng_0_en	r/w	0	0:disable 1:enable aes
1	se_trng_0_trig_1t	w1p	0	1:trigger trng engine
0	se_trng_0_busy	r	0	1:trng engine busy

34.5.46 se_trng_0_status

Address：0x20004204

se_trng_0_status

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

se_trng_0_status

Bits	Name	Type	Reset	Description
31:0	se_trng_0_status	r	32'h100020	

34.5.47 se_trng_0_dout_0

Address：0x20004208

se_trng_0_dout_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

se_trng_0_dout_0

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_0	r	0	random value

34.5.48 se_trng_0_dout_1

Address：0x2000420c

se_trng_0_dout_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

se_trng_0_dout_1

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_1	r	0	random value

34.5.49 se_trng_0_dout_2

Address：0x20004210

se_trng_0_dout_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

se_trng_0_dout_2

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_2	r	0	random value

34.5.50 se_trng_0_dout_3

Address：0x20004214

se_trng_0_dout_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

se_trng_0_dout_3

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_3	r	0	random value

34.5.51 se_trng_0_dout_4

Address：0x20004218

se_trng_0_dout_4

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

se_trng_0_dout_4

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_4	r	0	random value

34.5.52 se_trng_0_dout_5

Address：0x2000421c

se_trng_0_dout_5

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

se_trng_0_dout_5

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_5	r	0	random value

34.5.53 se_trng_0_dout_6

Address：0x20004220

se_trng_0_dout_6

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

se_trng_0_dout_6

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_6	r	0	random value

34.5.54 se_trng_0_dout_7

Address：0x20004224

se_trng_0_dout_7

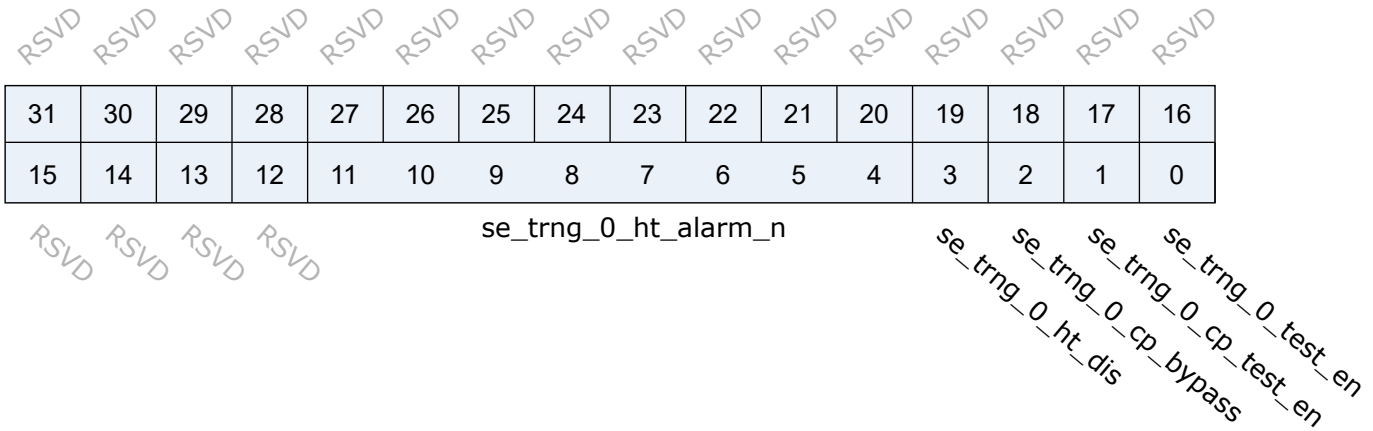
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se_trng_0_dout_7

Bits	Name	Type	Reset	Description
31:0	se_trng_0_dout_7	r	0	random value

34.5.55 se_trng_0_test

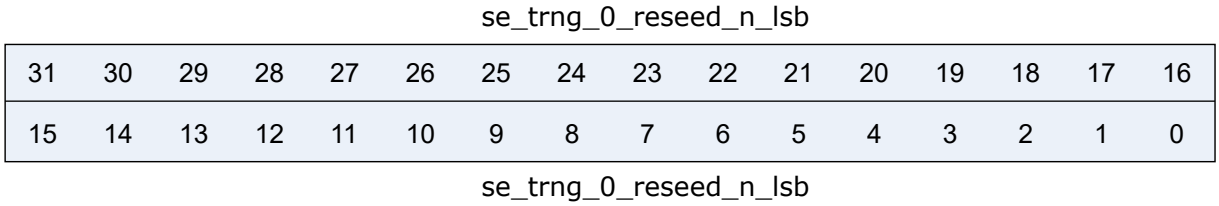
Address：0x20004228



Bits	Name	Type	Reset	Description
31:12	RSVD			
11:4	se_trng_0_ht_alarm_n	r/w	8'd0	health test alarm number 0:alarm if health test error >= 1 1:alarm if health test error >= 2
3	se_trng_0_ht_dis	r/w	0	1:disable health test
2	se_trng_0_cp_bypass	r/w	0	1:bypass conditional component
1	se_trng_0_cp_test_en	r/w	0	1:enable trng conditional component test mode
0	se_trng_0_test_en	r/w	0	1:enable trng test mode

34.5.56 se_trng_0_ctrl_1

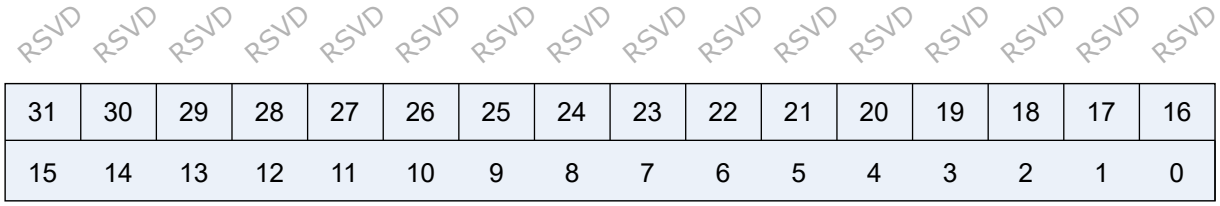
Address：0x2000422c



Bits	Name	Type	Reset	Description
31:0	se_trng_0_reseed_n_lsb	r/w	32'hffff	reload seed when number of used random value is larger than reseed_n

34.5.57 se_trng_0_ctrl_2

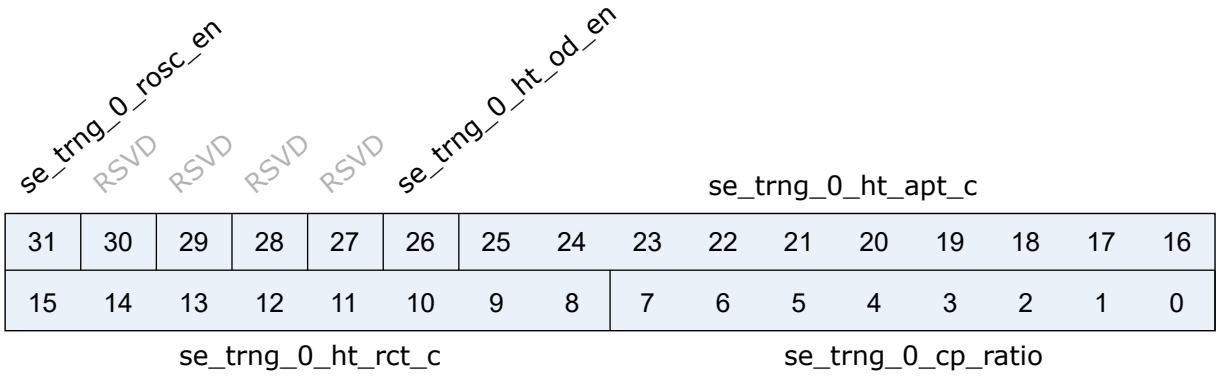
Address：0x20004230



Bits	Name	Type	Reset	Description
31:16	RSVD			
15:0	se_trng_0_reseed_n_msb	r/w	16'hff	reload seed when number of used random value is larger than reseed_n

34.5.58 se_trng_0_ctrl_3

Address：0x20004234



Bits	Name	Type	Reset	Description
31	se_trng_0_rosc_en	r/w	0	trng rosc enable
30:27	RSVD			
26	se_trng_0_ht_od_en	r/w	0	health test on demand test enable
25:16	se_trng_0_ht_apt_c	r/w	10'd890	health test adaptive proportion test cut off value
15:8	se_trng_0_ht_rct_c	r/w	8'd66	health test repetition count test cut off value
7:0	se_trng_0_cp_ratio	r/w	8'd3	conditional component compression ration

34.5.59 se_trng_0_test_out_0

Address：0x20004240

se_trng_0_test_out_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

se_trng_0_test_out_0

Bits	Name	Type	Reset	Description
31:0	se_trng_0_test_out_0	r	0	

34.5.60 se_trng_0_test_out_1

Address：0x20004244

se_trng_0_test_out_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

se_trng_0_test_out_1

Bits	Name	Type	Reset	Description
31:0	se_trng_0_test_out_1	r	0	

34.5.61 se_trng_0_test_out_2

Address：0x20004248

se_trng_0_test_out_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

se_trng_0_test_out_2

Bits	Name	Type	Reset	Description
31:0	se_trng_0_test_out_2	r	0	

34.5.62 se_trng_0_test_out_3

Address：0x2000424c

se_trng_0_test_out_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

se_trng_0_test_out_3

Bits	Name	Type	Reset	Description
31:0	se_trng_0_test_out_3	r	0	

34.5.63 se_trng_0_ctrl_prot

Address：0x200042fc

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	se_trng_id1_en	se_trng_id0_en	RSVD

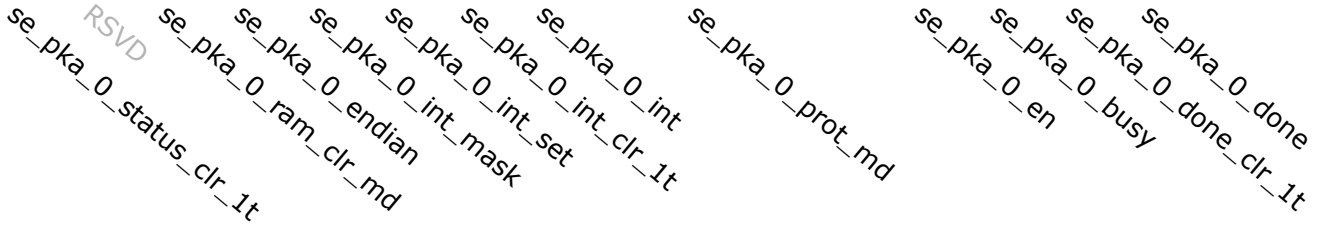
Bits	Name	Type	Reset	Description
31:3	RSVD			
2	se_trng_id1_en	r/w	1	id1 access right
1	se_trng_id0_en	r/w	1	id0 access right
0	RSVD			

34.5.64 se_pka_0_ctrl_0

Address：0x20004300

se_pka_0_status

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



Bits	Name	Type	Reset	Description
31:16	se_pka_0_status	r	0	[31]cmd_err, [30:26]cmd_err_idx[4:0] [25]opq_full, [24]last_opc, [23]err_cam_full, [22]err_div_by_0, [21]err_invalid_src0 [20]err_invalid_src1 [19]err_invalid_src2 [18]err_opq_overflow [17]err_unknown_opc [16]prime_fail
15	se_pka_0_status_clr_1t	w1p	0	
14	RSVD			
13	se_pka_0_ram_clr_md	r/w	0	
12	se_pka_0_endian	r/w	0	
11	se_pka_0_int_mask	r/w	0	
10	se_pka_0_int_set	r/w	0	1:set interrupt
9	se_pka_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_pka_0_int	r	0	interrupt value
7:4	se_pka_0_prot_md	r/w	0	
3	se_pka_0_en	r/w	0	
2	se_pka_0_busy	r	0	
1	se_pka_0_done_clr_1t	w1p	0	
0	se_pka_0_done	r	0	

34.5.65 se_pka_0_seed

Address：0x2000430c

se_pka_0_seed

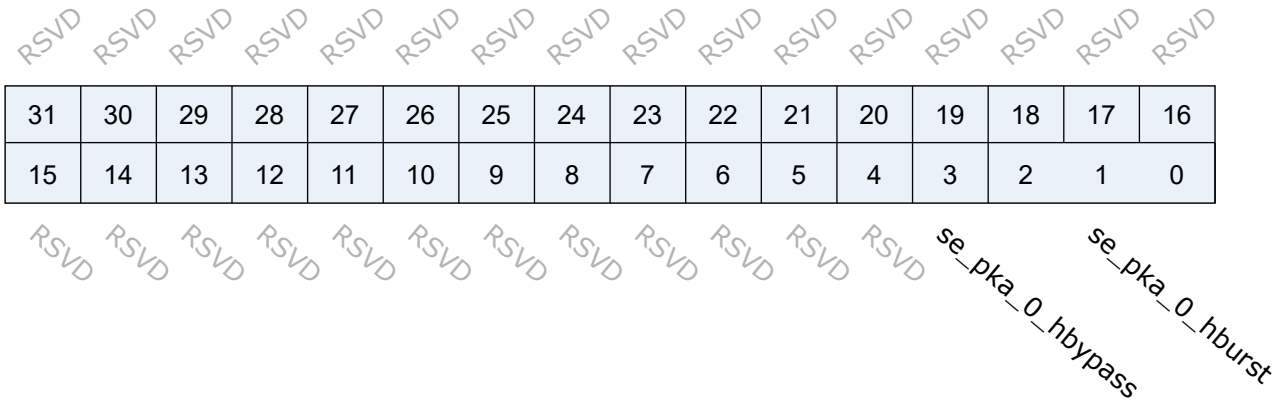
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

se_pka_0_seed

Bits	Name	Type	Reset	Description
31:0	se_pka_0_seed	r/w	0	

34.5.66 se_pka_0_ctrl_1

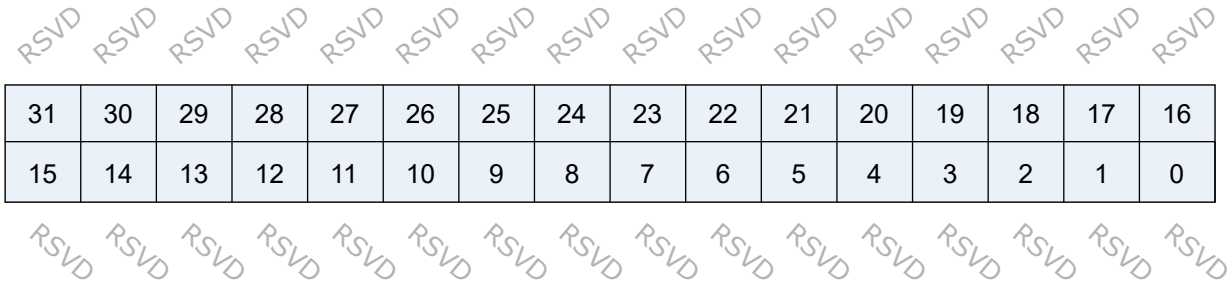
Address：0x20004310



Bits	Name	Type	Reset	Description
31:4	RSVD			
3	se_pka_0_hbypass	r/w	0	
2:0	se_pka_0_hburst	r/w	3'd5	3'b000:single 3'b001:incr (undefined length) 3'b010:4-beat wrap 3'b011:4-beat incr 3'b100:8-beat wrap 3'b101:8-beat incr(default)

34.5.67 se_pka_0_rw

Address：0x20004340

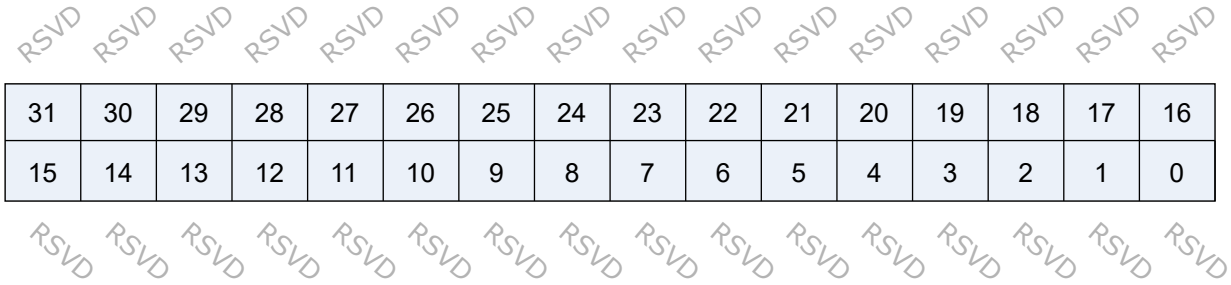


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Reset	Description
31:0	se_pka_0_rw	r/w	0	0x340 0x35F single write for command

34.5.68 se_pka_0_rw_burst

Address：0x20004360

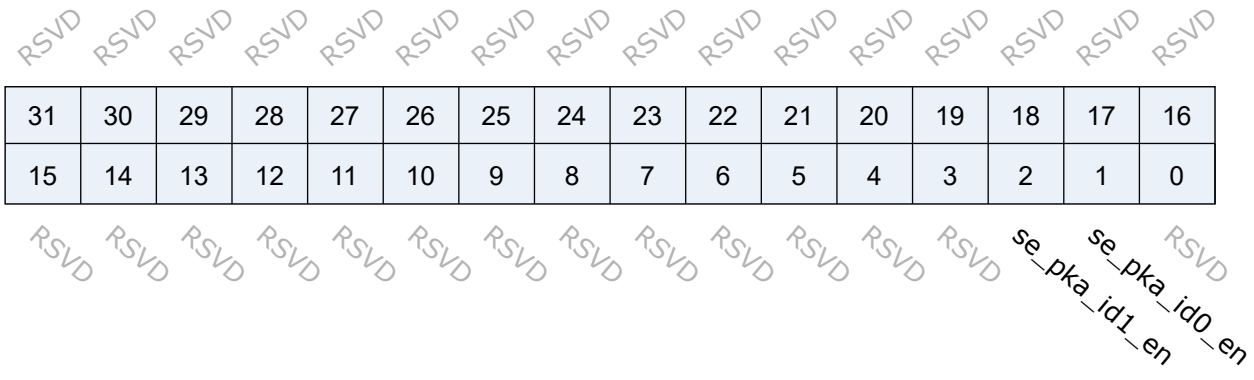


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Reset	Description
31:0	se_pka_0_rw_burst	r/w	0	0x360 0x37F burst write for data

34.5.69 se_pka_0_ctrl_prot

Address：0x200043fc

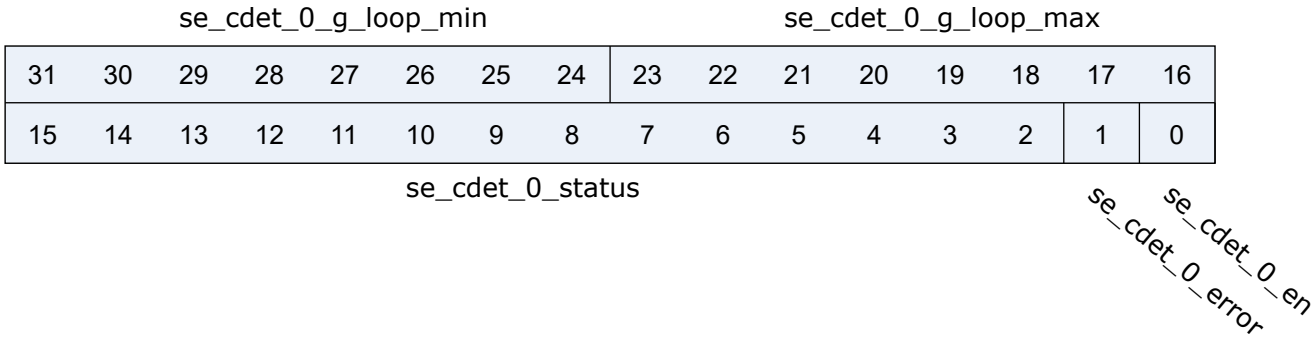


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	se_pka_id1_en	r/w	1	id1 access right
1	se_pka_id0_en	r/w	1	id0 access right
0	RSVD			

34.5.70 se_cdet_0_ctrl_0

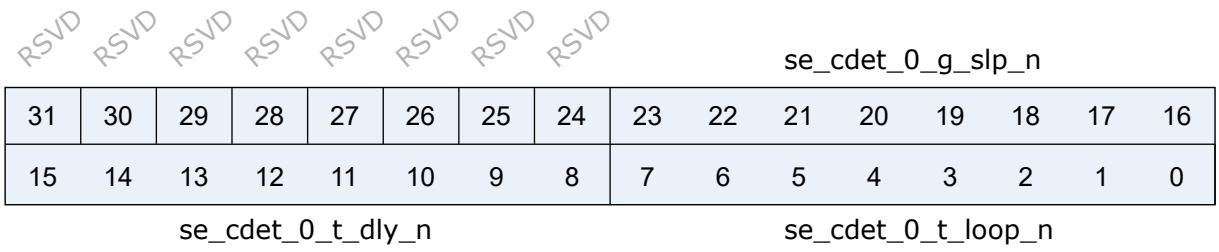
Address：0x20004400



Bits	Name	Type	Reset	Description
31:24	se_cdet_0_g_loop_min	r/w	8’d33	
23:16	se_cdet_0_g_loop_max	r/w	8’d100	
15:2	se_cdet_0_status	r	1	
1	se_cdet_0_error	r	0	
0	se_cdet_0_en	r/w	0	

34.5.71 se_cdet_0_ctrl_1

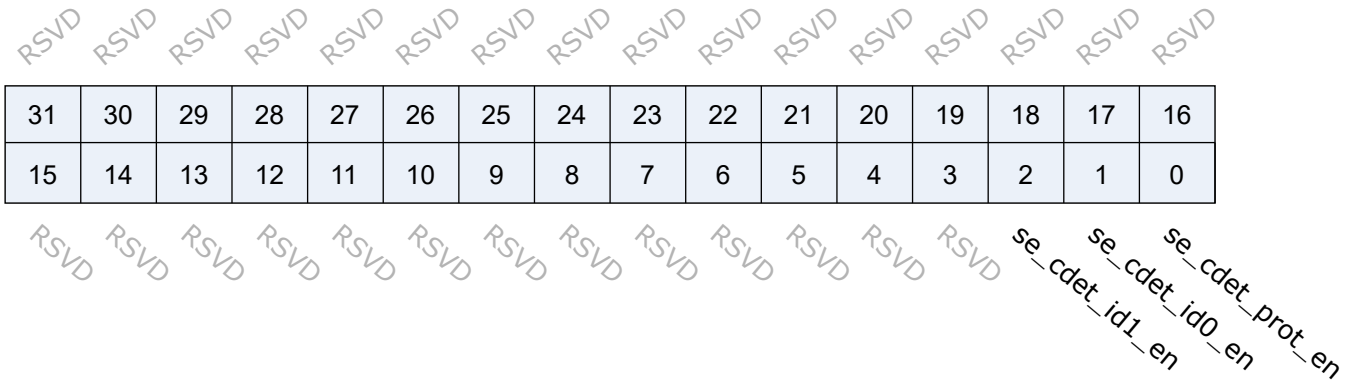
Address：0x20004404



Bits	Name	Type	Reset	Description
31:24	RSVD			
23:16	se_cdet_0_g_slp_n	r/w	8'd255	
15:8	se_cdet_0_t_dly_n	r/w	8'd3	
7:0	se_cdet_0_t_loop_n	r/w	8'd50	

34.5.72 se_cdet_0_ctrl_prot

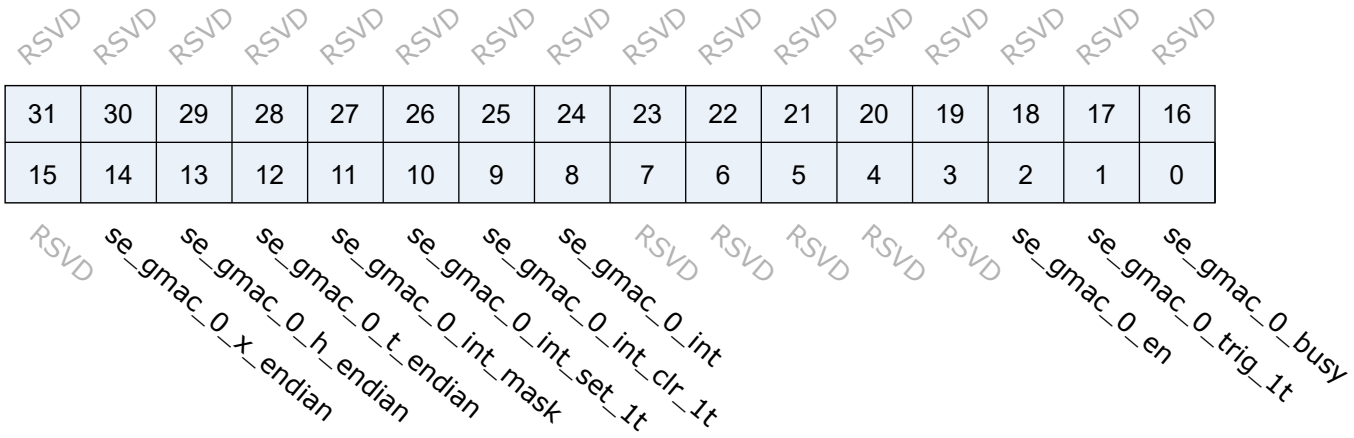
Address：0x200044fc



Bits	Name	Type	Reset	Description
31:3	RSVD			
2	se_cdet_id1_en	r/w	1	id1 access right
1	se_cdet_id0_en	r/w	1	id0 access right
0	se_cdet_prot_en	r/w	1	1:control register protection enable

34.5.73 se_gmac_0_ctrl_0

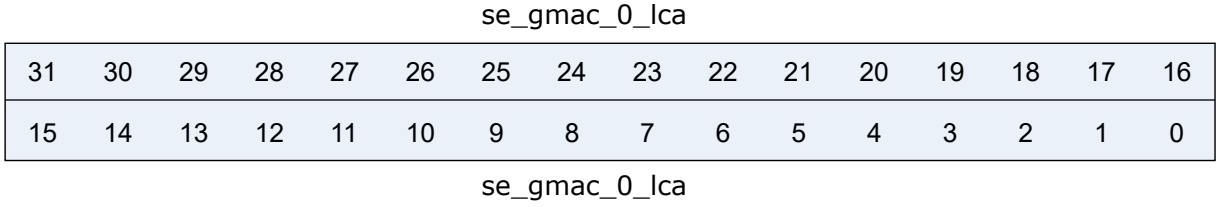
Address：0x20004500



Bits	Name	Type	Reset	Description
31:15	RSVD			
14	se_gmac_0_x_endian	r/w	1	0:little-endian 1:big-endian
13	se_gmac_0_h_endian	r/w	1	0:little-endian 1:big-endian
12	se_gmac_0_t_endian	r/w	1	0:little-endian 1:big-endian
11	se_gmac_0_int_mask	r/w	0	1:mask interrupt
10	se_gmac_0_int_set_1t	w1p	0	1:set interrupt
9	se_gmac_0_int_clr_1t	w1p	0	1:clear interrupt
8	se_gmac_0_int	r	0	interrupt value
7:3	RSVD			
2	se_gmac_0_en	r/w	0	0:disable 1:enable gmac
1	se_gmac_0_trig_1t	w1p	0	1:trigger gmac engine
0	se_gmac_0_busy	r	0	1:gmac engine busy

34.5.74 se_gmac_0_lca

Address：0x20004504



Bits	Name	Type	Reset	Description
31:0	se_gmac_0_lca	r/w	0	gmac link config address(word align)

34.5.75 se_gmac_0_status

Address：0x20004508

se_gmac_0_status

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

se_gmac_0_status

Bits	Name	Type	Reset	Description
31:0	se_gmac_0_status	r	32'hf1000000	

34.5.76 se_gmac_0_ctrl_prot

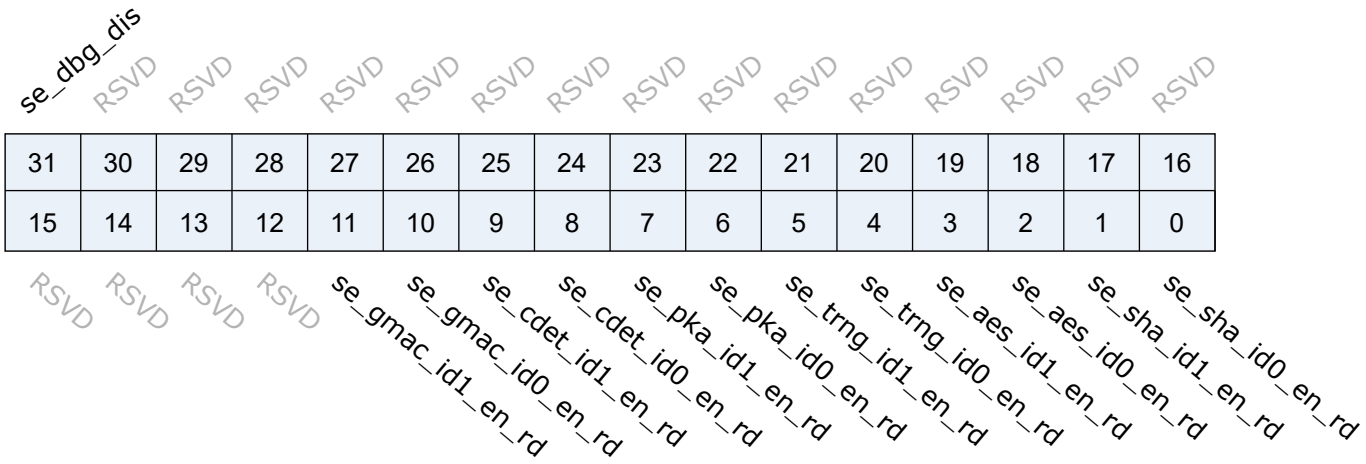
Address：0x200045fc

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	se_gmac_id1_en	se_gmac_id0_en	RSVD

Bits	Name	Type	Reset	Description
31:3	RSVD			
2	se_gmac_id1_en	r/w	1	id1 access right
1	se_gmac_id0_en	r/w	1	id0 access right
0	RSVD			

34.5.77 se_ctrl_prot_rd

Address：0x20004f00



Bits	Name	Type	Reset	Description
31	se_dbg_dis	r	0	1:disable aes debug mode
30:12	RSVD			
11	se_gmac_id1_en_rd	r	1	read only status of id1 access right
10	se_gmac_id0_en_rd	r	1	read only status of id0 access right
9	se_cdet_id1_en_rd	r	1	read only status of id1 access right
8	se_cdet_id0_en_rd	r	1	read only status of id0 access right
7	se_pka_id1_en_rd	r	1	read only status of id1 access right
6	se_pka_id0_en_rd	r	1	read only status of id0 access right
5	se_trng_id1_en_rd	r	1	read only status of id1 access right
4	se_trng_id0_en_rd	r	1	read only status of id0 access right
3	se_aes_id1_en_rd	r	1	read only status of id1 access right
2	se_aes_id0_en_rd	r	1	read only status of id0 access right
1	se_sha_id1_en_rd	r	1	read only status of id1 access right
0	se_sha_id0_en_rd	r	1	read only status of id0 access right

34.5.78 se_ctrl_reserved_0

Address：0x20004f04

se_ctrl_reserved_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

se_ctrl_reserved_0

Bits	Name	Type	Reset	Description
31:0	se_ctrl_reserved_0	r/w	0	

34.5.79 se_ctrl_reserved_1

Address：0x20004f08

se_ctrl_reserved_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

se_ctrl_reserved_1

Bits	Name	Type	Reset	Description
31:0	se_ctrl_reserved_1	r/w	32'hffffffff	

34.5.80 se_ctrl_reserved_2

Address：0x20004f0c

se_ctrl_reserved_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

se_ctrl_reserved_2

Bits	Name	Type	Reset	Description
31:0	se_ctrl_reserved_2	r	0	

35 Revision history

Table 35.1: Document revision history

Date	Revision	Changes
2021/2/8	0.9	Initial release
2022/9/13	1.0	Add register description