Datasheet for Telink

Multi-Standard Wireless SoC

TL721x

DS-TL721x-E15

Ver 0.8.4

2025/12/18

Keyword

Bluetooth® LE; Zigbee; Thread; Matter; 2.4 GHz

Brief

This datasheet is dedicated for Telink multi-standard wireless SoC TL7218A/D/H/J and TL7215A/D.

In this datasheet, function block diagram, key features, electrical specifications, and typical applications of the TL7218A/D/H/J and TL7215A/D are introduced.

Published by

Telink Semiconductor

10-11/F, Building 1, 61 Shengxia Road,

Pudong District, Shanghai, China 201203

Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Information

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinksales@telink-semi.com

telinksupport@telink-semi.com

Revision History

Version	Change Description
0.1.0	Preliminary release
0.2.0	Updated 1.1 Block Diagram, 1.2 Key Features
0.2.1	Updated Figure 1-1, added external flash size and LSPI, changed USB 2.0 to Full Speed.
0.2.2	Minor edits and corrections
0.2.3	Updated 1.2.2 CPU and Memory, 1.2.5 Features of Power Management Module, Table 1-1 Ordering Information of TL721x Added Table 1-12 to Table 1-38
0.5.0	Added chapter 2 to 25
0.5.1	Update ordering part number, re-arrange the chapter sequence, other minor corrections
0.5.2	Updated 4.5.4 VBAT and VANT Power-Supply Mode, 5.3.2 I2S Protocol Added 11.3 JTAG and SDP Other minor corrections
0.5.3	Added TL7218DE11T38R, TL7217AE10T68R, TL7217DE10T38R Updated 1.4 Ordering Information, Figure 1-10 Pin Assignment of TL7218A, Table 1-8 Pin Function of TL7218A, Table 1-9 GPIO Pin Mux of TL7218A, Table 1-13 GPIO Pin Mux of TL7218H, Table 1-15 GPIO Pin Mux of TL7218J, Table 2-3 RX/TX Current and Sleep Current, Table 2-11 Analog Microphone / Line Input to ADC Path, Chapter 3 Reference Design, 4.5.3 LDO and DCDC, 5.2 Audio System Clock, 5.6.3.1 Sample Rate of CODEC Input
0.5.4	Change TL7217AE10T68R, TL7217DE10T38R to TL7215AE10T68R, TL7215DE10T38R
0.8.0	Added 2.5 I2C Timing Characteristics, 12.1 Overview of IR, 12.2 IR Analog (RX and TX circuit) Updated 1.6.1 Pin Assignment of TL7218A, 1.6.2 Pin Assignment of TL7218D, 1.6.5 Pin Assignment of TL7215A, 1.6.3 Pin Assignment of TL7218H, 1.6.6 Pin Assignment of TL7215D, Table 2-3 RX/TX Current and Sleep Current, Table 2-10 ADC Characteristics, Chapter 3 Reference Design, 11.4.1 Introduction of I2C, pictures in Chapter 13 Return to Zero (RZ)
0.8.1	Added note in Chapter 3 Reference Design, 4.6 Wakeup Source Updated 1.2.5 Features of Power Management Module, 1.6.1 Pin Assignment of TL7218A, 1.6.5 Pin Assignment of TL7215A, Table 2-1 Absolute Maximum Rating, Table 2-2 Recommend Operating Conditions, Table 2-8 Crystal Characteristics, 3.1 Reference Schematic of TL7218A, 3.9 Reference Schematic of TL7215A, Figure 12-1 Block Diagram of IR module, 17.4 Battery Voltage Sampling
0.8.2	Added note in Table 1-13 GPIO Pin Mux of TL7218H, Table 11-1 GPIO Pad Function Mux Updated 1.2.3 RF Features, 2.4.1 RF Performance, Table 16-1 PWM Registers, 19.3 Select Mode and Input Channel for Reference
0.8.3	Added note for PD[5], PD[6] and PE[1] to PE[5]; added note for GSPI and LSPI in 1.2.1 General Features Updated Figure 5-16 Audio CODEC Decimation

Version	Change Description
0.8.4	Updated Figure 3-1 Reference Schematic of TL7218A, Figure 3-2 Reference Schematic of TL7218D, Figure 3-3 Reference Schematic of TL7218H, Figure 3-5 Reference Schematic of TL7215A, Figure 3-6 Reference Schematic of TL7215D Other minor corrections

Table of Contents

List of Figures

List of Tables

1 Overview

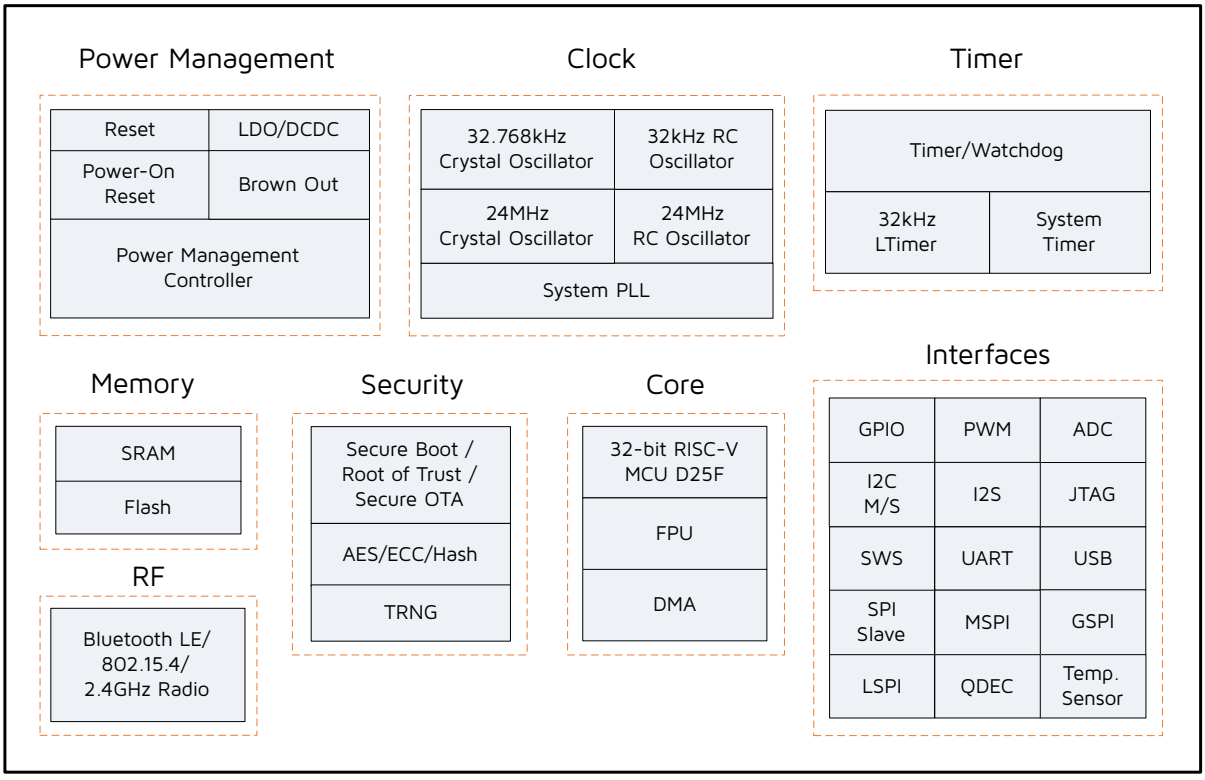
The TL721x (including TL7218A/D/H/J and TL7215A/D) is a single chip SoC for Bluetooth low energy (LE) and 802.15.4. The embedded 2.4GHz transceiver supports Bluetooth low energy, 802.15.4 as well as 2.4GHz proprietary operation. The TL721x supports standards and industrial alliance specifications including Bluetooth LE, Zigbee, Thread, Matter, 2.4GHz proprietary standard.

The TL721x combines the features and functions needed for high quality wireless IoT equipments into a single System on Chip.

1.1 Block Diagram

The TL721x is designed to offer high integration, ultra-low power application capabilities. The system's block diagram is as shown in the following figure.

Figure 1-1 Block Diagram of the System



The TL721x integrates a powerful 32-bit RISC-V MCU, 512 or 256 KB SRAM including up to 256 KB retention SRAM, 2 MB (2048 KB) or 1 MB (1024 KB) embedded flash (depending on detailed parts, see section 1.4 Ordering Information), 12-bit ADC, PWM, flexible IO interfaces, and other peripheral blocks required for IoT applications.

With the high integration level of TL721x, few external components are needed to satisfy customers' ultra-low cost requirements.

1.2 Key Features

1.2.1 General Features

General features are as follows:

1. Supports 128-bit Unique ID (UID)
2. Security solution
   - Root of Trust
   - Secure Boot
   - Secure OTA
   - Firmware encryption
   - Secure Debug Port Control
   - Signature and verification based on RSA2048 or ECC256
   - Embedded hardware AES block cipher with 128-bit & 256-bit keys and software AES-CCM
   - Embedded hardware acceleration for elliptical curve cryptography (ECC) including ECC-P192, ECC-P256, ECC-P384, ECC-P512, ECDSA+ECDH, ED25519, and Curve25519
   - Embedded hardware acceleration for Hash function including SHA-1, SHA-2, SHA-256, SHA-384, and SHA-512
   - Secure Key Accessing
3. RTC and other timers
   - Clock source of 24 MHz & 32.768 kHz Crystal and 24 MHz / 32 kHz embedded RC oscillator
   - Two general 32-bit timers with four selectable modes in active mode
   - Watchdog timer
   - A low-frequency 32 kHz timer available in low power mode
   - A low-frequency 32 kHz Platform-Level Machine Timer (PLMT)
4. A rich set of digital and analog interfaces
   - Up to 48 GPIOs (differs for specific part number, refers to 1.4 Ordering Information)
   - 4 different SPI channels
     - MSPI: Memory SPI[1]
       - Up to 48 MHz SPI clock
       - Supports quad/dual/single data line
       - Supports NOR Flash interface
       - Supports PSRAM interface
       - Supports 4-channel CS_N for Multi-SPI Slave
     - GSPI: Global SPI
       - Up to 48 MHz SPI clock[2]
       - Supports quad/dual/single data line

1. The MSPI interface is used internally and is not available externally; the corresponding pins are not bonded out on the SoC.
2. The GSPI clock reaches 48MHz only when PA[3] and PB[3:0] are configured as GSPI signal.

        - Supports PSRAM interface
        - Supports NOR Flash interface
        - Supports 4-channel CS_N for Multi-SPI Slave
    - LSPI: LCD SPI
        - Up to 48 MHz SPI clock[1]
        - Supports quad/dual/single data line
        - Supports TFT panel interface
        - Supports TFT panel interface without internal RAM
    - SPI Slave (SSPI): SPI Slave for external accessing
- 2x I2C
- 3x I2S
- 3x UART with hardware flow control and 7816 protocol support
- Supports JTAG/SDP/SWS debug interface
    - One JTAG/SDP interface for RISC-V core
    - SWS interface for whole system
- USB 2.0 device, Full Speed
- Up to 7 channels of differential PWM
    - IR transmitter with DMA
    - Deep diming synchronization function
    - Preventing dead zones function
    - Enhanced resolution with dithering function
- IR Learning hardware
    - Hardware accelerator for learning
    - BOM-saving for external components
- One quadrature decoder (QDEC), two-phase input selectable
- Single-channel AMIC (Analog MIC) and Dual-channel DMIC with common audio chain
- 12-bit auxiliary ADC
- Low power comparator
- Supports RZ (Return to Zero)
- Supports PDM interface
- Supports PEM (Peripheral Event Matrix)

5. Supports OTA upgrade and Secure Boot switch, allowing convenient product feature roll out and upgrade
6. Operating temperature range:
    - E version: -40°C ~ +85°C
7. Completely RoHS-compliant package
    - TL7218A, 68-pin QFN 8x8x0.75mm
    - TL7218D, 38-pin QFN 4x4x0.85mm
    - TL7218H, 94-pin BGA 4x6x0.96mm

1. The LSPI clock reaches 48MHz only when PE[5:1] are configured as LSPI signal.

- TL7218J, 56-pin BGA 3.4x3x0.865mm
- TL7215A, 68-pin QFN 8x8x0.75mm
- TL7215D, 38-pin QFN 4x4x0.85mm

1.2.2 CPU and Memory

1. 32-bit RISC-V micro-controller, D25F
   - Instruction and data cache controller (16KB I-Cache and 8KB D-Cache)
   - Maximum running speed up to 240 MHz
   - Integrated DSP extension instructions
     - SIMD Data Processing Instructions
     - Partial-SIMD Data Processing Instructions
     - 64-bit Profile Instructions
     - Non-SIMD Instructions
     - Overflow Status Manipulation Instructions
   - RISC-V RV32I base integer instruction set
   - RISC-V RVC standard extension for compressed instructions
   - RISC-V RVM standard extension for integer multiplication and division
   - RISC-V RVA standard extension for atomic instructions
   - RISC-V "F" standard extensions for single-precision floating-point
2. Memory architecture
   - Program memory: 2 MB (2048KB) or 1 MB (1024KB) embedded flash
   - 512/256 KB SRAM including up to 256 KB retention SRAM
   - Supply PSRAM from 2MB to 16MB
   - 8K bits OTP (One Time Program)
   - ROM for secure boot solution

1.2.3 RF Features

RF features include:

1. Bluetooth/802.15.4/2.4 GHz RF transceiver in worldwide 2.4 GHz ISM band
2. Bluetooth LE 1 Mbps and 2 Mbps, Long Range 125 kbps and 500 kbps, Hyper-length
3. Bluetooth Channel Sounding
4. IEEE 802.15.4 compliant, 250 kbps, and proprietary 2 Mbps High-Rate mode
5. 2.4 GHz proprietary 1 Mbps/2 Mbps/250 kbps/500 kbps mode
6. Supports flexible 2.4G link layer with configurable packet format
7. Rx Sensitivity: -96 dBm @ Bluetooth LE 1 Mbps, -93 dBm @ Bluetooth LE 2 Mbps mode, -103 dBm @ Long Range 125 kbps, -99 dBm @ Long Range 500 kbps; -103 dBm @ 802.15.4
8. TX output power: up to +10dBm @ GFSK modulation
9. 50 Ω matched single-pin antenna input
10. RSSI monitoring with +/-1 dB resolution
11. Auto acknowledgment, retransmission and flow control

12. Supports PTA (Packet Traffic Arbitrator) for Wi-Fi co-existence

1.2.4 Audio Features

Audio features include:

1. Supports audio CODECs such as AEC, LC3, LC3+, LC3+HR, OPUS, SBC etc.
2. Supports audio processing of EQ, ASRC, CVSD
3. Supports AI / NN noise canceling algorithm
4. Supports Environmental Noise Cancellation (ENC)
5. Supports Voice Activity Detection (VAD)
6. Supports Automatic Gain Control (AGC)

1.2.5 Features of Power Management Module

Features of power management module include:

1. Power supply
   - VDD (battery): 1.8 V ~ 4.3 V
   - VBUS (USB): 4.5 V ~ 5.5 V (exclude TL7218A and TL7215A)
2. Embedded LDO and DCDC
   - DCDC for 1.8 V Flash with bypass LDO
   - DCDC for chip with bypass LDO
3. Battery monitor for low battery voltage detection
4. Brownout detection/shutdown and Power-On-Reset
5. Supports power reduction in different Clock Scenarios
6. IO power supply configurable of 1.8 V or 3.3 V
7. Low power consumption:
   - Whole chip, BLE Receive: 1.6 mA @ BLE Rx 4.2 V DCDC mode; 2.0 mA @ BLE Rx 3.3 V DCDC mode, 5.5 mA @BLE Rx LDO mode
   - Whole chip, BLE Transmit: 2.4 mA @ BLE Tx 4.2 V DCDC mode; 2.9 mA @ BLE Tx 0dBm, 3.3 V DCDC mode, 8.8 mA @ BLE Tx 0dBm LDO mode
   - Deep sleep with IO wakeup (without SRAM retention): 1.0 µA
   - Deep sleep with 32K RC oscillator and SRAM retention: 2.0 µA (with 32 KB SRAM retention)

NOTE:

- The power consumption data is based on the test results of the chip for evaluation kit, and they are to be updated after all chips are tested.

1.2.6 Bluetooth Features

Bluetooth LE features include:

1. Qualified Bluetooth LE 6.0, main features include:
   - 1Mbps, 2Mbps, Long Range S2 (500 Kbps), S8 (125 Kbps)
   - High duty cycle non-connectable ADV

- Extended ADV
- LE channel selection algorithm #2

2. Bluetooth SIG Mesh support
3. Bluetooth based location and indoor positioning support with AoA and AoD, up to 64 antennae of the antenna array for data transmission and receiving
4. Bluetooth ISO channel support with broadcast and connected mode

1.2.7 Zigbee Features

Zigbee features include:

1. Supports Zigbee Pro R23, compatible with Zigbee Pro R22 and Zigbee Pro R21, supports Touchlink
2. Supports devices including Coordinator, Router and End Device
3. Supports Zigbee Direct protocol
4. Supports Sink mode and Proxy mode of Green Power

1.2.8 RF4CE Features

RF4CE features include:

1. Compliant to IEEE 802.15.4
2. Utilizes the industry standard AES-128 security scheme
3. Low-power and low-latency control for remote control products
4. Simple and intuitive discovery and pairing mechanism

1.2.9 OpenThread Features

OpenThread features include:

1. Implement all Thread networking layers (IPv6, 6LoWPAN, IEEE 802.15.4 with MAC security, Mesh Link Establishment, Mesh Routing)
2. Supports Full Thread Device (FTD) and Minimal Thread Device (MTD)
3. Supports rich applications
   - IPv6 configuration and raw data interface
   - UDP sockets
   - CoAP client and server

1.2.10 Matter Features

Matter features include:

1. Supports Matter over Thread
2. Supports Multi-Admin works across & with multiple ecosystems

1.2.11 Flash Features

The TL7218A/D/H/J and TL7215A/D embed flash with features below:

1. Total up to 2 MB (2048KB) embedded flash for TL7218A/D/H/J and 1 MB (1024KB) embedded flash for TL7215A/D

2. Flexible architecture: 4 KB per sector, 64 KB / 32 KB per block
3. Up to 256 bytes per programmable page
4. Write protect all or portions of memory
5. Sector erase (4 KB)
6. Block erase (32 KB / 64 KB)
7. Cycle endurance: 100,000 program/erases
8. Data retention: typical 20-year retention

1.2.12 Concurrent Mode Features

In concurrent mode, the chip supports multiple standards working concurrently.

Typical combinations include Bluetooth LE + Zigbee, Bluetooth LE + Thread, Bluetooth LE + 802.15.4, Bluetooth LE + 2.4GHz, etc. In concurrent modes, stacks can run concurrently with one application state but different protocols for interacting with different devices.

1.2.13 Timer Features

1. The SoC has two timers: timer0 and timer1
2. Both timers support four modes:
   - Mode 0 (System Clock Mode)
   - Mode 1 (GPIO Trigger Mode)
   - Mode 2 (GPIO Pulse Width Mode)
   - Mode 3 (Tick Mode)
3. The input capture function and output compare function of timer0 and timer1 are new
4. The Input Capture function can be used with mode 0 and mode 3
5. The Output Compare function can be used with the mode
6. The Input Capture function and the Output Compare function can work at the same time
7. New Input Capture function for System Timer

1.2.14 RZ Features

1. The RZ (Return to Zero) module adopts single wire return to zero code protocol for communication transmission, and can drive serial or parallel pixel ICs
2. RZ Code Timing can be configured flexibly to match different Pixel ICs
3. There are two addressing modes for pixel ICs:
   - Serial sequential addressing
   - Parallel random addressing
4. The RZ module has no CPU intervention during transmission
5. DMA handling data is bounded by 8 bits

1.3 Typical Applications

The TL721x is an ideal SoC for IoT applications. Its typical applications include, but are not limited to the following:

- Keyboard, mouse, dongle for TL7218 series
- Gaming applications for TL7215 series
  - Collar-clip microphone
  - Wireless karaoke set
  - 2.4GHz headset
  - Hearing-aid headphones
- IoT applications
  - Smart home device
  - Smart lighting device
  - Smart RF remote control
  - Wireless HID
  - Wearable device
  - Asset tracking device

1.4 Ordering Information

Table 1-1 Ordering Information of TL721x

Product Series	Ordering No.	SRAM (KB)	Flash (KB)	BLE 6.0	802.15.4	2.4 GHz	USB	GPIO	Package	Temp.	Packing[a]	MOQ[b]
TL7218	TL7218AE 11T68R	512	2048	Y	Y	Y	FS[c]	47	QFN68[d] 8x8x0.75mm	-40°C ~+85°C	T&R	3000
	TL7218DE 11T38R	512	2048	Y	Y	Y	FS	20	QFN38[e] 4x4x0.85mm	-40°C ~+85°C	T&R	3000
	TL7218HE 11B94R	512	2048	Y	Y	Y	FS	48	BGA94[f] 4x6x0.96mm	-40°C ~+85°C	T&R	3000
	TL7218J E11B56R	512	2048	Y	Y	Y	FS	30	BGA56[g] 3.4x3x0.865 mm	-40°C ~+85°C	T&R	3000
TL7215	TL7215AE 10T68R	256	1024	Y	Y	Y	FS	47	QFN68 8x8x0.75mm	-40°C ~+85°C	T&R	3000
	TL7215DE 10T38R	256	1024	Y	Y	Y	FS	20	QFN38 4x4x0.85mm	-40°C ~+85°C	T&R	3000

a. Packing method "T&R" means tape and reel. The tape and reel material DO NOT support baking under high temperature.

b. MOQ stands for Minimum Ordering Quantity.

c. FS stands for Full Speed.

d. QFN68: 10x ADC channel, 2x DMIC, 2x Timer, 2x I2C, 3x I2S, 3x UART, 4x SPI, 7x PWM, 1x JTAG

e. QFN38: 5x ADC channel, 2x DMIC, 2x Timer, 2x I2C, 3x I2S, 3x UART, 4x SPI, 7x PWM

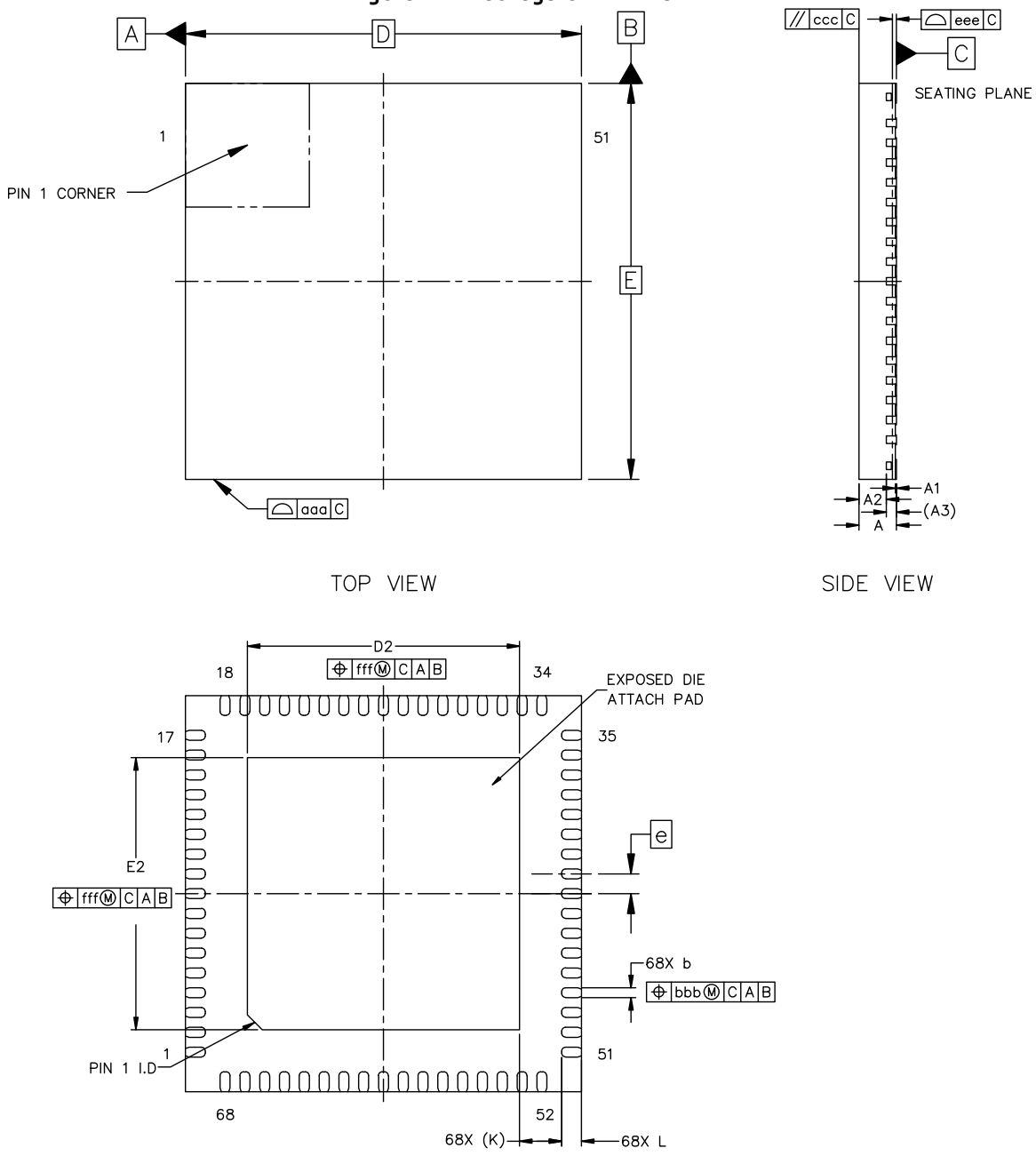
f. BGA94: 10x ADC channel, 2x DMIC, 2x Timer, 2x I2C, 3x I2S, 3x UART, 4x SPI, 7x PWM, 1x JTAG

g. BGA56: 4x ADC channel, 2x DMIC, 2x Timer, 2x I2C, 3x I2S, 3x UART, 4x SPI, 7x PWM, 1x JTAG

1.5 Package

1.5.1 Package Dimensions of TL7218A

Figure 1-2 Package of TL7218A



NOTES

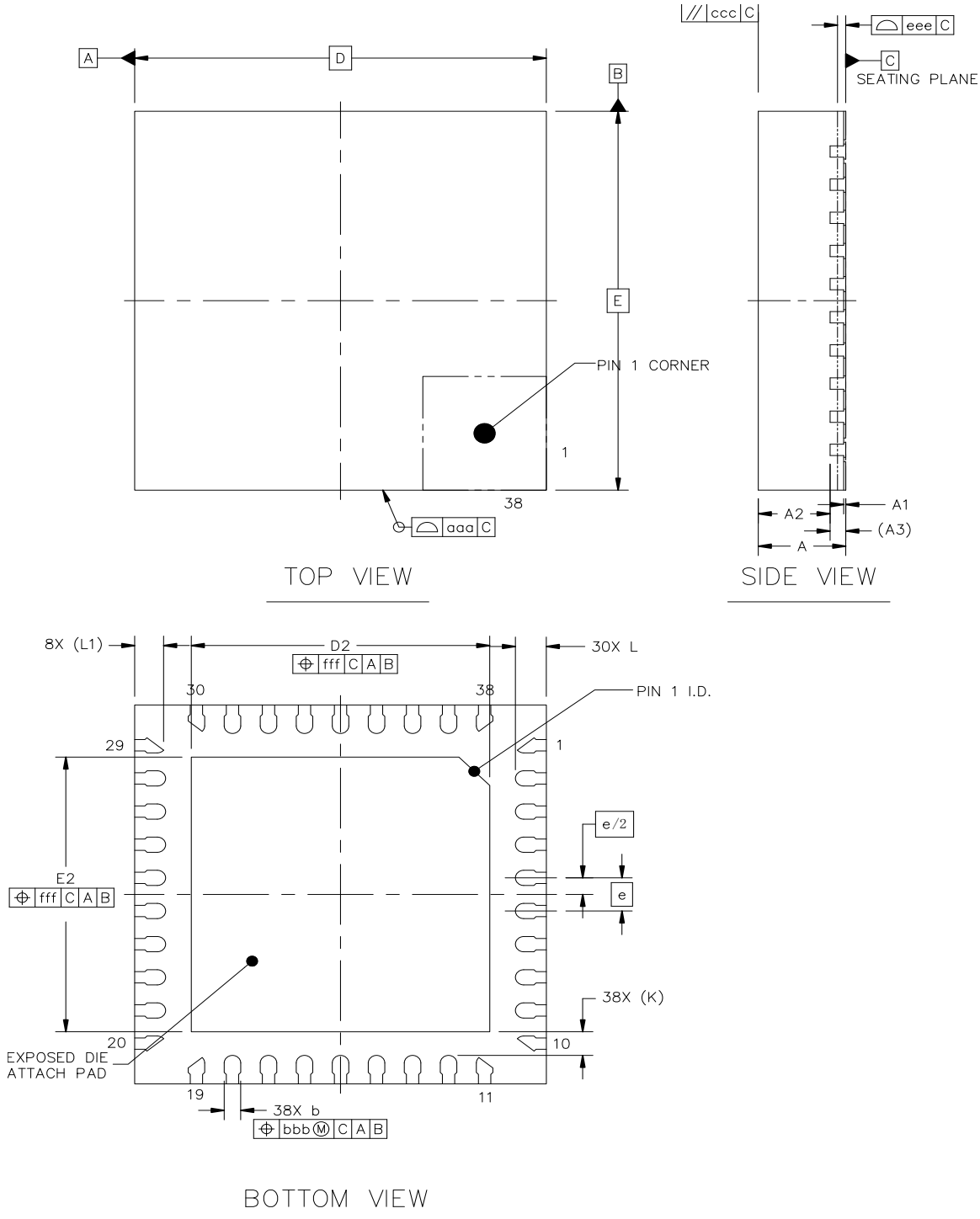
1.REFER TO JEDEC MO-220;
2.COPLANARITY APPLIES TO LEADS, CORNER LEADS AND DIE ATTACH PAD;
3.BAN TO USE THE LEVEL 1 ENVIRONMENT-RELATED SUBSTANCES;
4.FINISH: Cu/EP Sn8~20s

Table 1-2 Mechanical Dimension of TL7218A

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.7	0.75	0.8
A1	0	0.02	0.05
A2	-	0.55	-
A3	0.203 REF		
b	0.15	0.20	0.25
D	8 BSC		
E	8 BSC		
e	0.4 BSC		
D2	5.4	5.5	5.6
E2	5.4	5.5	5.6
L	0.324	0.4	0.476
K	0.85 REF		
aaa	0.1		
ccc	0.1		
eee	0.08		
bbb	0.07		
fff	0.1		

1.5.2 Package Dimensions of TL7218D

Figure 1-3 Package of TL7218D



NOTES

1.REFER TO JEDEC MO-220;
2.COPLANARITY APPLIES TO LEADS, CORNER LEADS AND DIE ATTACH PAD;
3.BAN TO USE THE LEVEL 1 ENVIRONMENT-RELATED SUBSTANCES;
4.FINISH: Cu/EP •Sn8~20s

Table 1-3 Mechanical Dimension of TL7218D

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.8	0.85	0.85
A1	0	0.02	0.05
A2	-	0.7	-
A3	0.152 REF		
b	0.11	0.16	0.21
D	4 BSC		
E	4 BSC		
e	0.35 BSC		
D2	2.8	2.9	3.0
E2	2.8	2.9	3.0
L	0.2	0.3	0.4
L1	0.28 REF		
K	0.25 REF		
aaa	0.1		
ccc	0.1		
eee	0.08		
bbb	0.07		
fff	0.1		

1.5.3 Package Dimensions of TL7218H

Figure 1-4 Package of TL7218H

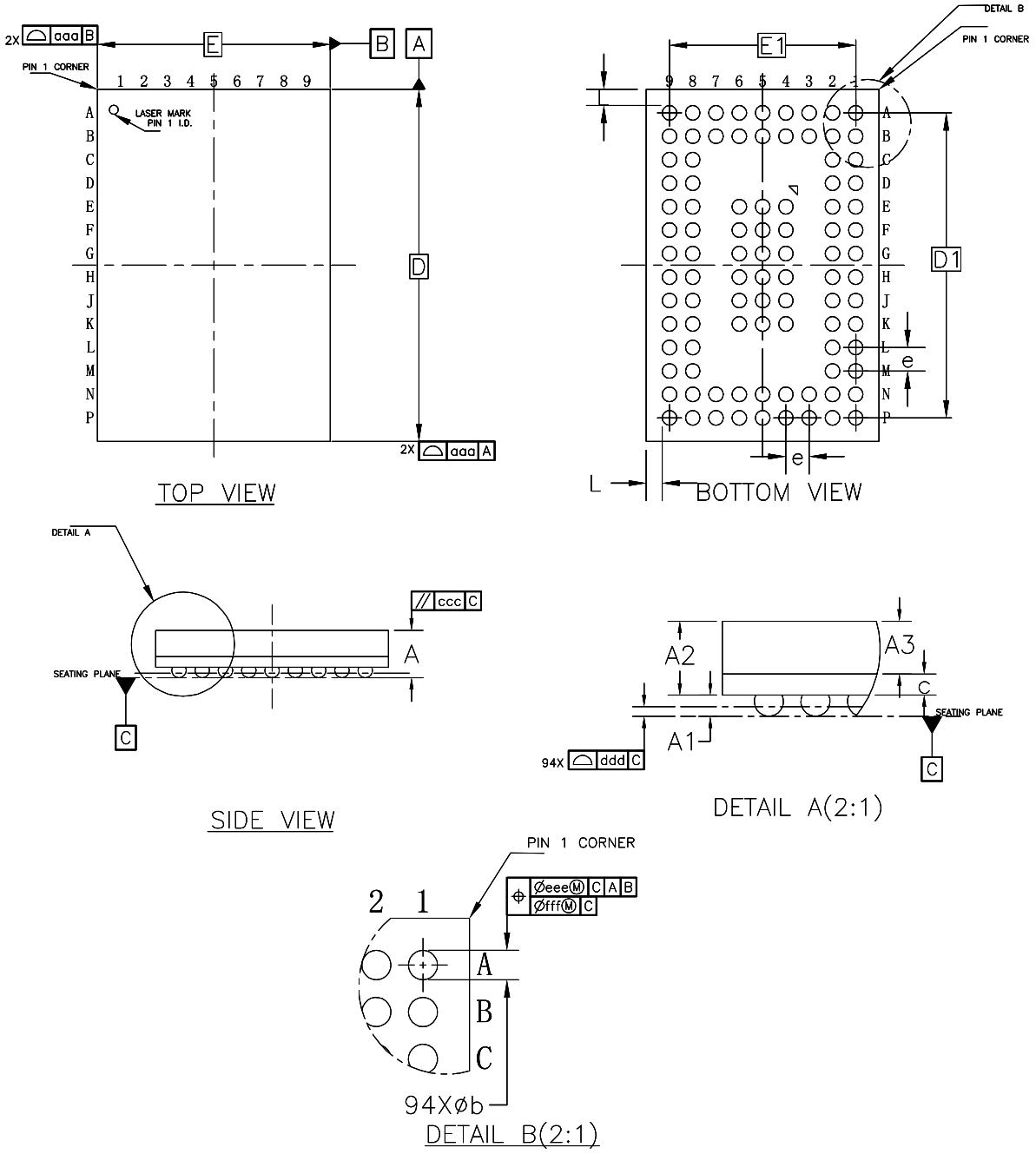
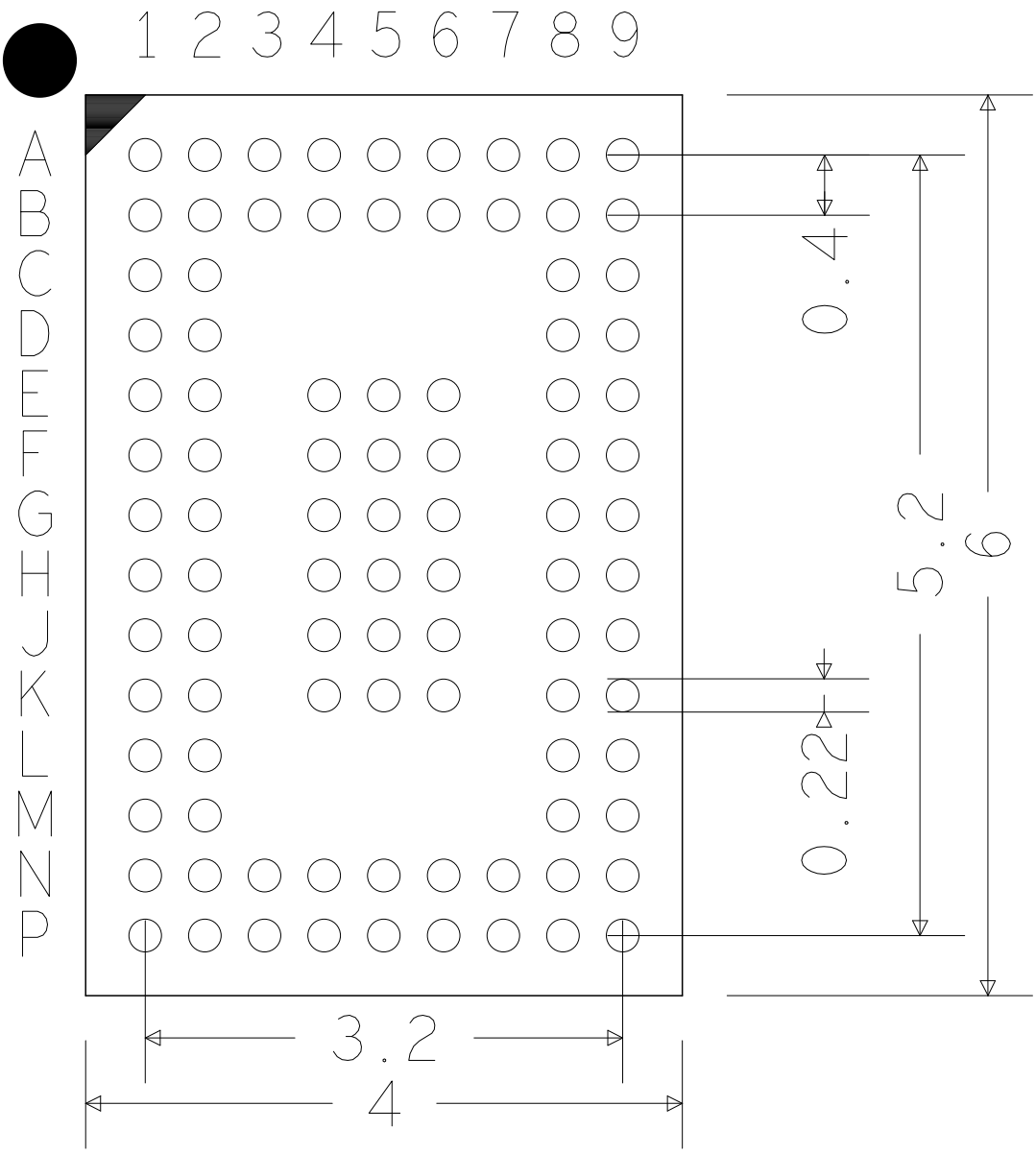


Table 1-4 Mechanical Dimension of TL7218H

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.90	0.96	1.02
A1	0.13	0.18	0.23
A2	0.73	0.78	0.83
A3	0.60 BASIC		
c	0.15	0.18	0.21
D	5.90	6.00	6.10
D1	5.20 BASIC		
E	3.90	4.00	4.10
E1	3.20 BASIC		
e	0.40 BASIC		
b	0.20	0.25	0.30
L	0.275 REF		
aaa	0.10		
ccc	0.08		
ddd	0.08		
eee	0.15		
fff	0.05		

The recommended dimensions of footprint for TL7218H are shown below. The unit is mm.

Figure 1-5 Reference Footprint of TL7218H



1.5.4 Package Dimensions of TL7218J

Figure 1-6 Package of TL7218J

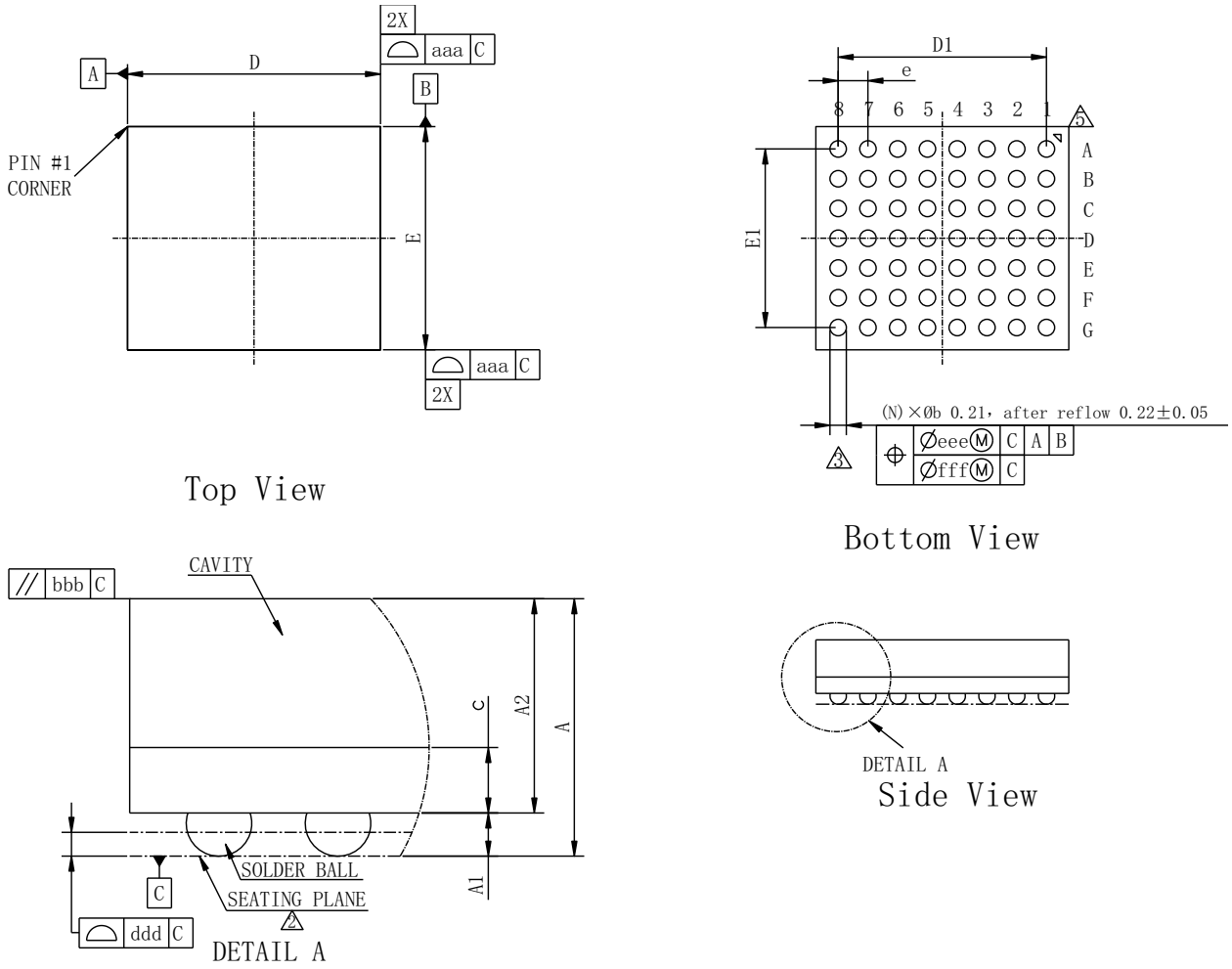


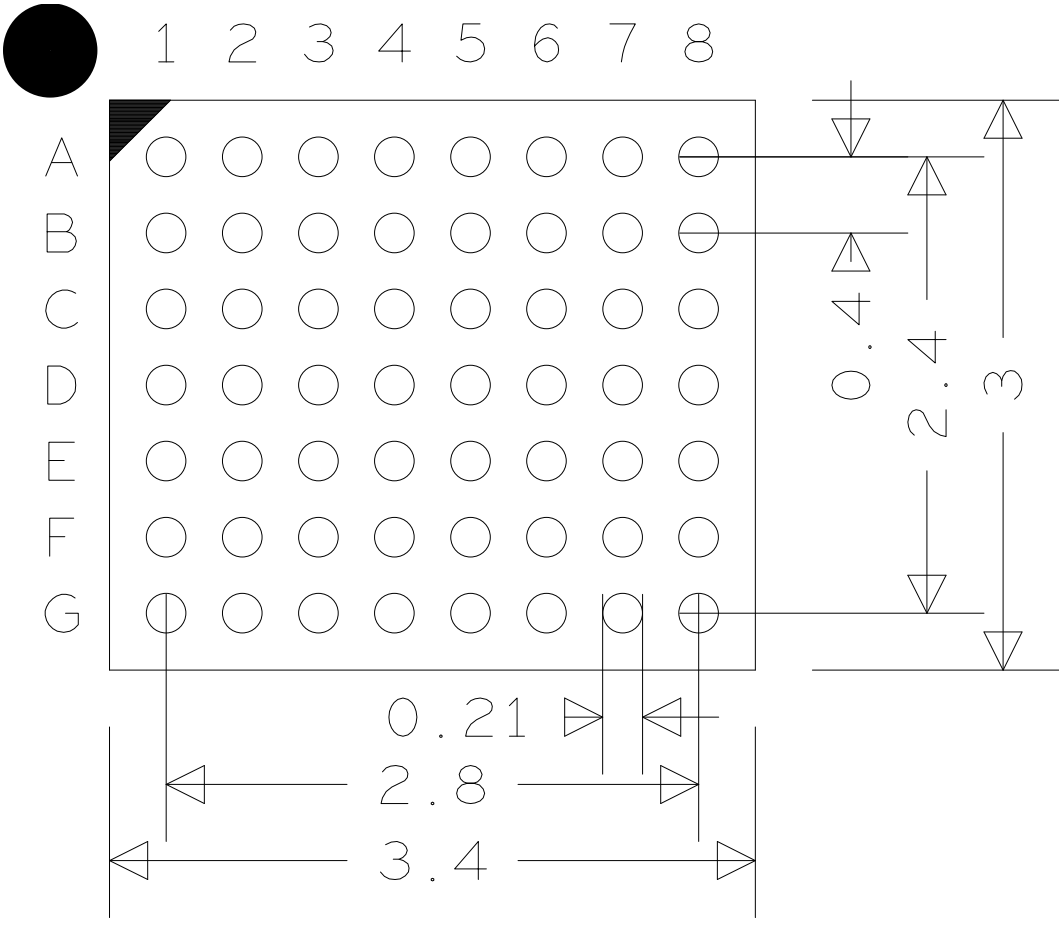
Table 1-5 Mechanical Dimension of TL7218J

Symbol	Millimeter		
	Min	Nom	Max
A	0.765	0.865	0.965
A1	0.095	0.145	0.195
A2	0.660	0.720	0.780
c	0.190	0.220	0.250
D	3.300	3.400	3.500
E	2.900	3.000	3.100
D1	-	2.800	-
E1	-	2.400	-

Symbol	Millimeter		
	Min	Nom	Max
e		0.400	
b	0.170	0.220	0.270
aaa		0.100	
bbb		0.100	
ddd		0.080	
eee		0.150	
fff		0.050	
Ball diameter		0.210	
N		56	
MD/ME		8/7	

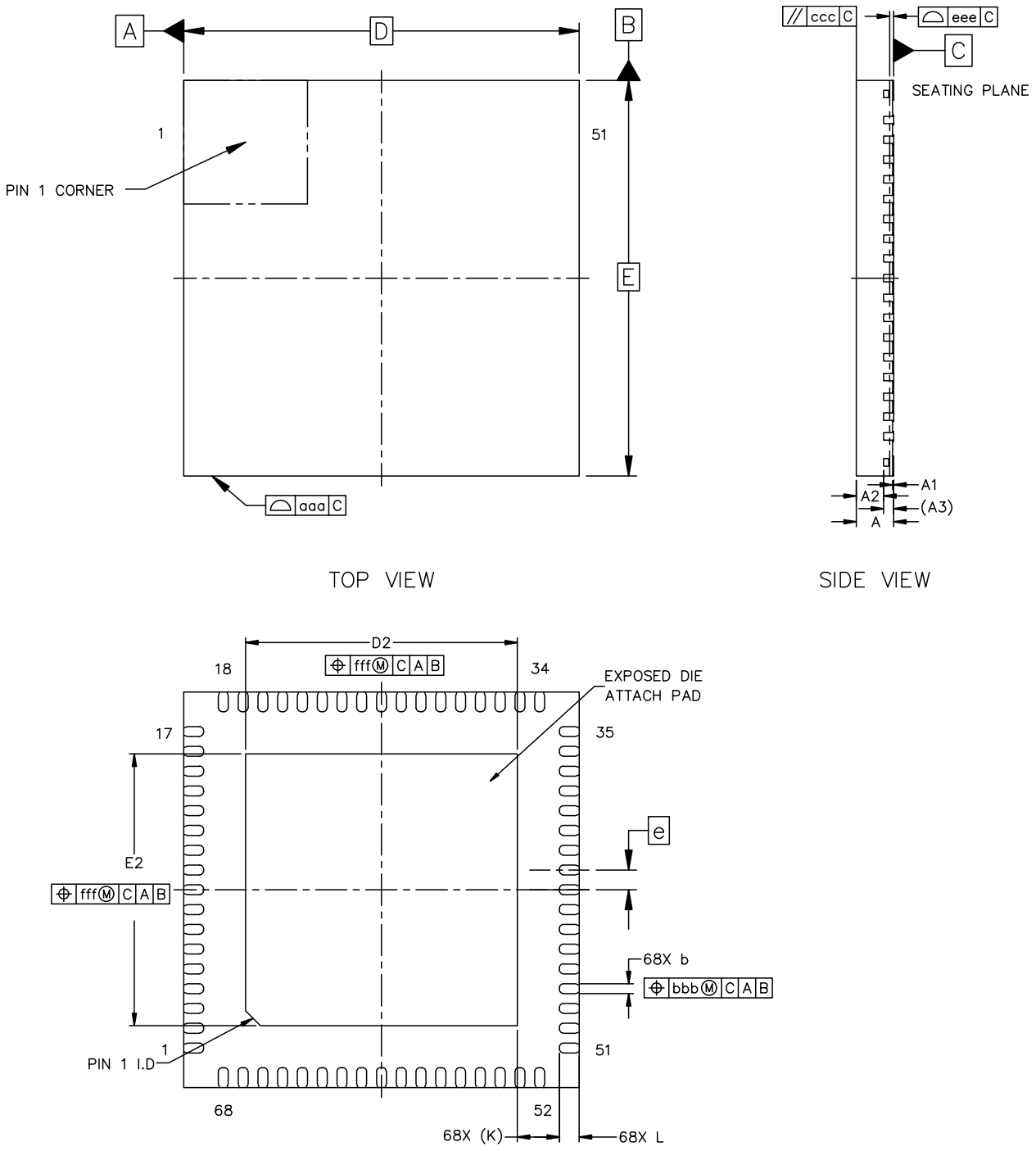
The recommended dimensions of footprint for TL7218J are shown below. The unit is mm.

Figure 1-7 Reference Footprint of TL7218J



1.5.5 Package Dimensions of TL7215A

Figure 1-8 Package of TL7215A



NOTES

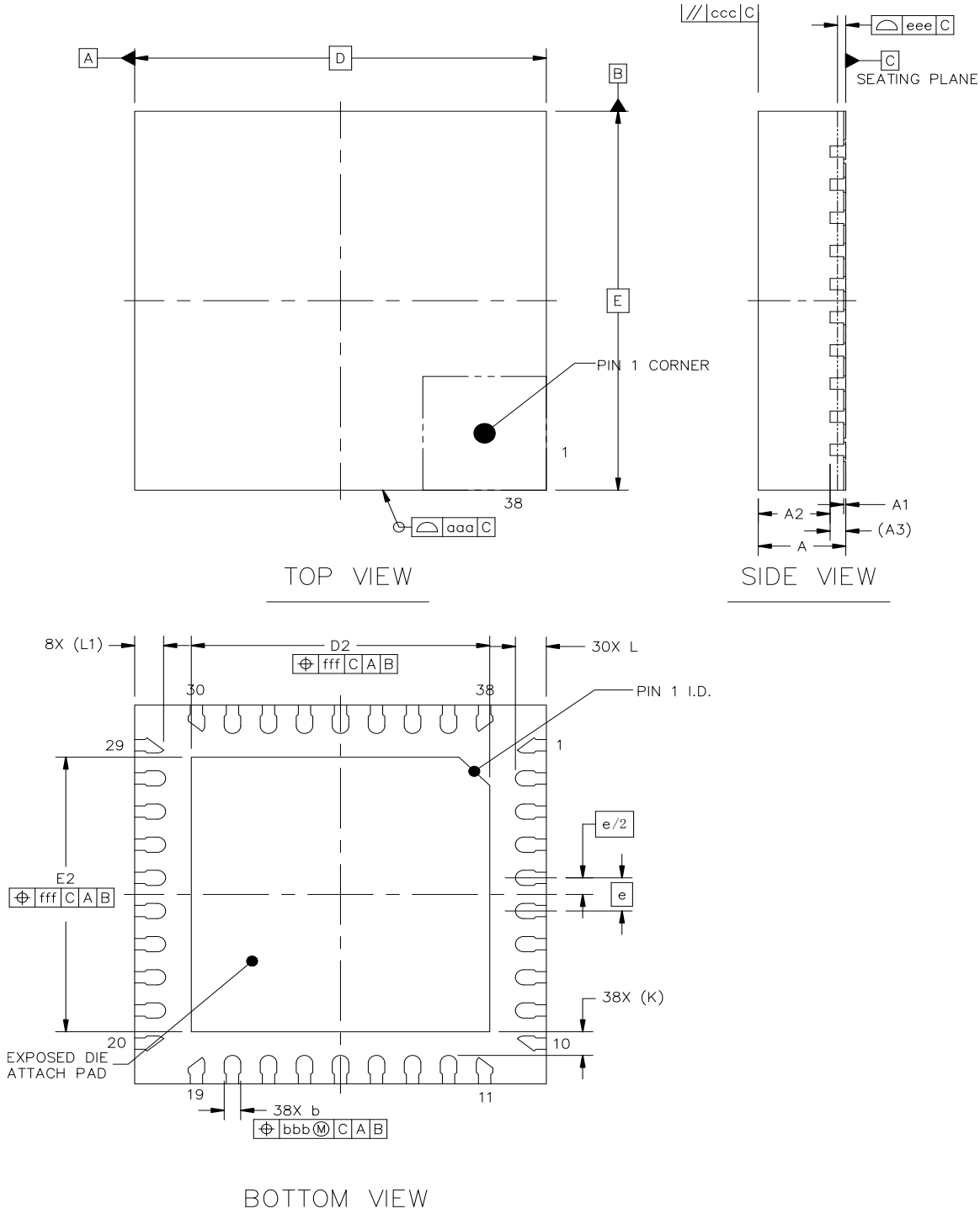
1.REFER TO JEDEC MO-220;
2.COPLANARITY APPLIES TO LEADS, CORNER LEADS AND DIE ATTACH PAD;
3.BAN TO USE THE LEVEL 1 ENVIRONMENT-RELATED SUBSTANCES;
4.FINISH: Cu/EP Sn8~20s

Table 1-6 Mechanical Dimension of TL7215A

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.7	0.75	0.8
A1	0	0.02	0.05
A2	-	0.55	-
A3	0.203 REF		
b	0.15	0.20	0.25
D	8 BSC		
E	8 BSC		
e	0.4 BSC		
D2	5.4	5.5	5.6
E2	5.4	5.5	5.6
L	0.324	0.4	0.476
K	0.85 REF		
aaa	0.1		
ccc	0.1		
eee	0.08		
bbb	0.07		
fff	0.1		

1.5.6 Package Dimensions of TL7215D

Figure 1-9 Package of TL7215D



NOTES

1.REFER TO JEDEC MO-220;
2.COPLANARITY APPLIES TO LEADS, CORNER LEADS AND DIE ATTACH PAD;
3.BAN TO USE THE LEVEL 1 ENVIRONMENT-RELATED SUBSTANCES;
4.FINISH: Cu/EP •Sn8~20s

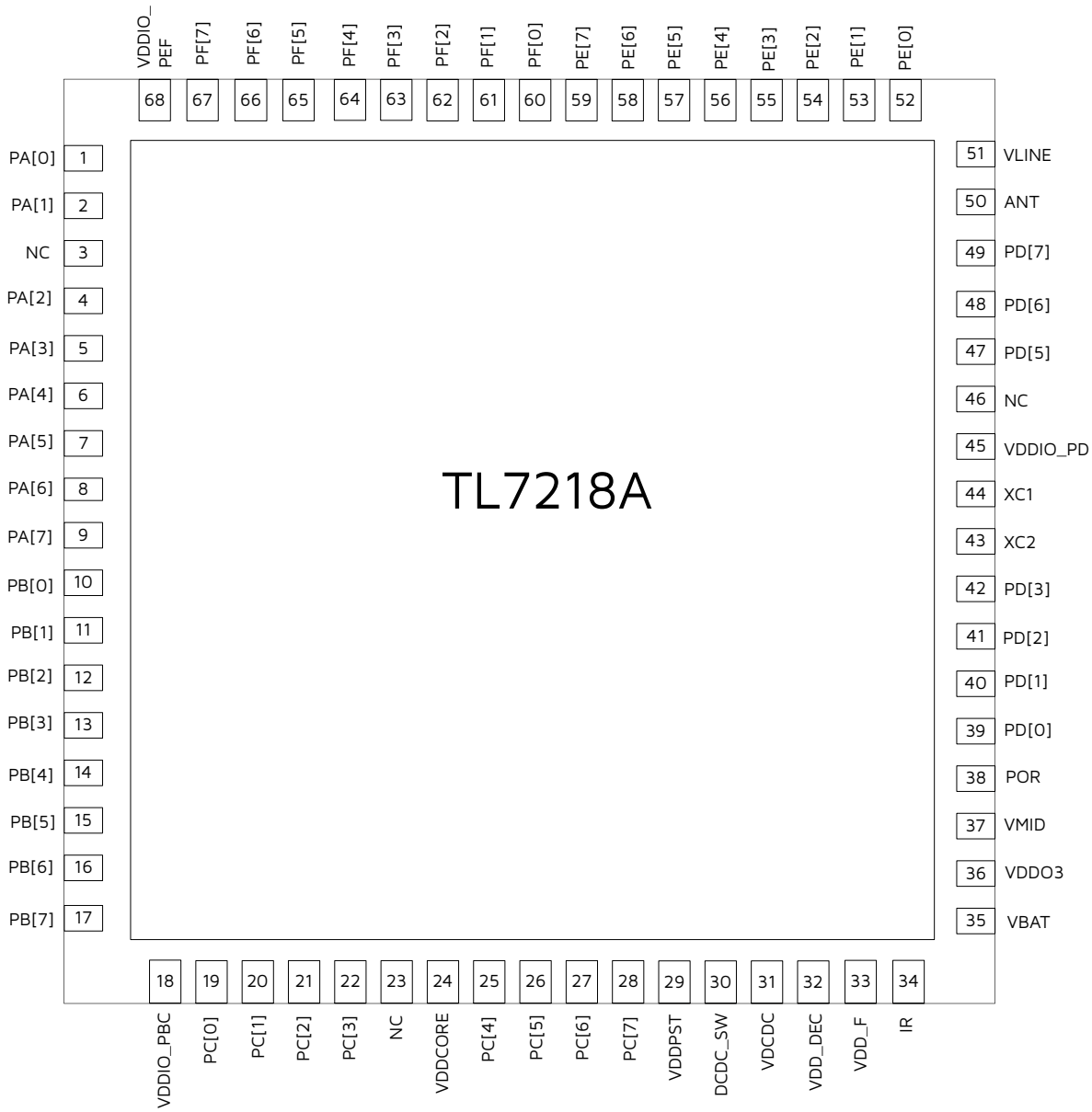
Table 1-7 Mechanical Dimension of TL7215D

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.8	0.85	0.85
A1	0	0.02	0.05
A2	-	0.7	-
A3	0.152 REF		
b	0.11	0.16	0.21
D	4 BSC		
E	4 BSC		
e	0.35 BSC		
D2	2.8	2.9	3.0
E2	2.8	2.9	3.0
L	0.2	0.3	0.4
L1	0.28 REF		
K	0.25 REF		
aaa	0.1		
ccc	0.1		
eee	0.08		
bbb	0.07		
fff	0.1		

1.6 Pin Layout

1.6.1 Pin Assignment of TL7218A

Figure 1-10 Pin Assignment of TL7218A



Functions of 68 pins of TL7218A are described in table below.

Table 1-8 Pin Function of TL7218A

No.	Pin Name	Type	Description
1	PA[0]	GPIO	GPIO PA[0]
2	PA[1]	GPIO	GPIO PA[1]
3	NC	-	Not connected

No.	Pin Name	Type	Description
4	PA[2]	GPIO	GPIO PA[2]
5	PA[3]	GPIO	GPIO PA[3]
6	PA[4]	GPIO	GPIO PA[4]
7	PA[5]	GPIO	GPIO PA[5]
8	PA[6]	GPIO	GPIO PA[6]
9	PA[7]	GPIO	GPIO PA[7]
10	PB[0]	GPIO	GPIO PB[0]
11	PB[1]	GPIO	GPIO PB[1]
12	PB[2]	GPIO	GPIO PB[2]
13	PB[3]	GPIO	GPIO PB[3]
14	PB[4]	GPIO	GPIO PB[4]
15	PB[5]	GPIO	GPIO PB[5]
16	PB[6]	GPIO	GPIO PB[6]
17	PB[7]	GPIO	GPIO PB[7]
18	VDDIO_PBC	PWR	IO voltage for PB0 ~ PB7, PC0 ~ PC7 and PA7, configurable to 3.3V or 1.8V
19	PC[0]	GPIO	GPIO PC[0]
20	PC[1]	GPIO	GPIO PC[1]
21	PC[2]	GPIO	GPIO PC[2]
22	PC[3]	GPIO	GPIO PC[3]
23	NC	-	Not connected
24	VDDCORE	PWR	Digital core power supply
25	PC[4]	GPIO	GPIO PC[4]
26	PC[5]	GPIO	GPIO PC[5]
27	PC[6]	GPIO	GPIO PC[6]
28	PC[7]	GPIO	GPIO PC[7]
29	VDDPST	PWR	3.3V input, connected with external capacitor
30	DCDC_SW	Analog	Connected with VDCDC via external inductor
31	VDCDC	Analog	Connected with DCDC_SW via external inductor

No.	Pin Name	Type	Description
32	VDD_DEC	PWR	0.94V digital power supply
33	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor
34	IR	Analog	Infrared radiation learning
35	VBAT	PWR	Lion-Battery power supply
36	VDDO3	PWR	3.3V LDO output
37	VMID	Analog	Audio reference voltage, 0V by default, 0.9V when audio is enabled
38	POR	Analog	Power on reset
39	PD[0]	GPIO	GPIO PD[0]
40	PD[1]	GPIO	GPIO PD[1]
41	PD[2]	GPIO	GPIO PD[2]
42	PD[3]	GPIO	GPIO PD[3]
43	XC2	Analog	Crystal oscillator pin 2
44	XC1	Analog	Crystal oscillator pin 1
45	VDDIO_PD	PWR	IO voltage for PD0 ~ PD7, configurable to 3.3V or 1.8V
46	NC	-	Not connected
47	PD[5]	GPIO	GPIO PD[5]
48	PD[6]	GPIO	GPIO PD[6]
49	PD[7]	GPIO	GPIO PD[7]
50	ANT	Analog	Pin to connect to the antenna through the matching network
51	VLINE	PWR	0.94V power supply of RF Transceiver
52	PE[0]	GPIO	GPIO PE[0]
53	PE[1]	GPIO	GPIO PE[1]
54	PE[2]	GPIO	GPIO PE[2]
55	PE[3]	GPIO	GPIO PE[3]
56	PE[4]	GPIO	GPIO PE[4]
57	PE[5]	GPIO	GPIO PE[5]
58	PE[6]	GPIO	GPIO PE[6]
59	PE[7]	GPIO	GPIO PE[7]

No.	Pin Name	Type	Description
60	PF[0]	GPIO	GPIO PF[0]
61	PF[1]	GPIO	GPIO PF[1]
62	PF[2]	GPIO	GPIO PF[2]
63	PF[3]	GPIO	GPIO PF[3]
64	PF[4]	GPIO	GPIO PF[4]
65	PF[5]	GPIO	GPIO PF[5]
66	PF[6]	GPIO	GPIO PF[6]
67	PF[7]	GPIO	GPIO PF[7]
68	VDDIO_PEF	PWR	IO voltage for PE0 ~ PE7, PF0 ~ PF7, PA0 ~ PA4, configurable to 3.3V or 1.8V

GPIO pin mux functions of TL7218A are shown in the table below.

Table 1-9 GPIO Pin Mux of TL7218A

Pad	Default	Function1	Function2	Analog Function
PA[0]	GPIO	All functions[a]	SWM	-
PA[1]	GPIO	All functions	SWM	-
PA[2]	GPIO	All functions	SWM	-
PA[3]	GPIO	All functions	SWM	-
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[0]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[1]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[2]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[3]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[4]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[5]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[6]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[7]	GPIO	All functions	SWM	sar_in/lc_cmp_in

Pad	Default	Function1	Function2	Analog Function
PC[0]	SSPI_CN	GPIO, All functions	SSPI_CN	-
PC[1]	SSPI_CK	GPIO, All functions	SSPI_CK	-
PC[2]	SSPI_SI	GPIO, All functions	SSPI_SI	-
PC[3]	SSPI_SO	GPIO, All functions	SSPI_SO	-
PC[4]	TDI	GPIO, All functions	TDI	-
PC[5]	TDO	GPIO, All functions	TDO	-
PC[6]	TMS	GPIO, All functions	TMS	-
PC[7]	TCK	GPIO, All functions	TCK	-
PD[0]	GPIO	All functions	SWM	Audio in/sar in
PD[1]	GPIO	All functions	SWM	led_strip_in1/sar_in
PD[2]	GPIO	All functions	SWM	32K xc2/led_strip_in2
PD[3]	GPIO	All functions	SWM	32K xc1/diag_hv_ana
PD[5][b]	GPIO	All functions	SWM	atb1
PD[6][c]	GPIO	All functions	SWM	-
PD[7]	GPIO	All functions	SWM	-
PE[0]	GPIO	All functions	SWM	-
PE[1][d]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PE[6]	GPIO	All functions	SWM	-
PE[7]	GPIO	All functions	SWM	-
PF[0]	GPIO	All functions	SWM	-
PF[1]	GPIO	All functions	SWM	-
PF[2]	GPIO	All functions	SWM	-
PF[3]	GPIO	All functions	SWM	-
PF[4]	GPIO	All functions	SWM	-

Pad	Default	Function1	Function2	Analog Function
PF[5]	GPIO	All functions	SWM	-
PF[6]	GPIO	All functions	SWM	-
PF[7]	GPIO	All functions	SWM	-

a. "All functions" include 89 functions: SWM, RZ_TX, MSPI_CN1, LSPI_CN, TMR1_CMP, TMR0_CMP, PWM6_N, PWM6, PWM_SYNC, SSPI_SO, SSPI_SI, SSPI_CK, SSPI_CN, I2S2_CLK, I2S2_DAT1, I2S2_LR1, I2S2_DAT0, I2S2_LR0, I2S2_BCK, SDM1_N, SDM1_P, SDM0_N, SDM0_P, UART2_RTX, UART2_TX, UART2_RTS, UART2_CTS, IR_LEARN, ATSEL_5, ATSEL_4, GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, MSPI_CN3, MSPI_CN2, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT1, I2S1_LR1, I2S1_DAT0, I2S1_LR0, I2S1_BCK, I2S0_CLK, I2S0_DAT1, I2S0_LR1, I2S0_DAT0, I2S0_LR0, I2S0_BCK, CLK_7816, UART1_RTX, UART1_TX, UART1_RTS, UART1_CTS, UART0_RTX, UART0_TX, UART0_RTS, UART0_CTS, I2C_SDA, I2C_SCL, GSPI_MOSI, GSPI_MISO, GSPI_IO2, GSPI_IO3, GSPI_CK, GSPI_CN0, PWM5_N, PWM4_N, PWM3_N, PWM2_N, PWM1_N, PWM0_N, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0.

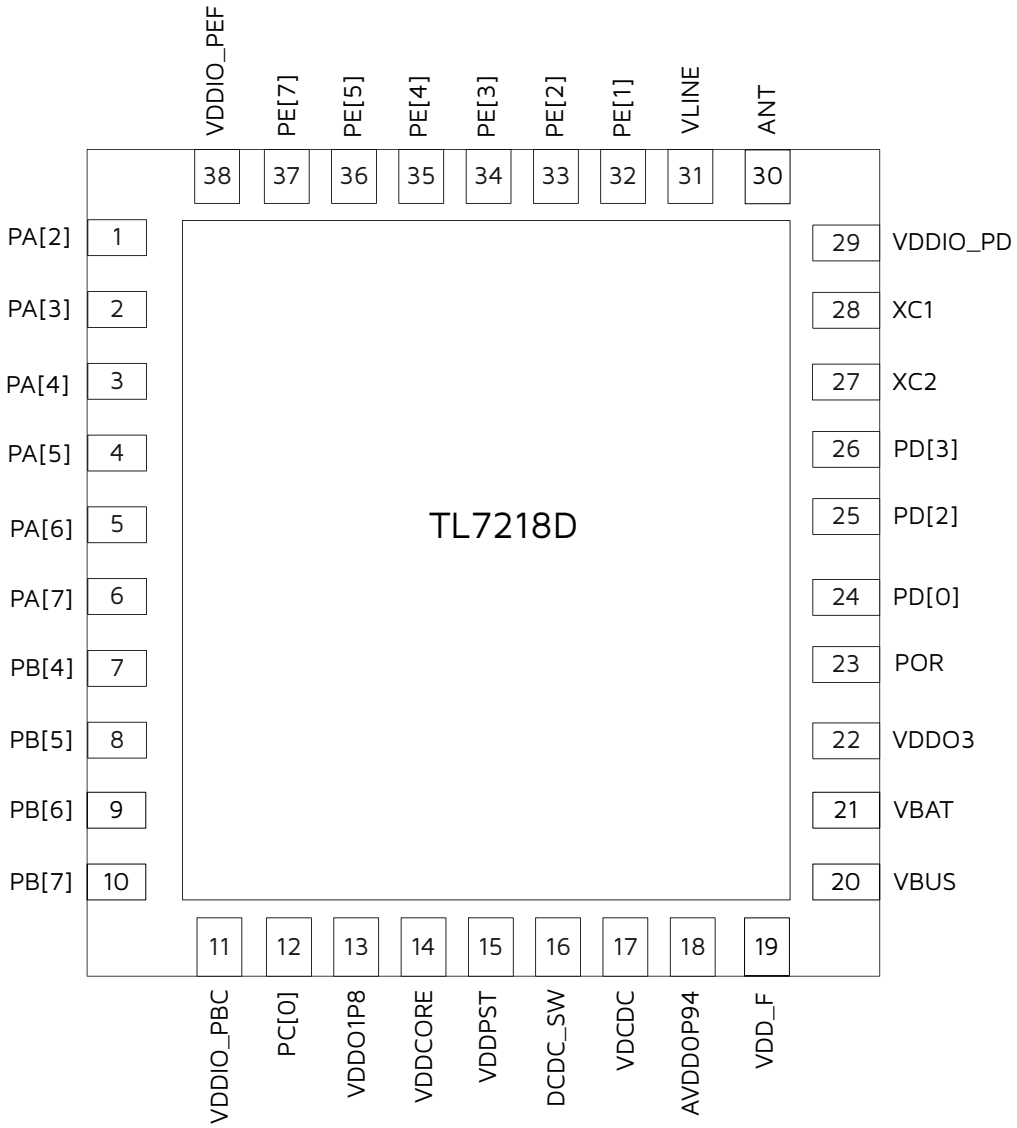
b. (1) PD[5], PD[6], PD[7] of TL7218A are not used for Channel Sounding application. (2) PD[5] is recommended to be used as output only, the details refer to the hardware design guideline.

c. PD[6] and PD[7] are not recommended to be used as PWM output.

d. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

1.6.2 Pin Assignment of TL7218D

Figure 1-11 Pin Assignment of TL7218D



Functions of 38 pins of TL7218D are described in table below.

Table 1-10 Pin Function of TL7218D

NO.	Pin Name	Type	Description
1	PA[2]	GPIO	GPIO PA[2]
2	PA[3]	GPIO	GPIO PA[3]
3	PA[4]	GPIO	GPIO PA[4]
4	PA[5]	GPIO	GPIO PA[5]
5	PA[6]	GPIO	GPIO PA[6]
6	PA[7]	GPIO	GPIO PA[7]

NO.	Pin Name	Type	Description
7	PB[4]	GPIO	GPIO PB[4]
8	PB[5]	GPIO	GPIO PB[5]
9	PB[6]	GPIO	GPIO PB[6]
10	PB[7]	GPIO	GPIO PB[7]
11	VDDIO_PBC	PWR	IO voltage for PB4 ~ PB7, PC0 and PA7, configurable to 3.3V or 1.8V
12	PC[0]	GPIO	GPIO PC[0]
13	VDDO1P8	PWR	1.8V LDO output
14	VDDCORE	PWR	Digital core power supply
15	VDDPST	PWR	3.3V input, connected with external capacitor
16	DCDC_SW	Analog	Connected with VDCDC via external inductor
17	VDCDC	Analog	Connected with DCDC_SW via external inductor
18	AVDD0P94	PWR	0.94V analog power supply
19	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor
20	VBUS	PWR	5V USB power supply
21	VBAT	PWR	Lion-Battery power supply
22	VDDO3	PWR	3.3V LDO output
23	POR	Analog	Power on reset
24	PD[0]	GPIO	GPIO PD[0]
25	PD[2]	GPIO	GPIO PD[2]
26	PD[3]	GPIO	GPIO PD[3]
27	XC2	Analog	Crystal oscillator pin 2
28	XC1	Analog	Crystal oscillator pin 1
29	VDDIO_PD	PWR	IO voltage for PD0, PD2 and PD3, configurable to 3.3V or 1.8V
30	ANT	Analog	Pin to connect to the Antenna through the matching network
31	VLINE	PWR	0.94V power supply of RF Transceiver
32	PE[1]	GPIO	GPIO PE[1]
33	PE[2]	GPIO	GPIO PE[2]

NO.	Pin Name	Type	Description
34	PE[3]	GPIO	GPIO PE[3]
35	PE[4]	GPIO	GPIO PE[4]
36	PE[5]	GPIO	GPIO PE[5]
37	PE[7]	GPIO	GPIO PE[7]
38	VDDIO_PEF	PWR	IO voltage for PE1 ~ PE5, PE7, PA2 ~ PA4, configurable to 3.3V or 1.8V

GPIO pin mux functions of TL7218D are shown in the table below.

Table 1-11 GPIO Pin Mux of TL7218D

Pad	Default	Function1	Function2	Analog Function
PA[2]	GPIO	All functions[a]	SWM	-
PA[3]	GPIO	All functions	SWM	-
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[4]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[5]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[6]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[7]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PC[0]	SSPI_CN	GPIO, All functions	SSPI_CN	-
PD[0]	GPIO	All functions	SWM	Audio in/sar in
PD[2]	GPIO	All functions	SWM	32K xc2/led_strip_in2
PD[3]	GPIO	All functions	SWM	32K xc1/diag_hv_ana
PE[1][b]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PE[7]	GPIO	All functions	SWM	-

a. "All functions" include 89 functions: SWM, RZ_TX, MSPI_CN1, LSPI_CN, TMR1_CMP, TMR0_CMP, PWM6_N, PWM6, PWM_SYNC, SSPI_SO, SSPI_SI, SSPI_CK, SSPI_CN, I2S2_CLK, I2S2_DAT1, I2S2_LR1, I2S2_DAT0, I2S2_LR0, I2S2_BCK, SDM1_N, SDM1_P, SDM0_N, SDM0_P, UART2_RTX, UART2_TX, UART2_RTS, UART2_CTS, IR_LEARN, ATSEL_5, ATSEL_4, GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, MSPI_CN3, MSPI_CN2, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT1, I2S1_LR1, I2S1_DAT0, I2S1_LR0, I2S1_BCK, I2S0_CLK, I2S0_DAT1, I2S0_LR1, I2S0_DAT0, I2S0_LR0, I2S0_BCK, CLK_7816, UART1_RTX, UART1_TX, UART1_RTS, UART1_CTS, UART0_RTX, UART0_TX, UART0_RTS, UART0_CTS, I2C_SDA, I2C_SCL, GSPI_MOSI, GSPI_MISO, GSPI_IO2, GSPI_IO3, GSPI_CK, GSPI_CN0, PWM5_N, PWM4_N, PWM3_N, PWM2_N, PWM1_N, PWM0_N, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0.

b. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

1.6.3 Pin Assignment of TL7218H

Figure 1-12 Pin Assignment of TL7218H

	1	2	3	4	5	6	7	8	9
A	PF[4]	PF[0]	PE[6]	PE[5]	PE[0]	PE[2]	VLINE	GLINE	ANT
B	PF[5]	PF[1]	PE[7]	PE[4]	PE[1]	PE[3]	POR	PD[7]	GANT
C	PF[6]	PF[2]						PD[6]	VDDIO_PD
D	PF[7]	PF[3]						PD[5]	XC1
E	VDDIO_PEF	PA[2]		NC	VSS	VSS		PD[4]	XC2
F	PA[1]	PA[3]		NC	VSS	VSS		PD[3]	VMID
G	PA[0]	PA[4]		NC	VSS	VSS		PD[2]	PD[1]
H	PA[7]	PB[2]		NC	VSS	VSS		VDDIO_AMS	PD[0]
J	PA[6]	PB[3]		NC	DVSS	AVSS		VDDO3	VBAT
K	PA[5]	PB[4]		NC	DVSS	AVSS		VBUS	VBUS
L	PB[5]	PB[6]						VDDPST	VDCDC
M	PB[0]	PB[7]						VDD_F	DCDC_SW
N	VDDIO_PB	PB[1]	DVSS	PC[0]	PC[1]	PC[4]	PC[5]	IR_EX_DIOD	DVDD
P	VDDIO_PC	VDDO1P8	VDDCORE	PC[2]	PC[3]	PC[6]	PC[7]	IR	AVDD

Functions of 94 pins of TL7218H are described in table below.

Table 1-12 Pin Function of TL7218H

NO.	Pin Name	Type	Description
A1	PF[4]	GPIO	GPIO PF[4]
A2	PF[0]	GPIO	GPIO PF[0]
A3	PE[6]	GPIO	GPIO PE[6]
A4	PE[5]	GPIO	GPIO PE[5]
A5	PE[0]	GPIO	GPIO PE[0]
A6	PE[2]	GPIO	GPIO PE[2]
A7	VLINE	PWR	0.94V power supply of RF Transceiver
A8	GLINE	GND	Ground of RF transceiver
A9	ANT	Analog	Pin to connect to the Antenna through the matching network
B1	PF[5]	GPIO	GPIO PF[5]
B2	PF[1]	GPIO	GPIO PF[1]
B3	PE[7]	GPIO	GPIO PE[7]
B4	PE[4]	GPIO	GPIO PE[4]
B5	PE[1]	GPIO	GPIO PE[1]
B6	PE[3]	GPIO	GPIO PE[3]
B7	POR	Analog	Power on reset
B8	PD[7]	GPIO	GPIO PD[7]
B9	GANT	GND	Ground of antenna
C1	PF[6]	GPIO	GPIO PF[6]
C2	PF[2]	GPIO	GPIO PF[2]
C8	PD[6]	GPIO	GPIO PD[6]
C9	VDDIO_PD	PWR	IO voltage for PD0 ~ PD7, configurable to 3.3V or 1.8V
D1	PF[7]	GPIO	GPIO PF[7]
D2	PF[3]	GPIO	GPIO PF[3]
D8	PD[5]	GPIO	GPIO PD[5]
D9	XC1	Analog	Crystal oscillator pin 1
E1	VDDIO_PEF	PWR	IO voltage for PE0 ~ PE7, PF0 ~ PF7, PA0 ~ PA4, configurable to 3.3V or 1.8V

NO.	Pin Name	Type	Description
E2	PA[2]	GPIO	GPIO PA[2]
E4	NC	NC	No connection
E5	VSS	GND	GND
E6	VSS	GND	GND
E8	PD[4]	GPIO	GPIO PD[4]
E9	XC2	Analog	Crystal oscillator pin 2
F1	PA[1]	GPIO	GPIO PA[1]
F2	PA[3]	GPIO	GPIO PA[3]
F4	NC	NC	No connection
F5	VSS	GND	GND
F6	VSS	GND	GND
F8	PD[3]	GPIO	GPIO PD[3]
F9	VMID	Analog	Audio reference voltage, 0V by default, 0.9V when audio is enabled
G1	PA[0]	GPIO	GPIO PA[0]
G2	PA[4]	GPIO	GPIO PA[4]
G4	NC	NC	No connection
G5	VSS	GND	GND
G6	VSS	GND	GND
G8	PD[2]	GPIO	GPIO PD[2]
G9	PD[1]	GPIO	GPIO PD[1]
H1	PA[7]	GPIO	GPIO PA[7]
H2	PB[2]	GPIO	GPIO PB[2]
H4	NC	NC	No connection
H5	VSS	GND	GND
H6	VSS	GND	GND
H8	VDDIO_AMS	PWR	IO voltage for AMS
H9	PD[0]	GPIO	GPIO PD[0]
J1	PA[6]	GPIO	GPIO PA[6]

NO.	Pin Name	Type	Description
J2	PB[3]	GPIO	GPIO PB[3]
J4	NC	NC	No connection
J5	DVSS	GND	Digital core ground
J6	AVSS	GND	Analog core ground
J8	VDDO3	PWR	3.3V LDO output
J9	VBAT	PWR	Lion-Battery power supply
K1	PA[5]	GPIO	GPIO PA[5]
K2	PB[4]	GPIO	GPIO PB[4]
K4	NC	NC	No connection
K5	DVSS	GND	Digital core ground
K6	AVSS	GND	Analog core ground
K8	VBUS	PWR	5V USB power supply
K9	VBUS	PWR	5V USB power supply
L1	PB[5]	GPIO	GPIO PB[5]
L2	PB[6]	GPIO	GPIO PB[6]
L8	VDDPST	PWR	3.3V input, connected with external capacitor
L9	VDCDC	Analog	Connected with DCDC_SW via external inductor
M1	PB[0]	GPIO	GPIO PB[0]
M2	PB[7]	GPIO	GPIO PB[7]
M8	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor
M9	DCDC_SW	Analog	Connected with VDCDC via external inductor
N1	VDDIO_PB	PWR	IO voltage for PB0 ~ PB7 and PA7, configurable to 3.3V or 1.8V
N2	PB[1]	GPIO	GPIO PB[1]
N3	DVSS	GND	Digital core ground
N4	PC[0]	GPIO	GPIO PC[0]
N5	PC[1]	GPIO	GPIO PC[1]
N6	PC[4]	GPIO	GPIO PC[4]

NO.	Pin Name	Type	Description
N7	PC[5]	GPIO	GPIO PC[5]
N8	IR_EX_DIOD	Analog	Infrared radiation learning
N9	DVDD	PWR	Digital power supply
P1	VDDIO_PC	PWR	IO voltage for PC0 ~ PC7, configurable to 3.3V or 1.8V
P2	VDDO1P8	PWR	1.8V LDO output
P3	VDDCORE	PWR	Digital core power supply
P4	PC[2]	GPIO	GPIO PC[2]
P5	PC[3]	GPIO	GPIO PC[3]
P6	PC[6]	GPIO	GPIO PC[6]
P7	PC[7]	GPIO	GPIO PC[7]
P8	IR	Analog	Infrared radiation learning
P9	AVDD	PWR	Analog power supply

GPIO pin mux functions of TL7218H are shown in the table below.

Table 1-13 GPIO Pin Mux of TL7218H

Pad	Default	Function1	Function2	Analog Function
PA[0]	GPIO	All functions[a]	SWM	-
PA[1]	GPIO	All functions	SWM	-
PA[2]	GPIO	All functions	SWM	-
PA[3]	GPIO	All functions	SWM	-
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[0]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[1]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[2]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[3]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[4]	GPIO	All functions	SWM	sar_in/lc_cmp_in

Pad	Default	Function1	Function2	Analog Function
PB[5]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[6]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[7]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PC[0]	SSPI_CN	GPIO, All functions	SSPI_CN	-
PC[1]	SSPI_CK	GPIO, All functions	SSPI_CK	-
PC[2]	SSPI_SI	GPIO, All functions	SSPI_SI	-
PC[3]	SSPI_SO	GPIO, All functions	SSPI_SO	-
PC[4]	TDI	GPIO, All functions	TDI	-
PC[5]	TDO	GPIO, All functions	TDO	-
PC[6]	TMS	GPIO, All functions	TMS	-
PC[7]	TCK	GPIO, All functions	TCK	-
PD[0]	GPIO	All functions	SWM	Audio in/sar in
PD[1]	GPIO	All functions	SWM	led_strip_in1/sar_in
PD[2]	GPIO	All functions	SWM	32K xc2/led_strip_in2
PD[3]	GPIO	All functions	SWM	32K xc1/diag_hv_ana
PD[4][b]	GPIO	All functions	SWM	atb0
PD[5][c]	GPIO	All functions	SWM	atb1
PD[6][d]	GPIO	All functions	SWM	-
PD[7]	GPIO	All functions	SWM	-
PE[0]	GPIO	All functions	SWM	-
PE[1][e]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PE[6]	GPIO	All functions	SWM	-
PE[7]	GPIO	All functions	SWM	-
PF[0]	GPIO	All functions	SWM	-

Pad	Default	Function1	Function2	Analog Function
PF[1]	GPIO	All functions	SWM	-
PF[2]	GPIO	All functions	SWM	-
PF[3]	GPIO	All functions	SWM	-
PF[4]	GPIO	All functions	SWM	-
PF[5]	GPIO	All functions	SWM	-
PF[6]	GPIO	All functions	SWM	-
PF[7]	GPIO	All functions	SWM	-

a. "All functions" include 89 functions: SWM, RZ_TX, MSPI_CN1, LSPI_CN, TMR1_CMP, TMR0_CMP, PWM6_N, PWM6, PWM_SYNC, SSPI_SO, SSPI_SI, SSPI_CK, SSPI_CN, I2S2_CLK, I2S2_DAT1, I2S2_LR1, I2S2_DAT0, I2S2_LR0, I2S2_BCK, SDM1_N, SDM1_P, SDM0_N, SDM0_P, UART2_RTX, UART2_TX, UART2_RTS, UART2_CTS, IR_LEARN, ATSEL_5, ATSEL_4, GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, MSPI_CN3, MSPI_CN2, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT1, I2S1_LR1, I2S1_DAT0, I2S1_LR0, I2S1_BCK, I2S0_CLK, I2S0_DAT1, I2S0_LR1, I2S0_DAT0, I2S0_LR0, I2S0_BCK, CLK_7816, UART1_RTX, UART1_TX, UART1_RTS, UART1_CTS, UART0_RTX, UART0_TX, UART0_RTS, UART0_CTS, I2C_SDA, I2C_SCL, GSPI_MOSI, GSPI_MISO, GSPI_IO2, GSPI_IO3, GSPI_CK, GSPI_CN0, PWM5_N, PWM4_N, PWM3_N, PWM2_N, PWM1_N, PWM0_N, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0.

b. PD[4] is for internal use and not recommended for customer to use. Contact Telink FAE for details.

c. PD[5] is recommended to be used as output only, the details refer to the hardware design guideline.

d. PD[6] and PD[7] are not recommended to be used as PWM output.

e. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

1.6.4 Pin Assignment of TL7218J

Figure 1-13 Pin Assignment of TL7218J

	1	2	3	4	5	6	7	8
A	VDDIO_PEF	PF[7]	PF[4]	PF[6]	PF[3]	VLINE	VSS	ANT
B	PA[0]	PA[1]	PF[5]	VSS	PF[2]	PF[1]	PF[0]	VDDIO_PD
C	PA[2]	PA[3]	PA[4]	VSS	VSS	PE[3]	PE[0]	XC1
D	PA[5]	PA[6]	PA[7]	VSS	VSS	PE[4]	PE[1]	XC2
E	PB[0]	PB[1]	PB[2]	VSS	VDDO3	PE[5]	PE[2]	POR
F	PB[3]	PC[6]	PC[5]	DVSS	VDDPST	VDD_DEC	VDD_F	VBAT
G	VDDIO_PBC	VDDO1P8	VDDCORE	PC[4]	PC[7]	DCDC_SW	VDCDC	VBUS

Functions of 56 pins of TL7218J are described in table below.

Table 1-14 Pin Function of TL7218J

NO.	Pin Name	Type	Description
A1	VDDIO_PEF	PWR	IO voltage for PE0 ~ PE7, PF0 ~ PF7, PA0 ~ PA4, configurable to 3.3V or 1.8V
A2	PF[7]	GPIO	GPIO PF[7]
A3	PF[4]	GPIO	GPIO PF[4]
A4	PF[6]	GPIO	GPIO PF[6]
A5	PF[3]	GPIO	GPIO PF[3]
A6	VLINE	PWR	0.94V power supply of RF Transceiver
A7	VSS	GND	GND
A8	ANT	Analog	Pin to connect to the Antenna through the matching network
B1	PA[0]	GPIO	GPIO PA[0]
B2	PA[1]	GPIO	GPIO PA[1]
B3	PF[5]	GPIO	GPIO PF[5]
B4	VSS	GND	GND
B5	PF[2]	GPIO	GPIO PF[2]
B6	PF[1]	GPIO	GPIO PF[1]

NO.	Pin Name	Type	Description
B7	PF[0]	GPIO	GPIO PF[0]
B8	VDDIO_PD	PWR	leave unconnected for TL7218J
C1	PA[2]	GPIO	GPIO PA[2]
C2	PA[3]	GPIO	GPIO PA[3]
C3	PA[4]	GPIO	GPIO PA[4]
C4	VSS	GND	GND
C5	VSS	GND	GND
C6	PE[3]	GPIO	GPIO PE[3]
C7	PE[0]	GPIO	GPIO PE[0]
C8	XC1	Analog	Crystal oscillator pin 1
D1	PA[5]	GPIO	GPIO PA[5]
D2	PA[6]	GPIO	GPIO PA[6]
D3	PA[7]	GPIO	GPIO PA[7]
D4	VSS	GND	GND
D5	VSS	GND	GND
D6	PE[4]	GPIO	GPIO PE[4]
D7	PE[1]	GPIO	GPIO PE[1]
D8	XC2	Analog	Crystal oscillator pin 2
E1	PB[0]	GPIO	GPIO PB[0]
E2	PB[1]	GPIO	GPIO PB[1]
E3	PB[2]	GPIO	GPIO PB[2]
E4	VSS	GND	GND
E5	VDDO3	PWR	3.3V LDO output
E6	PE[5]	GPIO	GPIO PE[5]
E7	PE[2]	GPIO	GPIO PE[2]
E8	POR	Analog	Power on reset
F1	PB[3]	GPIO	GPIO PB[3]
F2	PC[6]	GPIO	GPIO PC[6]

NO.	Pin Name	Type	Description
F3	PC[5]	GPIO	GPIO PC[5]
F4	DVSS	GND	Digital ground
F5	VDDPST	PWR	3.3V input, connected with external capacitor
F6	VDD_DEC	PWR	0.94V digital power supply
F7	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor
F8	VBAT	PWR	Lion-Battery power supply
G1	VDDIO_PBC	PWR	IO voltage for PB0 ~ PB7, PC0 ~ PC7, PA7, configurable to 3.3V or 1.8V
G2	VDDO1P8	PWR	1.8V LDO output
G3	VDDCORE	PWR	Digital core power supply
G4	PC[4]	GPIO	GPIO PC[4]
G5	PC[7]	GPIO	GPIO PC[7]
G6	DCDC_SW	Analog	Connected with VDCDC via external inductor
G7	VDCDC	Analog	Connected with DCDC_SW via external inductor
G8	VBUS	PWR	5V USB power supply

GPIO pin mux functions of TL7218J are shown in the table below.

Table 1-15 GPIO Pin Mux of TL7218J

Pad	Default	Function1	Function2	Analog Function
PA[0]	GPIO	All functions[a]	SWM	-
PA[1]	GPIO	All functions	SWM	-
PA[2]	GPIO	All functions	SWM	-
PA[3]	GPIO	All functions	SWM	-
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[0]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[1]	GPIO	All functions	SWM	sar_in/lc_cmp_in

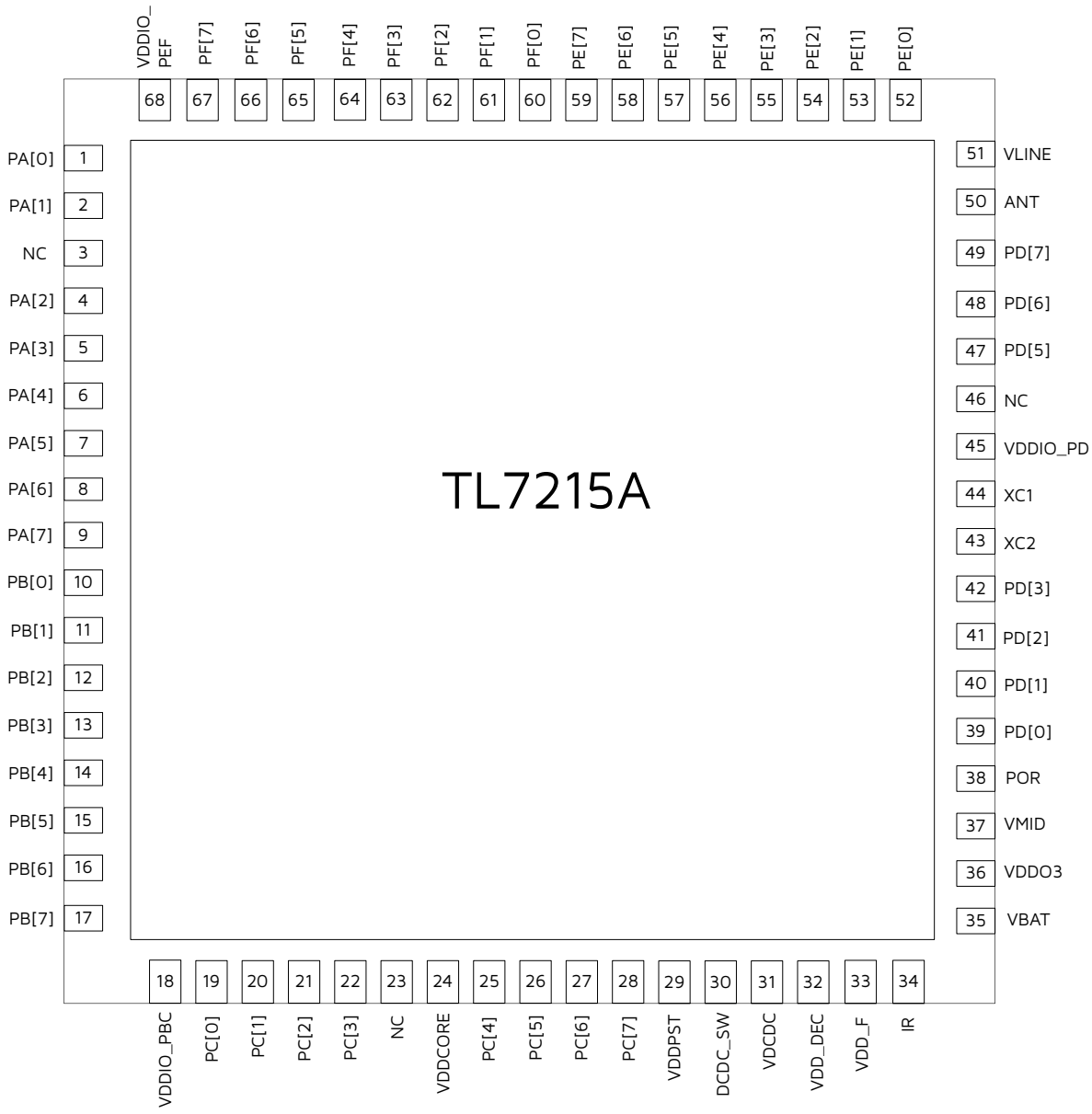
Pad	Default	Function1	Function2	Analog Function
PB[2]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[3]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PC[4]	TDI	GPIO, All functions	TDI	-
PC[5]	TDO	GPIO, All functions	TDO	-
PC[6]	TMS	GPIO, All functions	TMS	-
PC[7]	TCK	GPIO, All functions	TCK	-
PE[0]	GPIO	All functions	SWM	-
PE[1][b]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PF[0]	GPIO	All functions	SWM	-
PF[1]	GPIO	All functions	SWM	-
PF[2]	GPIO	All functions	SWM	-
PF[3]	GPIO	All functions	SWM	-
PF[4]	GPIO	All functions	SWM	-
PF[5]	GPIO	All functions	SWM	-
PF[6]	GPIO	All functions	SWM	-
PF[7]	GPIO	All functions	SWM	-

a. "All functions" include 89 functions: SWM, RZ_TX, MSPI_CN1, LSPI_CN, TMR1_CMP, TMR0_CMP, PWM6_N, PWM6, PWM_SYNC, SSPI_SO, SSPI_SI, SSPI_CK, SSPI_CN, I2S2_CLK, I2S2_DAT1, I2S2_LR1, I2S2_DAT0, I2S2_LR0, I2S2_BCK, SDM1_N, SDM1_P, SDM0_N, SDM0_P, UART2_RTX, UART2_TX, UART2_RTS, UART2_CTS, IR_LEARN, ATSEL_5, ATSEL_4, GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, MSPI_CN3, MSPI_CN2, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT1, I2S1_LR1, I2S1_DAT0, I2S1_LR0, I2S1_BCK, I2S0_CLK, I2S0_DAT1, I2S0_LR1, I2S0_DAT0, I2S0_LR0, I2S0_BCK, CLK_7816, UART1_RTX, UART1_TX, UART1_RTS, UART1_CTS, UART0_RTX, UART0_TX, UART0_RTS, UART0_CTS, I2C_SDA, I2C_SCL, GSPI_MOSI, GSPI_MISO, GSPI_IO2, GSPI_IO3, GSPI_CK, GSPI_CN0, PWM5_N, PWM4_N, PWM3_N, PWM2_N, PWM1_N, PWM0_N, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0.

b. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

1.6.5 Pin Assignment of TL7215A

Figure 1-14 Pin Assignment of TL7215A



Functions of 68 pins of TL7215A are described in table below.

Table 1-16 Pin Function of TL7215A

No.	Pin Name	Type	Description
1	PA[0]	GPIO	GPIO PA[0]
2	PA[1]	GPIO	GPIO PA[1]
3	NC	-	Not connected
4	PA[2]	GPIO	GPIO PA[2]
5	PA[3]	GPIO	GPIO PA[3]

No.	Pin Name	Type	Description
6	PA[4]	GPIO	GPIO PA[4]
7	PA[5]	GPIO	GPIO PA[5]
8	PA[6]	GPIO	GPIO PA[6]
9	PA[7]	GPIO	GPIO PA[7]
10	PB[0]	GPIO	GPIO PB[0]
11	PB[1]	GPIO	GPIO PB[1]
12	PB[2]	GPIO	GPIO PB[2]
13	PB[3]	GPIO	GPIO PB[3]
14	PB[4]	GPIO	GPIO PB[4]
15	PB[5]	GPIO	GPIO PB[5]
16	PB[6]	GPIO	GPIO PB[6]
17	PB[7]	GPIO	GPIO PB[7]
18	VDDIO_PBC	PWR	IO voltage for PB0 ~ PB7, PC0 ~ PC7 and PA7, configurable to 3.3V or 1.8V
19	PC[0]	GPIO	GPIO PC[0]
20	PC[1]	GPIO	GPIO PC[1]
21	PC[2]	GPIO	GPIO PC[2]
22	PC[3]	GPIO	GPIO PC[3]
23	NC	-	Not connected
24	VDDCORE	PWR	Digital core power supply
25	PC[4]	GPIO	GPIO PC[4]
26	PC[5]	GPIO	GPIO PC[5]
27	PC[6]	GPIO	GPIO PC[6]
28	PC[7]	GPIO	GPIO PC[7]
29	VDDPST	PWR	3.3V input, connected with external capacitor
30	DCDC_SW	Analog	Connected with VDCDC via external inductor
31	VDCDC	Analog	Connected with DCDC_SW via external inductor
32	VDD_DEC	PWR	0.94V digital power supply
33	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor

No.	Pin Name	Type	Description
34	IR	Analog	Infrared radiation learning
35	VBAT	PWR	Lion-Battery power supply
36	VDDO3	PWR	3.3V LDO output
37	VMID	Analog	Audio reference voltage, 0V by default, 0.9V when audio is enabled
38	POR	Analog	Power on reset
39	PD[0]	GPIO	GPIO PD[0]
40	PD[1]	GPIO	GPIO PD[1]
41	PD[2]	GPIO	GPIO PD[2]
42	PD[3]	GPIO	GPIO PD[3]
43	XC2	Analog	Crystal oscillator pin 2
44	XC1	Analog	Crystal oscillator pin 1
45	VDDIO_PD	PWR	IO voltage for PD0 ~ PD7, configurable to 3.3V or 1.8V
46	NC	-	Not connected
47	PD[5]	GPIO	GPIO PD[5]
48	PD[6]	GPIO	GPIO PD[6]
49	PD[7]	GPIO	GPIO PD[7]
50	ANT	Analog	Pin to connect to the antenna through the matching network
51	VLINE	PWR	0.94V power supply of RF Transceiver
52	PE[0]	GPIO	GPIO PE[0]
53	PE[1]	GPIO	GPIO PE[1]
54	PE[2]	GPIO	GPIO PE[2]
55	PE[3]	GPIO	GPIO PE[3]
56	PE[4]	GPIO	GPIO PE[4]
57	PE[5]	GPIO	GPIO PE[5]
58	PE[6]	GPIO	GPIO PE[6]
59	PE[7]	GPIO	GPIO PE[7]
60	PF[0]	GPIO	GPIO PF[0]
61	PF[1]	GPIO	GPIO PF[1]

No.	Pin Name	Type	Description
62	PF[2]	GPIO	GPIO PF[2]
63	PF[3]	GPIO	GPIO PF[3]
64	PF[4]	GPIO	GPIO PF[4]
65	PF[5]	GPIO	GPIO PF[5]
66	PF[6]	GPIO	GPIO PF[6]
67	PF[7]	GPIO	GPIO PF[7]
68	VDDIO_PEF	PWR	IO voltage for PE0 ~ PE7, PF0 ~ PF7, PA0 ~ PA4, configurable to 3.3V or 1.8V

GPIO pin mux functions of TL7215A are shown in the table below.

Table 1-17 GPIO Pin Mux of TL7215A

Pad	Default	Function1	Function2	Analog Function
PA[0]	GPIO	All functions[a]	SWM	-
PA[1]	GPIO	All functions	SWM	-
PA[2]	GPIO	All functions	SWM	-
PA[3]	GPIO	All functions	SWM	-
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[0]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[1]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[2]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[3]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[4]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[5]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[6]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[7]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PC[0]	SSPI_CN	GPIO, All functions	SSPI_CN	-
PC[1]	SSPI_CK	GPIO, All functions	SSPI_CK	-

Pad	Default	Function1	Function2	Analog Function
PC[2]	SSPI_SI	GPIO, All functions	SSPI_SI	-
PC[3]	SSPI_SO	GPIO, All functions	SSPI_SO	-
PC[4]	TDI	GPIO, All functions	TDI	-
PC[5]	TDO	GPIO, All functions	TDO	-
PC[6]	TMS	GPIO, All functions	TMS	-
PC[7]	TCK	GPIO, All functions	TCK	-
PD[0]	GPIO	All functions	SWM	Audio in/sar in
PD[1]	GPIO	All functions	SWM	led_strip_in1/sar_in
PD[2]	GPIO	All functions	SWM	32K xc2/led_strip_in2
PD[3]	GPIO	All functions	SWM	32K xc1/diag_hv_ana
PD[5][b]	GPIO	All functions	SWM	atb1
PD[6][c]	GPIO	All functions	SWM	-
PD[7]	GPIO	All functions	SWM	-
PE[0]	GPIO	All functions	SWM	-
PE[1][d]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PE[6]	GPIO	All functions	SWM	-
PE[7]	GPIO	All functions	SWM	-
PF[0]	GPIO	All functions	SWM	-
PF[1]	GPIO	All functions	SWM	-
PF[2]	GPIO	All functions	SWM	-
PF[3]	GPIO	All functions	SWM	-
PF[4]	GPIO	All functions	SWM	-
PF[5]	GPIO	All functions	SWM	-
PF[6]	GPIO	All functions	SWM	-

Pad	Default	Function1	Function2	Analog Function
PF[7]	GPIO	All functions	SWM	-

a. "All functions" include 89 functions: SWM, RZ_TX, MSPI_CN1, LSPI_CN, TMR1_CMP, TMR0_CMP, PWM6_N, PWM6, PWM_SYNC, SSPI_SO, SSPI_SI, SSPI_CK, SSPI_CN, I2S2_CLK, I2S2_DAT1, I2S2_LR1, I2S2_DAT0, I2S2_LR0, I2S2_BCK, SDM1_N, SDM1_P, SDM0_N, SDM0_P, UART2_RTX, UART2_TX, UART2_RTS, UART2_CTS, IR_LEARN, ATSEL_5, ATSEL_4, GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, MSPI_CN3, MSPI_CN2, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT1, I2S1_LR1, I2S1_DAT0, I2S1_LR0, I2S1_BCK, I2S0_CLK, I2S0_DAT1, I2S0_LR1, I2S0_DAT0, I2S0_LR0, I2S0_BCK, CLK_7816, UART1_RTX, UART1_TX, UART1_RTS, UART1_CTS, UART0_RTX, UART0_TX, UART0_RTS, UART0_CTS, I2C_SDA, I2C_SCL, GSPI_MOSI, GSPI_MISO, GSPI_IO2, GSPI_IO3, GSPI_CK, GSPI_CN0, PWM5_N, PWM4_N, PWM3_N, PWM2_N, PWM1_N, PWM0_N, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0.

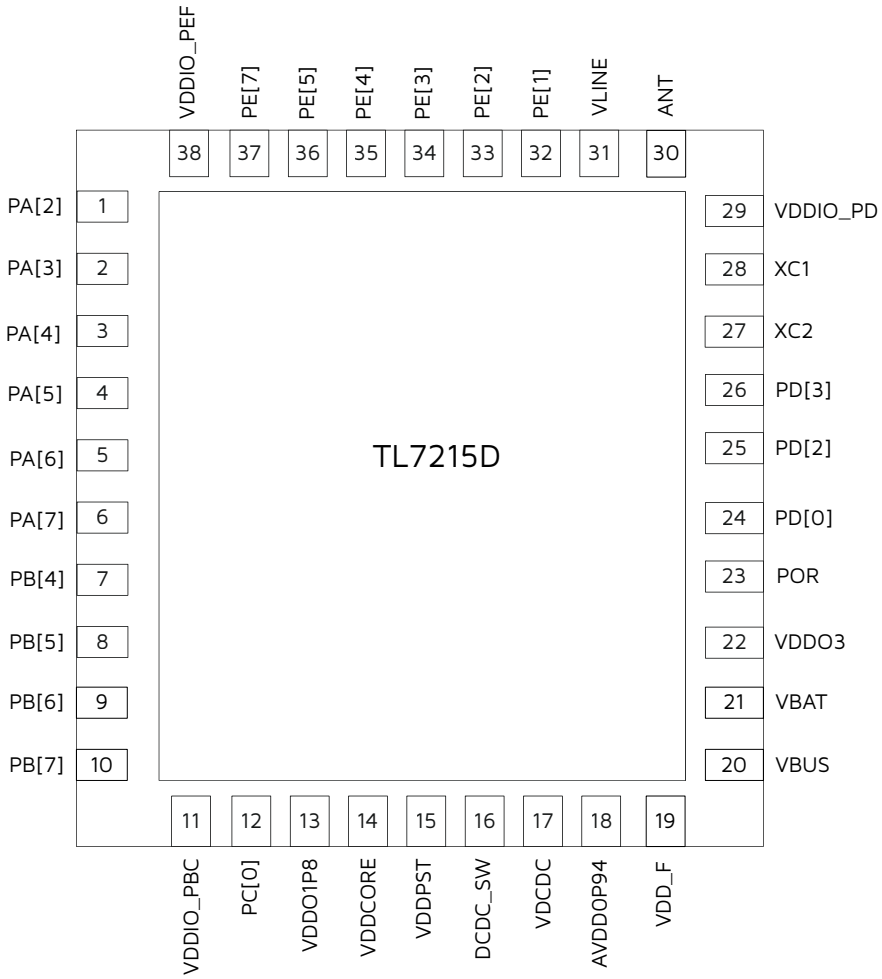
b. (1) PD[5], PD[6], PD[7] of TL7215A are not used for Channel Sounding application. (2) PD[5] is recommended to be used as output only, the details refer to the hardware design guideline.

c. PD[6] and PD[7] are not recommended to be used as PWM output.

d. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

1.6.6 Pin Assignment of TL7215D

Figure 1-15 Pin Assignment of TL7215D



Functions of 38 pins of TL7215D are described in table below.

Table 1-18 Pin Function of TL7215D

NO.	Pin Name	Type	Description
1	PA[2]	GPIO	GPIO PA[2]
2	PA[3]	GPIO	GPIO PA[3]
3	PA[4]	GPIO	GPIO PA[4]
4	PA[5]	GPIO	GPIO PA[5]
5	PA[6]	GPIO	GPIO PA[6]
6	PA[7]	GPIO	GPIO PA[7]
7	PB[4]	GPIO	GPIO PB[4]
8	PB[5]	GPIO	GPIO PB[5]

NO.	Pin Name	Type	Description
9	PB[6]	GPIO	GPIO PB[6]
10	PB[7]	GPIO	GPIO PB[7]
11	VDDIO_PBC	PWR	IO voltage for PB4 ~ PB7, PC0 and PA7, configurable to 3.3V or 1.8V
12	PC[0]	GPIO	GPIO PC[0]
13	VDDO1P8	PWR	1.8V LDO output
14	VDDCORE	PWR	Digital core power supply
15	VDDPST	PWR	3.3V input, connected with external capacitor
16	DCDC_SW	Analog	Connected with VDCDC via external inductor
17	VDCDC	Analog	Connected with DCDC_SW via external inductor
18	AVDD0P94	PWR	0.94V analog power supply
19	VDD_F	PWR	Internally generated power supply to flash. Connect to GND via external capacitor
20	VBUS	PWR	5V USB power supply
21	VBAT	PWR	Lion-Battery power supply
22	VDDO3	PWR	3.3V LDO output
23	POR	Analog	Power on reset
24	PD[0]	GPIO	GPIO PD[0]
25	PD[2]	GPIO	GPIO PD[2]
26	PD[3]	GPIO	GPIO PD[3]
27	XC2	Analog	Crystal oscillator pin 2
28	XC1	Analog	Crystal oscillator pin 1
29	VDDIO_PD	PWR	IO voltage for PD0, PD2, PD3, configurable to 3.3V or 1.8V
30	ANT	Analog	Pin to connect to the Antenna through the matching network
31	VLINE	PWR	0.94V power supply of RF Transceiver
32	PE[1]	GPIO	GPIO PE[1]
33	PE[2]	GPIO	GPIO PE[2]
34	PE[3]	GPIO	GPIO PE[3]
35	PE[4]	GPIO	GPIO PE[4]

NO.	Pin Name	Type	Description
36	PE[5]	GPIO	GPIO PE[5]
37	PE[7]	GPIO	GPIO PE[7]
38	VDDIO_PEF	PWR	IO voltage for PE1 ~ PE5, PE7, PA2 ~ PA4, configurable to 3.3V or 1.8V

GPIO pin mux functions of TL7215D are shown in the table below.

Table 1-19 GPIO Pin Mux of TL7215D

Pad	Default	Function1	Function2	Analog Function
PA[2]	GPIO	All functions[a]	SWM	-
PA[3]	GPIO	All functions	SWM	-
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[4]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[5]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[6]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PB[7]	GPIO	All functions	SWM	sar_in/lc_cmp_in
PC[0]	SSPI_CN	GPIO, All functions	SSPI_CN	-
PD[0]	GPIO	All functions	SWM	Audio in/sar in
PD[2]	GPIO	All functions	SWM	32K xc2/led_strip_in2
PD[3]	GPIO	All functions	SWM	32K xc1/diag_hv_ana
PE[1][b]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PE[7]	GPIO	All functions	SWM	-

a. "All functions" include 89 functions: SWM, RZ_TX, MSPI_CN1, LSPI_CN, TMR1_CMP, TMR0_CMP, PWM6_N, PWM6, PWM_SYNC, SSPI_SO, SSPI_SI, SSPI_CK, SSPI_CN, I2S2_CLK, I2S2_DAT1, I2S2_LR1, I2S2_DAT0, I2S2_LR0, I2S2_BCK, SDM1_N, SDM1_P, SDM0_N, SDM0_P, UART2_RTX, UART2_TX, UART2_RTS, UART2_CTS, IR_LEARN, ATSEL_5, ATSEL_4, GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, MSPI_CN3, MSPI_CN2, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT1, I2S1_LR1, I2S1_DAT0, I2S1_LR0, I2S1_BCK, I2S0_CLK, I2S0_DAT1, I2S0_LR1, I2S0_DAT0, I2S0_LR0, I2S0_BCK, CLK_7816, UART1_RTX, UART1_TX, UART1_RTS, UART1_CTS, UART0_RTX, UART0_TX, UART0_RTS, UART0_CTS, I2C_SDA, I2C_SCL, GSPI_MOSI, GSPI_MISO, GSPI_IO2, GSPI_IO3, GSPI_CK, GSPI_CN0, PWM5_N, PWM4_N, PWM3_N, PWM2_N, PWM1_N, PWM0_N, PWM5, PWM4, PWM3, PWM2, PWM1, PWM0.

b. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

Descriptions of each signal are listed in Table 1-20 to Table 1-38.

NOTE: *Insufficient pins lead to lack of corresponding function including I2C, I2S, UART, DMIC, USB, JTAG, SDP and PTA.*

Table 1-20 PWM Signal Description

Signal	Type	Description
PWM0	DO	PWM channel 0 output
PWM0_N	DO	PWM channel 0 inversion output
PWM1	DO	PWM channel 1 output
PWM1_N	DO	PWM channel 1 inversion output
PWM2	DO	PWM channel 2 output
PWM2_N	DO	PWM channel 2 inversion output
PWM3	DO	PWM channel 3 output
PWM3_N	DO	PWM channel 3 inversion output
PWM4	DO	PWM channel 4 output
PWM4_N	DO	PWM channel 4 inversion output
PWM5	DO	PWM channel 5 output
PWM5_N	DO	PWM channel 5 inversion output
PWM6	DO	PWM channel 6 output
PWM6_N	DO	PWM channel 6 inversion output
PWM_SYNC	DO	PWM synchronization

Table 1-21 I2C Signal Description

Signal	Type	Description
I2C_SCL	DIO	I2C SCL
I2C_SDA	DIO	I2C SDA

Table 1-22 I2S Signal Description

Signal	Type	Description
I2S_BCK	DIO	I2S bit CLK
I2S_CLK	DO	I2S base CLK
I2S_LR1	DIO	I2S left and right channel SEL
I2S_LR0	DIO	I2S left and right channel SEL
I2S_DAT1	DI	I2S data IN
I2S_DAT0	DO	I2S data OUT

Table 1-23 UART Signal Description

Signal	Type	Description
UART_CTS	DI	UART Clear to Send signal
UART_RTS	DO	UART Ready to Send signal
UART_RTX	DIO	UART RTX
UART_TX	DO	UART TX

Table 1-24 Audio Output Signal Description

Signal	Type	Description
SDM_N	DO	SDM diff output
SDM_P	DO	SDM diff output

Table 1-25 GSPI Signal Description

Signal	Type	Description
GSPI_CK	DIO	GSPI CLK
GSPI_CN0	DIO	GSPI CN0
GSPI_CN1	DIO	GSPI CN1

Signal	Type	Description
GSPI_CN2	DIO	GSPI CN2
GSPI_CN3	DIO	GSPI CN3
GSPI_MISO	DIO	GSPI MISO
GSPI_MOSI	DIO	GSPI MOSI
GSPI_IO2	DIO	GSPI IO2
GSPI_IO3	DIO	GSPI IO3

Table 1-26 LSPI Signal Description

Signal	Type	Description
LSPI_CK	DIO	LSPI CLK
LSPI_CN	DIO	LSPI CN
LSPI_MISO	DIO	LSPI MISO
LSPI_MOSI	DIO	LSPI MOSI
LSPI_IO2	DIO	LSPI IO2
LSPI_IO3	DIO	LSPI IO3

Table 1-27 SPI Slave Signal Description

Signal	Type	Description
SSPI_CK	DI	SSPI CLK
SSPI_CN	DI	SSPI CN
SSPI_SI	DIO	SSPI SI
SSPI_SO	DIO	SSPI SO

Table 1-28 7816 Signal Description

Signal	Type	Description
CLK_7816	DO	7816 CLK

Table 1-29 DMIC Signal Description

Signal	Type	Description
DMIC_CLK	DO	DMIC CLK

Signal	Type	Description
DMIC_DAT	DI	DMIC DATA IN

Table 1-30 Swire Signal Description

Signal	Type	Description
SWM	DIO	Swire Master
SWS	DIO	Swire Slave

Table 1-31 External Power Amplifier, Low Noise Amplifier Signal Description

Signal	Type	Description
RX_CYC2LNA	DO	External low noise amplifier
TX_CYC2PA	DO	External power amplifier

Table 1-32 USB Signal Description

Signal	Type	Description
DP	DIO	USB DP
DM	DIO	USB DM

Table 1-33 JTAG Signal Description

Signal	Type	Description
TDI	DI	Test data input
TDO	DO	Test data output
TMS	DIO	Test mode selection
TCK	DI	Test clock input

Table 1-34 SDP Signal Description

Signal	Type	Description
TCK	DI	Test clock input
TMS	DIO	Test mode selection, data input/output

Table 1-35 PTA Signal Description

Signal	Type	Description
BT_ACTIVITY	DIO	Bluetooth activity
BT_STATUS	DIO	Bluetooth status
WIFI_DENY	DI	WiFi deny

Table 1-36 ATSEL Signal Description

Signal	Type	Description
ATSEL_5	DIO	AoA/AoD antenna selection 5
ATSEL_4	DIO	AoA/AoD antenna selection 4
ATSEL_3	DIO	AoA/AoD antenna selection 3
ATSEL_2	DIO	AoA/AoD antenna selection 2
ATSEL_1	DIO	AoA/AoD antenna selection 1
ATSEL_0	DIO	AoA/AoD antenna selection 0

Table 1-37 Low Current Comparator Signal Description

Signal	Type	Description
lc_comp<0>	AI	Low current comparator channel 0
lc_comp<1>	AI	Low current comparator channel 1
lc_comp<2>	AI	Low current comparator channel 2
lc_comp<3>	AI	Low current comparator channel 3
lc_comp<4>	AI	Low current comparator channel 4
lc_comp<5>	AI	Low current comparator channel 5
lc_comp<6>	AI	Low current comparator channel 6
lc_comp<7>	AI	Low current comparator channel 7

Table 1-38 SAR ADC Signal Description

Signal	Type	Description
sar_in<0>	AI	SAR ADC input channel 0
sar_in<1>	AI	SAR ADC input channel 1
sar_in<2>	AI	SAR ADC input channel 2

Signal	Type	Description
sar_in<3>	AI	SAR ADC input channel 3
sar_in<4>	AI	SAR ADC input channel 4
sar_in<5>	AI	SAR ADC input channel 5
sar_in<6>	AI	SAR ADC input channel 6
sar_in<7>	AI	SAR ADC input channel 7
sar_in<8>	AI	SAR ADC input channel 8
sar_in<9>	AI	SAR ADC input channel 9

Table 1-39 Crystal Signal Description

Signal	Type	Description
xtl32k_out	AO	32kHz crystal output pin
xtl32k_in	AI	32kHz crystal input pin

NOTE:

- DI: Digital input
- DO: Digital output
- DIO: Digital input/output
- AI: Analog input
- AO: Analog output
- AIO: Analog input/output

2 Electrical Specifications

NOTE: *The electrical characteristics currently listed in this section are target specifications and only provided for reference. Some data may be updated according to actual test results.*

2.1 Absolute Maximum Rating

Table 2-1 Absolute Maximum Rating

Characteristics	Sym.	Min.	Max.	Unit	Test Condition
Supply voltage	VBAT	-0.3	4.3	V	-
USB voltage	VBUS	-0.3	5.5	V	Exclude TL7218A & TL7215A
Voltage on input pin	V_{in}	-0.3	VDD+0.3	V	-
Output voltage	V_{out}	0	VDD	V	-
Storage temperature range	T_{Str}	-65	150	°C	-
Soldering temperature	T_{Sld}	-	260	°C	-

NOTE:

- Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.
- VDD stands for IO voltage, the IO voltage for the GPIO pin refers to the specific group it belongs.

2.2 Recommended Operating Conditions

Table 2-2 Recommend Operating Conditions

Item	Sym.	Min.	Typ.	Max.	Unit	Condition
Power supply voltage	VBAT	1.8	3.7	4.3	V	-
USB supply voltage	VBUS	4.5	5.0	5.5	V	Exclude TL7218A & TL7215A
Supply rise time (from 1.6 V to 1.8 V)	T_R	-	-	10	ms	-
Operating temperature range	T_{Opr}	-40	-	85	°C	-

2.3 DC Characteristics

Unless otherwise stated, the general test conditions are: T = 25°C.

Table 2-3 RX/TX Current and Sleep Current

Item	Sym.	Min	Typ.	Max	Unit	Conditions
RX current	I_{Rx}	-	1.6[a]	-	mA	Whole chip, 4.2 V DCDC, BLE mode[b]
		-	2.0	-	mA	Whole chip, 3.3 V DCDC, BLE mode
		-	5.5	-	mA	Whole chip, LDO mdoe
TX current	I_{Tx}	-	2.4	-	mA	Whole chip @ 0 dBm with 4.2 V DCDC, BLE mode
		-	2.9	-	mA	Whole chip @ 0 dBm with 3.3 V DCDC, BLE mode
		-	8.8	-	mA	Whole chip @ 0 dBm LDO mode
Deep sleep with 32 KB SRAM retention	I_{Deep1}	-	2.0	-	µA	3.3V, without 32K RC[c]
Deep sleep with 64 KB SRAM retention		-	2.9	-	µA	
Deep sleep with 128 KB SRAM retention		-	4.6		µA	
Deep sleep with 256 KB SRAM retention		-	7.9		µA	
Deep sleep without SRAM retention	I_{Deep2}	-	1.0	-	µA	3.3V, without 32K RC
Deep sleep with 32 KB SRAM retention	I_{Deep3}	-	2.5	-	µA	3.3V, with 32K RC[d]
Deep sleep with 64 KB SRAM retention		-	3.3	-	µA	
Deep sleep with 128 KB SRAM retention			5.0		µA	
Deep sleep with 256 KB SRAM retention			8.4		µA	
Deep sleep without SRAM retention	I_{Deep4}	-	1.5	-	µA	3.3V, with 32K RC
Suspend current	I_{Susp}	-	0.15	-	mA	

a. The current data is based on the test results of the chip for evaluation kit, and they are to be updated after all chips are tested.

b. The complete test conditions for TX/RX current: (1) turn off the clock and power of the modules irrelevant to RF; (2) hold the modules irrelevant to RF; (3) disable PLL clock and RC_24M to make the Pad_24M is the only clock source for cclk; (4) stall the MCU.

c. Without 32K RC: The wakeup source is external signal from GPIO input, the internal 32K RC is disabled.

d. With 32K RC: The wakeup source is 32K RC, it is enabled.

Table 2-4 Digital Inputs/Outputs

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
Input high voltage	V_{IH}	0.7*VDD	-	VDD	V	-
Input low voltage	V_{IL}	VSS	-	0.3*VDD	V	-
Output high voltage	V_{OH}	0.9*VDD	-	VDD	V	-
Output low voltage	V_{OL}	VSS	-	0.1*VDD	V	-

2.4 AC Characteristics

NOTE: *The data in this section is only provided for reference and is to be updated after all chips are tested.*

2.4.1 RF Performance

Unless otherwise stated, the general test conditions are: VDD = 4.2 V, T = 25°C.

Table 2-5 RF Performance Characteristics

Item		Sym.	Min.	Typ.	Max.	Unit	Conditions
RF frequency range		-	2400	-	2483.5	MHz	Programmable in 1 MHz step
Data rate		Bluetooth LE/2.4G proprietary 1 Mbps, ±250 kHz deviation Bluetooth LE/2.4G proprietary 2 Mbps, ±500 kHz deviation Bluetooth LE 125 kbps, ±250 kHz deviation Bluetooth LE 500 kbps, ±250 kHz deviation IEEE 802.15.4 250 kbps, ±500 kHz deviation 2.4GHz proprietary 250 kbps, ±62.5 kHz deviation 2.4GHz proprietary 500 kbps, ±125 kHz deviation					
Bluetooth LE 1 Mbps RF_RX Performance (±250 kHz Deviation)							
Sensitivity	1 Mbps	-	-	-96	-	dBm	-
Frequency offset tolerance		-	-200	-	+200	kHz	-
Co-channel rejection		-	-	8	-	dB	Wanted signal at -67 dBm

Item		Sym.	Min.	Typ.	Max.	Unit	Conditions
In-band blocking rejection (equal modulation interference)	+1/-1 MHz offset	-	-	-4/-3	-	dB	Wanted signal at -67 dBm
	+2/-2 MHz offset	-	-	-41/-41	-	dB	
	≥ 3 MHz offset	-	-	-49	-	dB	
Image rejection		-	-	-38	-	dB	Wanted signal at -67 dBm; image frequency = RF_channel - 2MHz
Bluetooth LE 1 Mbps RF_TX Performance							
Output power, maximum setting		-	-	10	-	dBm	-
Output power, minimum setting		-	-	-24	-	dBm	-
Programmable output power range		-	34			dB	-
Modulation 20 dB bandwidth		-	-	1.4	-	MHz	-
Bluetooth LE 2 Mbps RF_RX Performance (±500 kHz Deviation)							
Sensitivity	2 Mbps	-	-	-93	-	dBm	-
Frequency offset tolerance		-	-300	-	+300	kHz	-
Co-channel rejection		-	-	8	-	dB	Wanted signal at -67 dBm
In-band blocking rejection	+2/-2 MHz offset	-	-	-4/-4	-	dB	Wanted signal at -67 dBm
	+4/-4 MHz offset	-	-	-35/-35	-	dB	
	> 6 MHz offset	-	-	-39	-	dB	
Image rejection		-	-	-25	-	dB	Wanted signal at -67 dBm; image frequency = RF_channel - 3 MHz

Item		Sym.	Min.	Typ.	Max.	Unit	Conditions
Bluetooth LE 2 Mbps RF_TX Performance							
Output power, maximum setting		-	-	10	-	dBm	-
Output power, minimum setting		-	-	-24	-	dBm	-
Programmable output power range				34		dB	-
Modulation 20 dB bandwidth		-	-	2.5	-	MHz	-
Bluetooth LE 125 kbps RF_RX Performance (±250kHz Deviation)							
Sensitivity	125 kbps	-	-	-103	-	dBm	-
Frequency offset tolerance		-	-200	-	+200	kHz	-
Co-channel rejection		-	-	2	-	dB	Wanted signal at -79 dBm
In-band blocking rejection (equal modulation interference)	+1/-1 MHz offset	-	-	-6/-6	-	dB	Wanted signal at -79 dBm
	+2/-2 MHz offset	-	-	-38/-38	-	dB	
	≥ 3 MHz offset	-	-	-48/-48	-	dB	
Image rejection		-	-	-27	-	dB	Wanted signal at -79 dBm; image frequency = RF_channel - 2 MHz
Bluetooth LE 125 kbps RF_TX Performance							
Output power, maximum setting		-	-	10	-	dBm	-
Output power, minimum setting (resolution)		-	-	-24	-	dBm	-
Programmable output power range		-		34		dB	-
Modulation 20 dB bandwidth		-	-	1.4	-	MHz	-

Item		Sym.	Min.	Typ.	Max.	Unit	Conditions
Bluetooth LE 500 kbps RF_RX Performance (±250 kHz Deviation)							
Sensitivity	500 kbps	-	-	-99	-	dBm	-
Frequency offset tolerance		-	-200	-	+200	kHz	-
Co-channel rejection		-	-	4	-	dB	Wanted signal at -72 dBm
In-band blocking rejection (equal modulation interference)	+1/-1 MHz offset	-	-	-6/-6	-	dB	Wanted signal at -72 dBm
	+2/-2 MHz offset	-	-	-42/-41	-	dB	
	≥ 3 MHz offset	-	-	-55	-	dB	
Image rejection		-	-	-40	-	dB	Wanted signal at -72 dBm; image frequency = RF_channel - 2 MHz
Bluetooth LE 500 kbps RF_TX Performance							
Output power, maximum setting		-	-	10	-	dBm	-
Output power, minimum setting		-	-	-24	-	dBm	-
Programmable output power range		-	34			dB	-
Modulation 20 dB bandwidth		-	-	1.4	-	MHz	-
Zigbee/RF4CE/6LoWPan/Thread RF_RX Performance (IEEE 802.15.4 250 kbps ±500 kHz Deviation)							
Sensitivity	250 kbps	-	-	-103	-	dBm	-
Frequency offset tolerance		-	-300	-	+300	kHz	-
Adjacent channel rejection (-1/+1 channel)		-	-	-38/-38	-	dB	Wanted signal at -82 dBm
Adjacent channel rejection (-2/+2 channel)		-	-	-38/-38	-	dB	Wanted signal at -82 dBm
Zigbee/RF4CE/6LoWPan/Thread RF_TX Performance (IEEE 802.15.4 250 kbps)							

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
Output power, maximum setting	-	-	10	-	dBm	-
Output power, minimum setting (resolution)	-	-	-24	-	dBm	-
Programmable output power range	-	34			dB	-
Modulation 20 dB bandwidth	-	-	2.7	-	MHz	-
Error vector magnitude	EVM	-	-	2%	-	Max (10 dBm) power output

Table 2-6 USB Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
USB output signal cross-over voltage	V_{Crs}	1.3	-	2.0	V	-

Table 2-7 RSSI Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
RSSI range	-	-100	-	0	dBm	-
Resolution	-	-	±1	-	dB	-

Table 2-8 Crystal Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
24 MHz Crystal						
Nominal frequency (parallel resonant)	f_{NOM}	-	24	-	MHz	-
Frequency tolerance	f_{TOL}	-10	-	+10	ppm	-
Load capacitance	C_L	-	6	-	pF	Not include the stray capacitance on the PCB.
Equivalent series resistance	ESR	-	50	100	Ohm	-
32.768 kHz Crystal						

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
Nominal frequency (parallel resonant)	f_{NOM}	-	32.768	-	kHz	-
Frequency tolerance	f_{TOL}	-100	-	+100	ppm	-
Load capacitance	C_L	6	9	12.5	pF	-
Equivalent series resistance	ESR	-	50	80	kOhm	-

Table 2-9 RC Oscillator Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
24 MHz RC Oscillator						
Nominal frequency	f_{NOM}	-	24	-	MHz	-
Frequency tolerance	f_{TOL}	-	1	-	%	On chip calibration
32 kHz RC Oscillator						
Nominal frequency	f_{NOM}	-	32	-	kHz	-
Frequency tolerance	f_{TOL}	-	0.1	-	%	On chip calibration
Calibration time	-	-	3	-	ms	-

Table 2-10 ADC Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
Differential nonlinearity	DNL	-	-	1	LSB	
Integral nonlinearity	INL	-	-	2	LSB	
Signal-to-noise and distortion ratio	SINAD	-	65	-	dB	f_{IN} = 1 kHz, f_S= 16 kHz
Effective number of bits	ENOB	-	10.5	-	bits	-
Sampling frequency	Fs	-	-	192	ksps	-

2.4.2 Audio Performance

2.4.2.1 Analog Input to Digital Output

Measurement conditions: Input sine wave with a frequency of 1kHz, measurement bandwidth 20Hz– Fs / 2 for Fs = 8 to 32 kHz, measurement bandwidth 20 Hz to 20 kHz for Fs = 44.1 kHz to 96 kHz, unless otherwise specified.

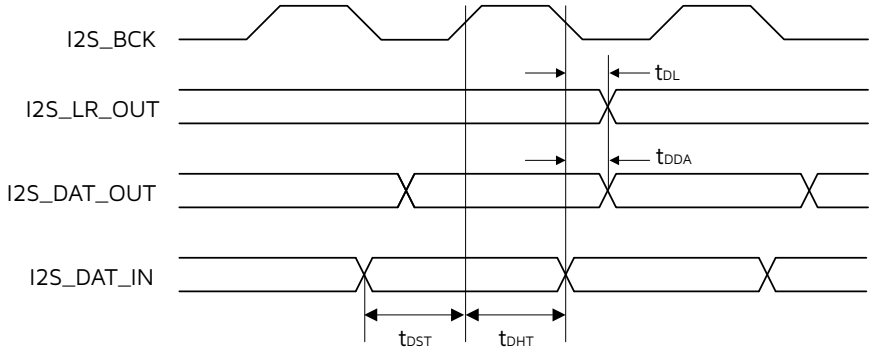
Table 2-11 Analog Microphone / Line Input to ADC Path

Parameter	Test conditions	Min.	Typ	Max.	Unit
SNR	500mVp input of 1.02 kHz, 0dB PGA gain	-	78.5	-	dB
THD+N	normal performance	-	-74	-	dB

2.4.3 I2S Performance

The I2S Timing is shown as below.

Figure 2-1 I2S Timing - Master Mode



Measurement conditions: VDD = 3.3 V, T = 25°C, I2S_BCK = 3.072 MHz, LRCLK = 48 kHz, unless otherwise specified.

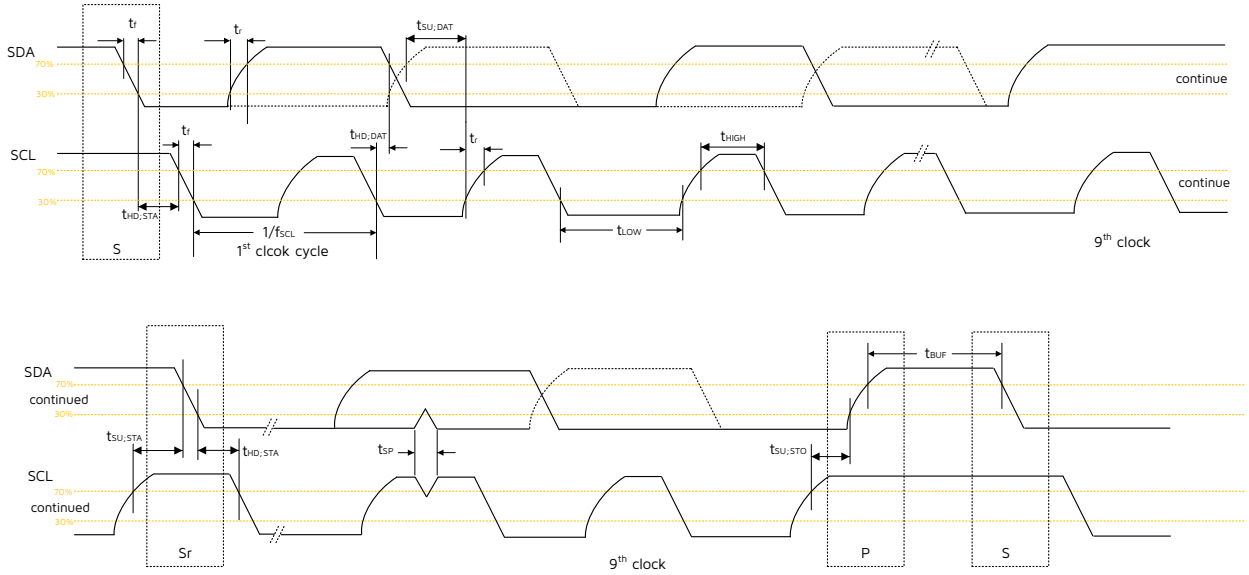
Table 2-12 I2S Timing Sequence

Symbol	Parameter	Min.	Typ	Max.	Unit
t_{DL}	I2S_LR_OUT propagation delay from BCK falling edge	3.8	-	10	ns
t_{DDA}	I2S_DAT_OUT propagation delay from BCK falling edge	2.2	-	10	ns
t_{DST}	I2S_DAT_IN setup time to BCK rising edge	9	-	-	ns
t_{DHT}	I2S_DAT_IN hold time from BCK rising edge	5	-	-	ns

2.5 I2C Timing Characteristics

The I2C timing sequence in fast mode is shown as below.

Figure 2-2 I2C Timing in Fast Mode



The I2C timing characteristics is listed as below.

Table 2-13 I2C Timing Sequence

Symbol	Parameter	Conditions	Standard mode		Fast mode		Unit
			Min.	Max.	Min.	Max.	
V_{IL}	LOW level input voltage	-	-0.5	0.3VDD	-0.5	0.3VDD	V
V_{IH}	HIGH level input voltage	-	0.7VDD	VDD+0.5	0.7VDD	VDD+0.5	V
t_{SP}	Pulse width of spikes that must be suppressed by input filter	-	-	-	0	50[a]	ns
f_{SCL}	SCL clock frequency	-	0	100	0	400	kHz
t_{LOW}	LOW period of the SCL clock	-	4.7	-	1.3	-	μs
t_{HIGH}	HIGH period of the SCL clock	-	4.0	-	0.6	-	μs
t_r	Rise time for both SDA and SCL	-	-	1000	20	300	ns
t_f	Fall time for both SDA and SCL	-	-	300	20xVDD	300	ns
$t_{HD;STA}$	Hold time for a repeated START condition	After this period, the first clock pulse is generated.	4.0	-	0.6	-	μs

Symbol	Parameter	Conditions	Standard mode		Fast mode		Unit
			Min.	Max.	Min.	Max.	
$t_{SU;STA}$	Set-up time for a repeated START condition	-	4.7	-	0.6	-	μs
$t_{HD;DATI}$	Input data hold time	I2C bus devices	0	-	0	-	μs
$t_{HD;DATO}$	Output data hold time	I2C bus devices	-	3.45	-	0.9	μs
$t_{SU;DAT}$	Data set-up time	-	250	-	100	-	ns
$t_{SU;STO}$	Set-up time for STOP condition	-	4.0	-	0.6	-	μs
t_{BUF}	Bus free time between a STOP and START condition	-	4.7	-	1.3	-	μs

a. Input filters on the SDA and SCL inputs suppress noise spikes of less than 50 ns.

2.6 Storage Conditions

The SoC series is applicable to Moisture Sensitivity Level 3 (based on JEDEC Standard).

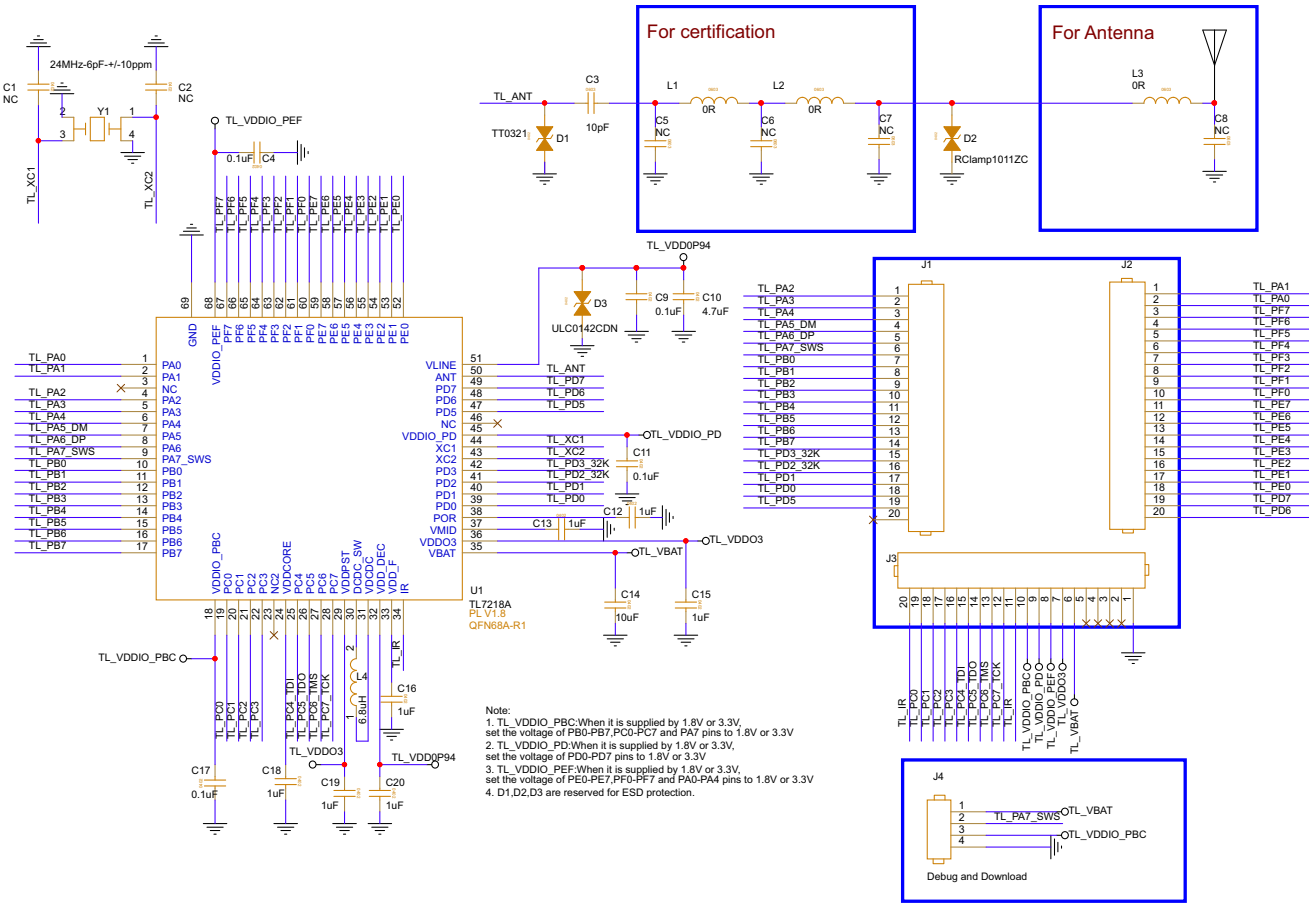
1. Calculated shelf life in sealed moisture barrier bag (MBB): 12 months at <40°C and <90% relative humidity (RH)
2. Peak package body temperature: 260°C
3. After bag is opened, devices that is subjected to reflow solder or other high temperature process must be
   - Mounted within: 168 hours of factory conditions <=30°C/60% RH, or
   - Stored at <10% RH
4. Devices require bake before mounting, if:
   - Humidity Indicator Card reads >10% when read at 23 ± 5°C
   - Both of the conditions in 3 are not met
5. If baking is required, devices may be baked for 24 hours at 125 ±5°C

Note: If device containers cannot be subjected to high temperature or shorter bake times are desired, please refer to IPC/JEDEC J-STD-033 for bake condition.

3 Reference Design

3.1 Reference Schematic of TL7218A

Figure 3-1 Reference Schematic of TL7218A



3.2 BOM (Bill of Material) for TL7218A

The bill of material table for TL7218A reference design is listed as below.

Table 3-1 BOM Table for TL7218A Reference Design

Quantity	Reference	Value	PCB Footprint	Description
2	C1, C2	NC	0402	CAP CER 16V X5R,±10%
1	C3	10pF	0402	CAP CER 16V X5R,±10%
4	C4,C9,C11,C17	0.1uF	0402	CAP CER 16V X5R,±10%
4	C5,C6,C7,C8	NC	0402	CAP expected, spec required for "CER 50V ±0.1pF C0G"
1	C10	4.7uF	0402	CAP CER 16V X5R,±10%

Quantity	Reference	Value	PCB Footprint	Description
7	C12,C13,C15,C16,C18,C19,C20	1uF	0402	CAP CER 16V X5R,±10%
1	C14(1)	10uF	0603	CAP CER 16V X5R,±10%
1	D1(2)	TT0321	0402	Air/Contact discharge: ±8kV/±15kV Vr=3.3V, Cj=0.3pF, Vc=6V(Ipp=5A)
1	D2	RClamp1011ZC	0201	Air/Contact discharge: ±10kV/±15kV Vr=1V, Cj=0.2pF, Vc=2.77V(Ipp=2.5A)
1	D3	ULC0142CDN	0402	Air/Contact discharge: ±20kV/±22kV Vr=1V, Cj=0.65pF, Vc=6V(Ipp=6A)
3	J1,J2,J3	header	1x20	1x20 header
1	J4	header	1x4	1x4 header for debug and download
3	L1,L2,L3	0R	0402	Ind expected, spec required for "100MHz 850mA 12.5% 0.11HMS"
1	L4(3)	6.8uH	1008L	IND CHK 1MHz 1.2A 20% DCR 0.33 Ref. MPN: DFE252012F-6R8M=P2
1	U1	TL7218A	QFN68	IC MCU RISC-V FLASH 2MB SRAM 512KB 1.8-4.3V
1	Y1	24MHz-6pF-+/-10ppm	OSCCC250X320X110	XTAL SMD 3225, 24 MHz, CI=6pF, total tol.±10ppm Ref. MPN: E3SB24E001D00E

NOTE: *(1) For the capacitance for the VBAT pin, it is 10uF by default. When using VBAT above 3.0V for voltage sampling in LDO power mode, it is recommend to place a 22uF capacitor for the VBAT pin to avoid sampling error.*

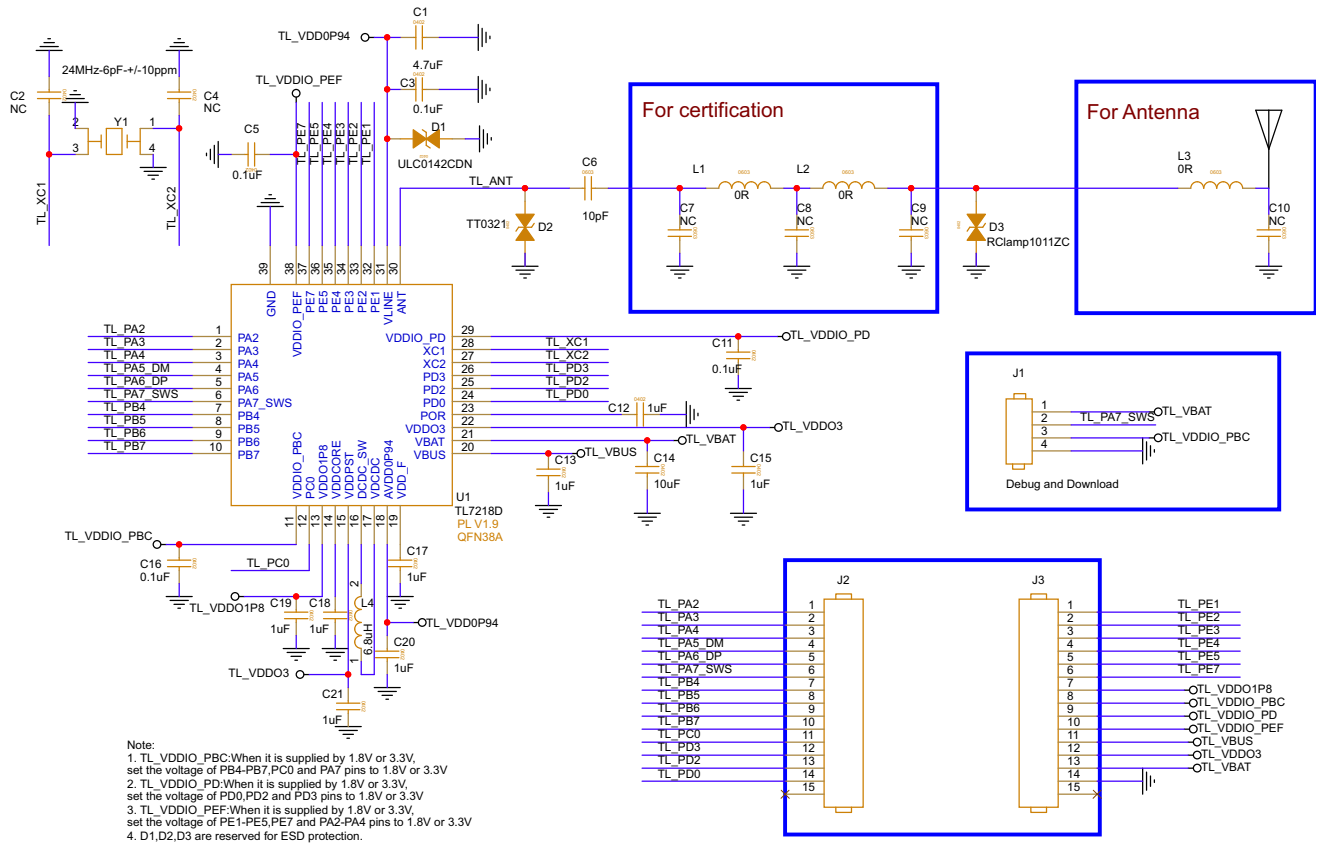
NOTE: *(2) For the ESD protection of the RF circuit, it is recommended to add a band pass filter if higher ESD performance is required.*

NOTE: *(3) For the 6.8uH inductor selection, it is recommended to*

- use wire wound type instead of multilayer type because wire wound inductors have higher quality and better temperature stability;
- select lower DCR (Direct Current Resistance), less than 0.5 Ohm;
- select larger rated current, at least 500 mA;
- select higher Q value (quality factor), at least 25.

3.3 Reference Schematic of TL7218D

Figure 3-2 Reference Schematic of TL7218D



3.4 BOM (Bill of Material) for TL7218D

The bill of material table for TL7218D reference design is listed as below.

Table 3-2 BOM Table for TL7218D Reference Design

Quantity	Reference	Value	PCB Footprint	Description
1	C1	4.7uF	0402	CAP CER 16V X5R,±10%
2	C2,C4	NC	0402	CAP CER 16V X5R,±10%
4	C7,C8,C9,C10	NC	0402	CAP expected, spec required for "CER 50V ±0.1pF C0G"
4	C3,C5,C11,C16	0.1uF	0402	CAP CER 16V X5R,±10%
1	C6	10pF	0402	CAP CER 16V X5R,±10%
8	C12,C13,C15,C17,C18, C19,C20,C21	1uF	0402	CAP CER 16V X5R,±10%
1	C14(1)	10uF	0603	CAP CER 16V X5R,±10%

Quantity	Reference	Value	PCB Footprint	Description
1	D1(2)	ULC0142CDN	0402	Air/Contact discharge: ±20kV/±22kV Vr=1V, Cj=0.65pF, Vc=6V(Ipp=6A)
1	D2	TT0321	0402	Air/Contact discharge: ±8kV/±15kV Vr=3.3V, Cj=0.3pF, Vc=6V(Ipp=5A)
1	D3	RClamp1011ZC	0201	Air/Contact discharge: ±10kV/±15kV Vr=1V, Cj=0.2pF, Vc=2.77V(Ipp=2.5A)
1	J1	header	1x4	1x4 header for debug and download
2	J2,J3	header	1x15	1x15 header
3	L1,L2,L3	0R	0402	Ind expected, spec required for "100MHz 850mA 12.5% 0.11HMS"
1	L4(3)	6.8uH	1008L	IND CHK 1MHz 1.2A 20% DCR 0.33 Ref. MPN: DFE252012F-6R8M=P2
1	U1	TL7218D	QFN38	IC MCU RISC-V FLASH 2MB SRAM 512KB 1.8-4.3V
1	Y1	24MHz-6pF-+/-10ppm	OSCCC250X320X110	XTAL SMD 3225, 24 MHz, CI=6pF, total tol.±10ppm Ref. MPN: E3SB24E001D00E

NOTE: *(1) For the capacitance for the VBAT pin, it is 10uF by default. When using VBAT above 3.0V for voltage sampling in LDO power mode, it is recommend to place a 22uF capacitor for the VBAT pin to avoid sampling error.*

NOTE: *(2) For the ESD protection of the RF circuit, it is recommended to add a band pass filter if higher ESD performance is required.*

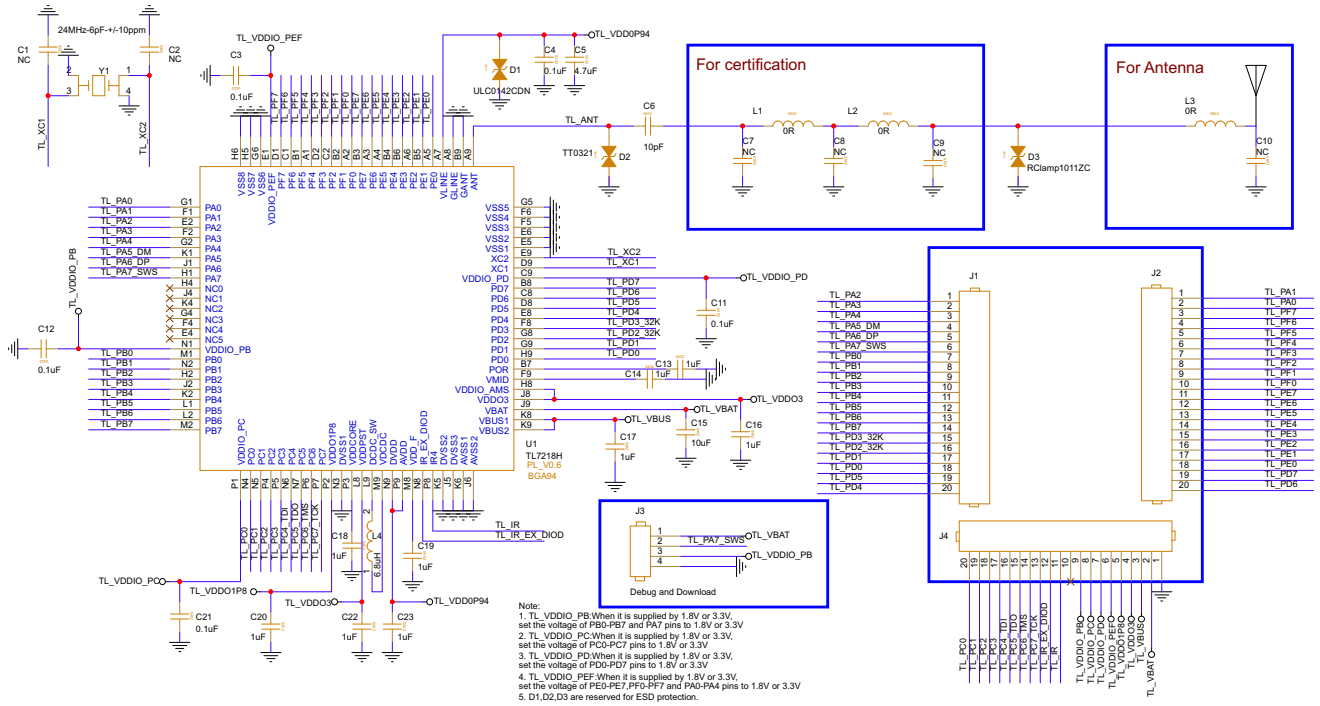
NOTE: *(3) For the 6.8uH inductor selection, it is recommended to*

- use wire wound type instead of multilayer type because wire wound inductors have higher quality and better temperature stability;
- select lower DCR (Direct Current Resistance), less than 0.5 Ohm;
- select larger rated current, at least 500 mA;
- select higher Q value (quality factor), at least 25.

3.5 Reference Schematic of TL7218H

The reference schematic of TL7218H is shown as below.

Figure 3-3 Reference Schematic of TL7218H



3.6 BOM (Bill of Material) for TL7218H

The bill of material table for TL7218H reference design is listed as below.

Table 3-3 BOM Table for TL7218H Reference Design

Quantity	Reference	Value	PCB Footprint	Description
2	C1,C2	NC	0402	CAP CER 16V X5R,±10%
4	C7,C8,C9,C10	NC	0402	CAP expected, spec required for "CER 50V ±0.1pF C0G"
5	C3,C4,C11,C12,C21	0.1uF	0402	CAP CER 16V X5R,±10%
1	C5	4.7uF	0402	CAP CER 16V X5R,±10%
1	C6	10pF	0402	CAP CER 16V X5R,±10%
9	C13,C14,C16,C17,C18,C19, C20,C22,C23	1uF	0402	CAP CER 16V X5R,±10%
1	C15(1)	10uF	0603	CAP CER 16V X5R,±10%
1	D1(2)	ULC0142CDN	0402	Air/Contact discharge: ±20kV/±22kV Vr=1V, Cj=0.65pF, Vc=6V(Ipp=6A)

Quantity	Reference	Value	PCB Footprint	Description
1	D2	TT0321	0402	Air/Contact discharge: ±8kV/±15kV Vr=3.3V, Cj=0.3pF, Vc=6V(Ipp=5A)
1	D3	RClamp1011ZC	0201	Air/Contact discharge: ±10kV/±15kV Vr=1V, Cj=0.2pF, Vc=2.77V(Ipp=2.5A)
3	J1,J2,J4	header	1x20	1x20 header
1	J3	header	1x4	1x4 header for debug and download
3	L1,L2,L3	0R	0402	Ind expected, spec required for "100MHz 850mA 12.5% 0.11HMS"
1	L4(3)	6.8uH	1008L	IND CHK 1MHz 1.2A 20% DCR 0.33 Ref. MPN: DFE252012F-6R8M=P2
1	U1	TL7218H	BGA94	IC MCU RISC-V FLASH 2MB SRAM 512KB 1.8-4.3V
1	Y1	24MHz-6pF-+/-10ppm	OSCCC250X320X110	XTAL SMD 3225, 24 MHz, CI=6pF, total tol.±10ppm Ref. MPN: E3SB24E001D00E

NOTE: *(1) For the capacitance for the VBAT pin, it is 10uF by default. When using VBAT above 3.0V for voltage sampling in LDO power mode, it is recommend to place a 22uF capacitor for the VBAT pin to avoid sampling error.*

NOTE: *(2) For the ESD protection of the RF circuit, it is recommended to add a band pass filter if higher ESD performance is required.*

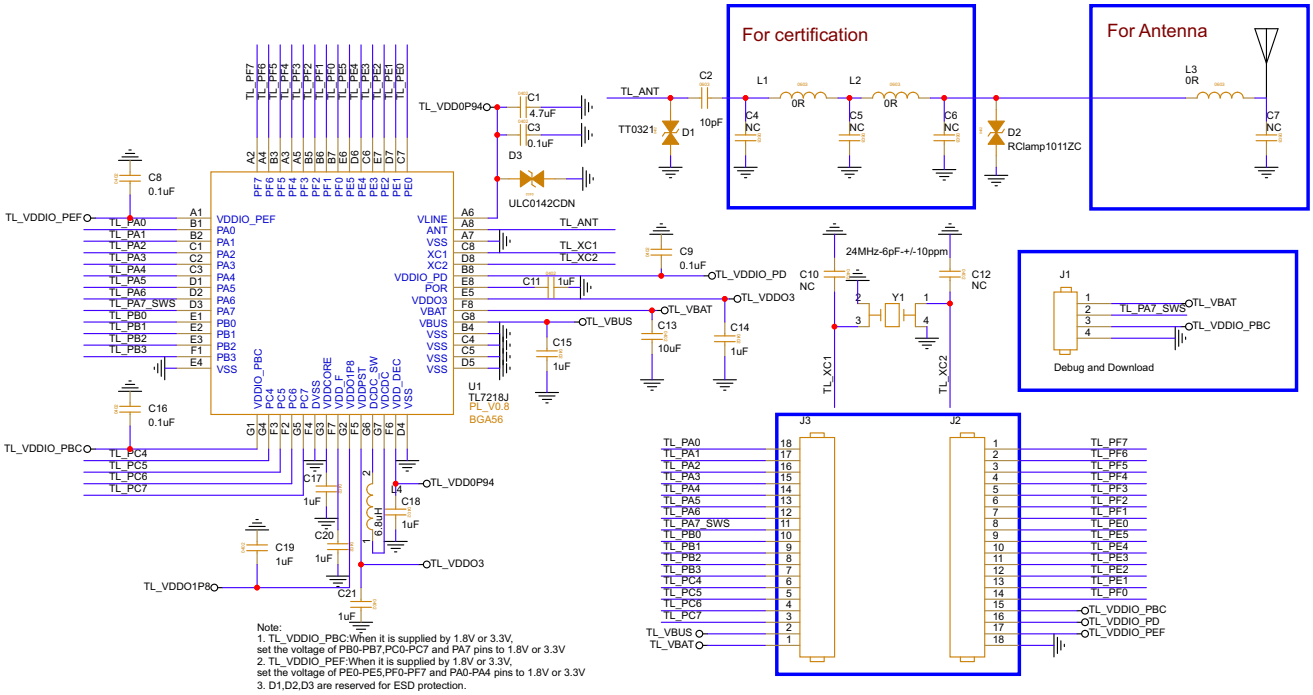
NOTE: *(3) For the 6.8uH inductor selection, it is recommended to*

- use wire wound type instead of multilayer type because wire wound inductors have higher quality and better temperature stability;
- select lower DCR (Direct Current Resistance), less than 0.5 Ohm;
- select larger rated current, at least 500 mA;
- select higher Q value (quality factor), at least 25.

3.7 Reference Schematic of TL7218J

The reference schematic of TL7218J is shown as below.

Figure 3-4 Reference Schematic of TL7218J



3.8 BOM (Bill of Material) for TL7218J

Table 3-4 BOM Table for TL7218J Reference Design

Quantity	Reference	Value	PCB Footprint	Description
1	C1	4.7uF	0402	CAP CER 16V X5R,±10%
1	C2	10pF	0402	CAP CER 16V X5R,±10%
4	C3,C8,C9,C16	0.1uF	0402	CAP CER 16V X5R,±10%
2	C10,C12	NC	0402	CAP CER 16V X5R,±10%
4	C4,C5,C6,C7	NC	0402	CAP expected, spec required for "CER 50V ±0.1pF C0G"
8	C11,C14,C15,C17,C18,C19, C20,C21	1uF	0402	CAP CER 16V X5R,±10%
1	C13(1)	10uF	0603	CAP CER 16V X5R,±10%
1	D1(2)	TT0321	0402	Air/Contact discharge: ±8kV/±15kV Vr=3.3V, Cj=0.3pF, Vc=6V(Ipp=5A)

Quantity	Reference	Value	PCB Footprint	Description
1	D2	RClamp1011ZC	0201	Air/Contact discharge: ±10kV/±15kV Vr=1V, Cj=0.2pF, Vc=2.77V(Ipp=2.5A)
1	D3	ULC0142CDN	0402	Air/Contact discharge: ±20kV/±22kV Vr=1V, Cj=0.65pF, Vc=6V(Ipp=6A)
1	J1	header	1x4	1x4 header for debug and download
2	J2,J3	header	1x18	1x18 header
3	L1,L2,L3	0R	0402	Ind expected, spec required for "100MHz 850mA 12.5% 0.11HMS"
1	L4(3)	6.8uH	1008L	IND CHK 1MHz 1.2A 20% DCR 0.33 Ref. MPN: DFE252012F-6R8M=P2
1	U1	TL7218J	BGA56	IC MCU RISC-V FLASH 2MB SRAM 512KB 1.8-4.3V
1	Y1	24MHz-6pF-+/-10ppm	OSCCC250X320X110	XTAL SMD 3225, 24 MHz, CI=6pF, total tol.±10ppm Ref. MPN: E3SB24E001D00E

NOTE: *(1) For the capacitance for the VBAT pin, it is 10uF by default. When using VBAT above 3.0V for voltage sampling in LDO power mode, it is recommend to place a 22uF capacitor for the VBAT pin to avoid sampling error.*

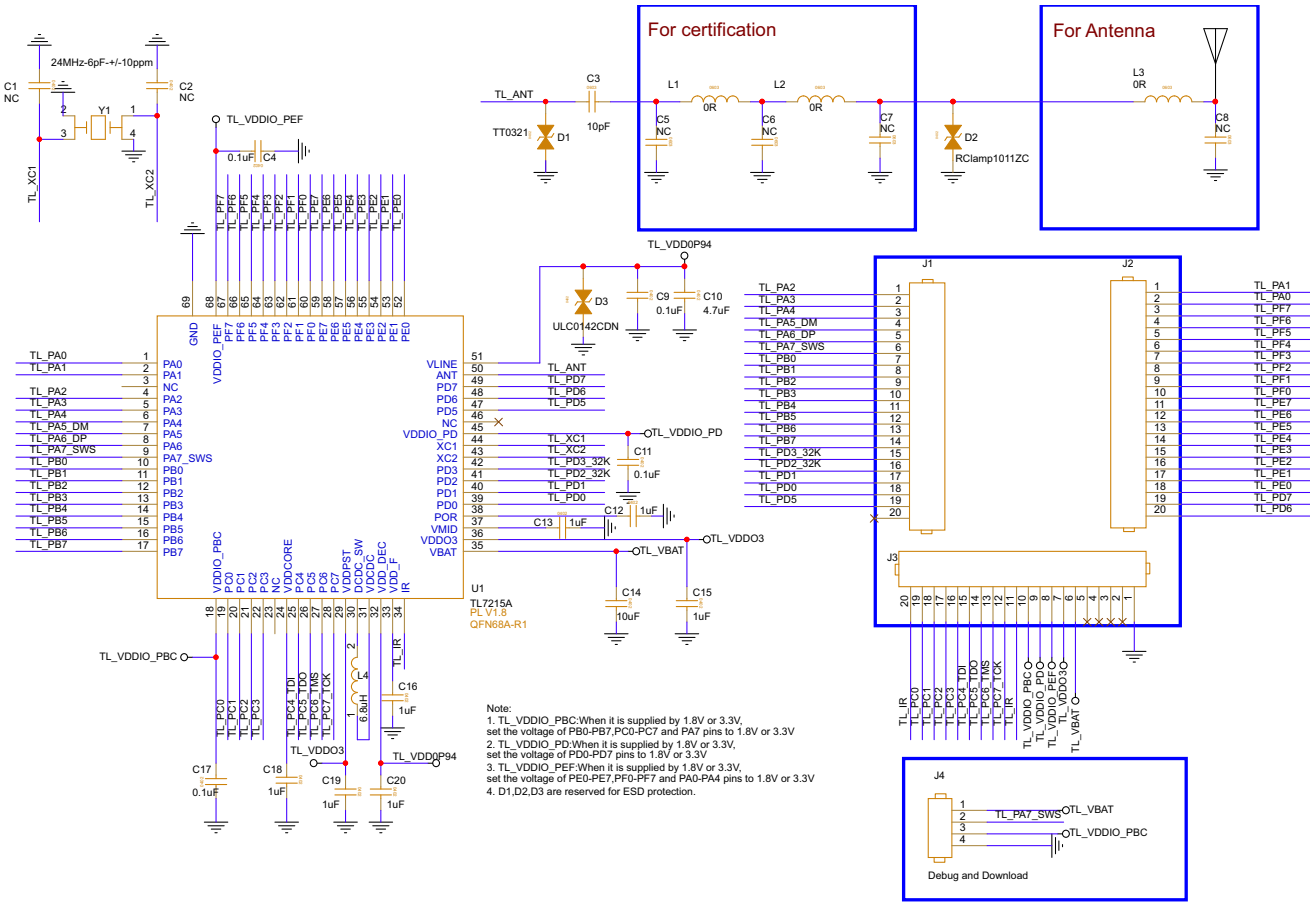
NOTE: *(2) For the ESD protection of the RF circuit, it is recommended to add a band pass filter if higher ESD performance is required.*

NOTE: *(3) For the 6.8uH inductor selection, it is recommended to*

- use wire wound type instead of multilayer type because wire wound inductors have higher quality and better temperature stability;
- select lower DCR (Direct Current Resistance), less than 0.5 Ohm;
- select larger rated current, at least 500 mA;
- select higher Q value (quality factor), at least 25.

3.9 Reference Schematic of TL7215A

Figure 3-5 Reference Schematic of TL7215A



3.10 BOM (Bill of Material) for TL7215A

The bill of material table for TL7215A reference design is listed as below.

Table 3-5 BOM Table for TL7215A Reference Design

Quantity	Reference	Value	PCB Footprint	Description
2	C1, C2	NC	0402	CAP CER 16V X5R,±10%
1	C3	10pF	0402	CAP CER 16V X5R,±10%
4	C4,C9,C11,C17	0.1uF	0402	CAP CER 16V X5R,±10%
4	C5,C6,C7,C8	NC	0402	CAP expected, spec required for "CER 50V ±0.1pF C0G"
1	C10	4.7uF	0402	CAP CER 16V X5R,±10%
7	C12,C13,C15,C16,C18,C19,C20	1uF	0402	CAP CER 16V X5R,±10%
1	C14(1)	10uF	0603	CAP CER 16V X5R,±10%

Quantity	Reference	Value	PCB Footprint	Description
1	D1(2)	TT0321	0402	Air/Contact discharge: ±8kV/±15kV Vr=3.3V, Cj=0.3pF, Vc=6V(Ipp=5A)
1	D2	RClamp1011ZC	0201	Air/Contact discharge: ±10kV/±15kV Vr=1V, Cj=0.2pF, Vc=2.77V(Ipp=2.5A)
1	D3	ULC0142CDN	0402	Air/Contact discharge: ±20kV/±22kV Vr=1V, Cj=0.65pF, Vc=6V(Ipp=6A)
3	J1,J2,J3	header	1x20	1x20 header
1	J4	header	1x4	1x4 header for debug and download
3	L1,L2,L3	0R	0402	Ind expected, spec required for "100MHz 850mA 12.5% 0.11HMS"
1	L4(3)	6.8uH	1008L	IND CHK 1MHz 1.2A 20% DCR 0.33 Ref. MPN: DFE252012F-6R8M=P2
1	U1	TL7215A	QFN68	IC MCU RISC-V FLASH 1MB SRAM 256KB 1.8-4.3V
1	Y1	24MHz-6pF-+/-10ppm	OSCCC250X320X110	XTAL SMD 3225, 24 MHz, CI=6pF, total tol.±10ppm Ref. MPN: E3SB24E001D00E

NOTE: *(1) For the capacitance for the VBAT pin, it is 10uF by default. When using VBAT above 3.0V for voltage sampling in LDO power mode, it is recommend to place a 22uF capacitor for the VBAT pin to avoid sampling error.*

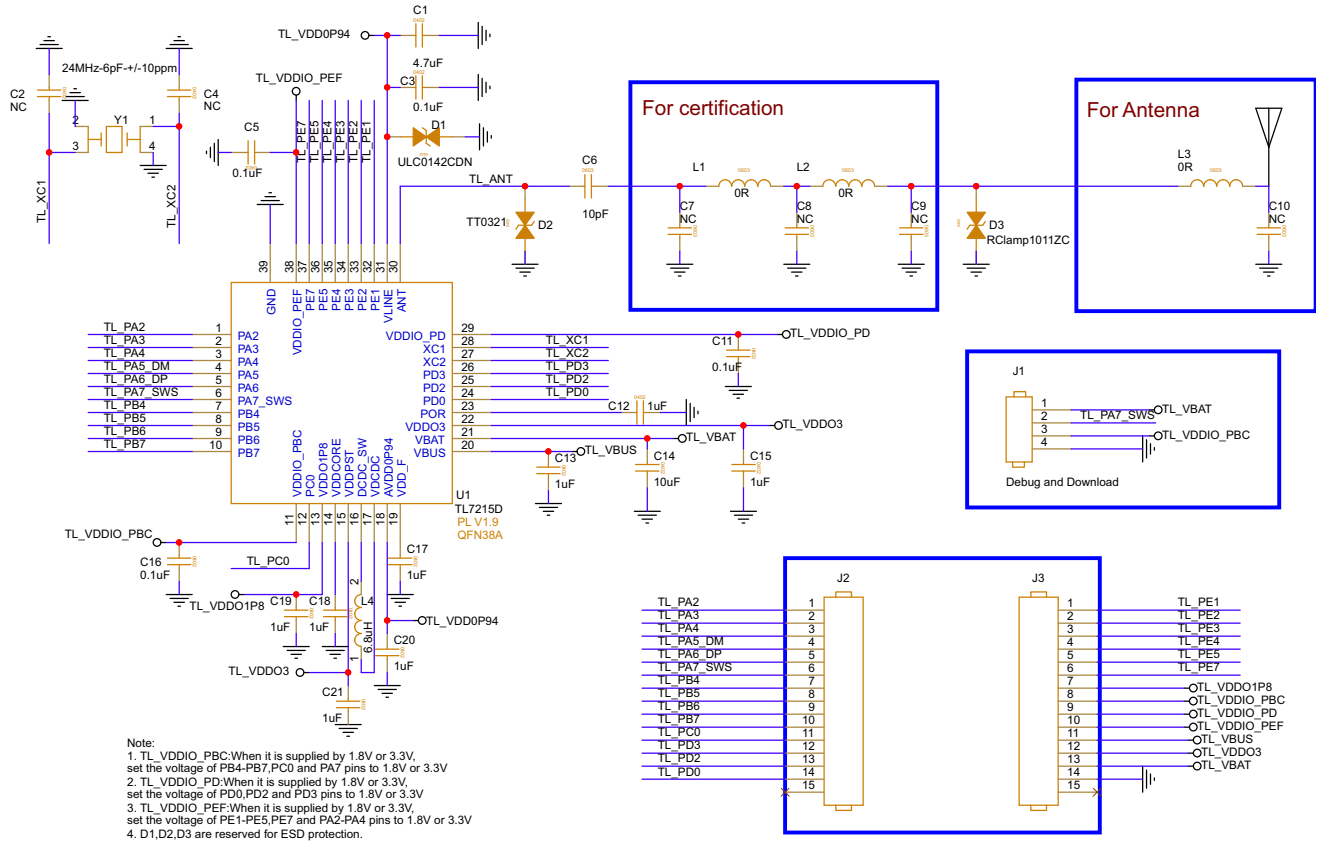
NOTE: *(2) For the ESD protection of the RF circuit, it is recommended to add a band pass filter if higher ESD performance is required.*

NOTE: *(3) For the 6.8uH inductor selection, it is recommended to*

- use wire wound type instead of multilayer type because wire wound inductors have higher quality and better temperature stability;
- select lower DCR (Direct Current Resistance), less than 0.5 Ohm;
- select larger rated current, at least 500 mA;
- select higher Q value (quality factor), at least 25.

3.11 Reference Schematic of TL7215D

Figure 3-6 Reference Schematic of TL7215D



3.12 BOM (Bill of Material) for TL7215D

The bill of material table for TL7215D reference design is listed as below.

Table 3-6 BOM Table for TL7215D Reference Design

Quantity	Reference	Value	PCB Footprint	Description
1	C1	4.7uF	0402	CAP CER 16V X5R,±10%
2	C2,C4	NC	0402	CAP CER 16V X5R,±10%
4	C7,C8,C9,C10	NC	0402	CAP expected, spec required for "CER 50V ±0.1pF C0G"
4	C3,C5,C11,C16	0.1uF	0402	CAP CER 16V X5R,±10%
1	C6	10pF	0402	CAP CER 16V X5R,±10%
8	C12,C13,C15,C17,C18,C19, C20,C21	1uF	0402	CAP CER 16V X5R,±10%
1	C14(1)	10uF	0603	CAP CER 16V X5R,±10%

Quantity	Reference	Value	PCB Footprint	Description
1	D1[2]	ULC0142CDN	0402	Air/Contact discharge: ±20kV/±22kV Vr=1V, Cj=0.65pF, Vc=6V(Ipp=6A)
1	D2	TT0321	0402	Air/Contact discharge: ±8kV/±15kV Vr=3.3V, Cj=0.3pF, Vc=6V(Ipp=5A)
1	D3	RClamp1011ZC	0201	Air/Contact discharge: ±10kV/±15kV Vr=1V, Cj=0.2pF, Vc=2.77V(Ipp=2.5A)
1	J1	header	1x4	1x4 header for debug and download
2	J2,J3	header	1x15	1x15 header
3	L1,L2,L3	0R	0402	Ind expected, spec required for "100MHz 850mA 12.5% 0.11HMS"
1	L4[3]	6.8uH	1008L	IND CHK 1MHz 1.2A 20% DCR 0.33 Ref. MPN: DFE252012F-6R8M=P2
1	U1	TL7215D	QFN38	IC MCU RISC-V FLASH 1MB SRAM 256KB 1.8-4.3V
1	Y1	24MHz-6pF-+/-10ppm	OSCCC250X320X110	XTAL SMD 3225, 24 MHz, CI=6pF, total tol.±10ppm Ref. MPN: E3SB24E001D00E

NOTE: *(1) For the capacitance for the VBAT pin, it is 10uF by default. When using VBAT above 3.0V for voltage sampling in LDO power mode, it is recommend to place a 22uF capacitor for the VBAT pin to avoid sampling error.*

NOTE: *(2) For the ESD protection of the RF circuit, it is recommended to add a band pass filter if higher ESD performance is required.*

NOTE: *(3) For the 6.8uH inductor selection, it is recommended to*

- use wire wound type instead of multilayer type because wire wound inductors have higher quality and better temperature stability;
- select lower DCR (Direct Current Resistance), less than 0.5 Ohm;
- select larger rated current, at least 500 mA;
- select higher Q value (quality factor), at least 25.

4 Memory, MCU and PMU

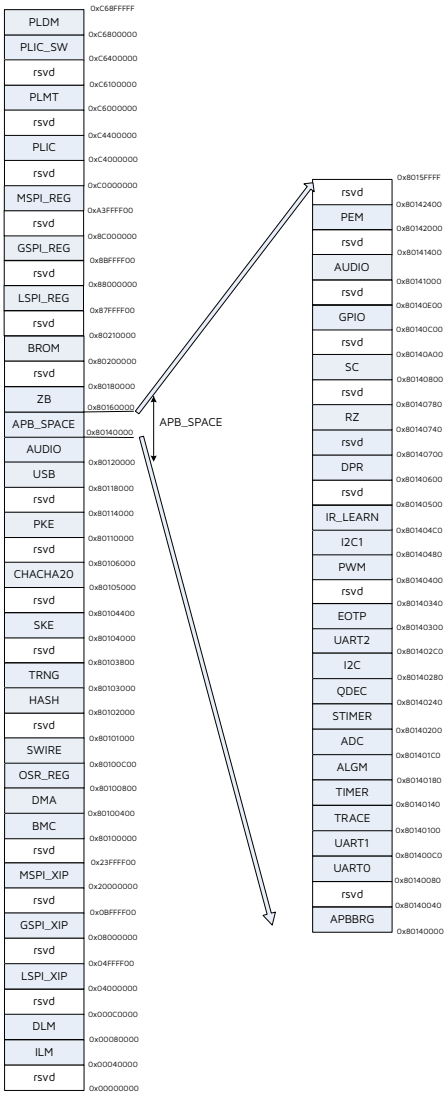
4.1 Memory

The SoC embeds 512KB SRAM (256 KB instruction memory and 256 KB data memory, in which 256 KB of instruction memory can be retention in deep sleep), or 256 KB SRAM (256 KB instruction, 0 KB data memory, all 256 KB can be retention in deep sleep), and external flash (the flash size varies on the sepcified parts, see section 1.4) as program memory.

4.1.1 SRAM

The memory map is shown below. As shown in the figure, the SoC embedded 2 SRAM, 256 KB instruction local memory (ILM) and 256 KB data local memory (DLM) for 512 KB SRAM chip. For ILM, the lower 256 KB is with retention in deep sleep. The 256 KB SRAM chip consists 256 KB instruction local memory (ILM) , all the 256 KB ILM can be used in retention in deep sleep mode. Please be noted, ILM can store both instruction and data while DLM can only store data, so the instruction stored in local memory should be less than 256 KB.

Figure 4-1 Memory Map



4.1.2 Flash

The internal flash mainly supports page program, sector/block/chip erase operations, and deep power down operation. Please refer to the corresponding SDK for flash memory operation details.

Please note that for 1MB embedded flash, the flash area ranging from 0xF8000 to 0xFFFFF is reserved for Telink internal use; for 2MB embedded flash, the flash area ranging from 0x1F0000 to 0x1FFFFF is reserved for Telink internal use.

The MCU uses the separate MSPI_CLK frequency to load instructions, and adopts flash driver to access (read/ write) flash with the same speed.

NOTE:

- *By default, the page write is 256 bytes at a time and it is not recommended to write less than 255 bytes data. For example, if users want to rewrite 64 to 128 bytes in one page, the first step is to read total 256 bytes of a page, then replace the 64 to 128 bytes data, and rewrite the corrective 256 bytes to program again after page erase.*

4.1.3 OTP

The chip embeds a 256 x 32 bits OTP device with the following features:

- 1-bit program operation
- Build in 1 bit charge pump
- Build in ECC scheme
- Data retention: > 10 years
- Bit program time: 30 μs (min) for temperature above 0°C and 40 μs (min) for temperature below 0°C

The OTP section is preloaded with OTP configuration shown as below.

Table 4-1 OTP Definition

Name	Length (Byte)	Description
MAC_ADDR	8	64 bits Telink-Assigned MAC Address
Reserved	4	Reserved
otp_boot_config3	4	[31:1] Reserved [0] fast_boot_enable, 1: enable, 0: disable
otp_boot_config1	4	[31:1] Reserved [0] mode selection, 0: secure boot mode, 1: normal mode
otp_boot_config2	4	[31:21] Reserved [20] enable_deep_flash_wkup_time [19:12] Flash_pwup_time_set: Flash_pwup_time = (255 - Flash_pwup_time_set) * 50 us [11:0] descriptor flash address = {[11:0], 12'h0};

Name	Length (Byte)	Description
pub_key_hash	32	Hash public key
chip_id	16	Chip ID, unmodifiable
debug_text	16	Plaintext for debugging
root_key	16	Root key
Interface functions	12	These bits are used for function configuration on die level by HW or SW. [95:65] Reserved [64] flash_encrypt_disable, 1'b0: flash encryption enable 1'b1: flash encryption disable [63:37] Reserved [36] otp_key_enable, 1'b0: otp key can't be read 1'b1: otp key can be read [35] sspi_dbg_enable, SPI slave function enable: 1'b0: disable; 1'b1: enable. [34] sws_dbg_enable, SWS function enable: 1'b0: disable; 1'b1: enable. [33] jtag_dbg_enable, JTAG function enable: 1'b0: disable; 1'b1: enable. [32] usb_dbg_enable, USB debug function enable: 1'b0: disable; 1'b1: enable. [31:0] Reserved

4.1.4 Unique ID

For chip identification and traceability, the flash is preloaded with 128-bit Unique ID (UID). This UID can be read via the interface in SDK.

4.2 MCU

The SoC embeds a 32-bit RISC-V micro-controller, features are listed as following:

1. 5-stage in-order execution pipeline
2. Fast Hardware multiplier
3. Hardware divider
4. Dynamic branch prediction
   - 128-entry branch target buffer (BTB)
5. Performance monitors
6. Misaligned memory accesses
7. RISC-V RV32I base integer instruction set
8. RISC-V RVC standard extension for compressed instructions
9. RISC-V RVM standard extension for integer multiplication and division
10. RISC-V RVA standard extension for atomic instructions
11. RISC-V "F" standard extensions for single-precision floating-point
12. DSP extension
13. I & D caches
14. I & D local memories
15. Machine mode and User mode
16. 8 entries PMP (Physical Memory Protection)

4.2.1 Physical Memory Protection

To support secure processing and contain faults, it is desirable to limit the physical addresses accessible by software running on a hart. Physical memory protection (PMP) unit provides hart machine-mode control registers to allow physical memory access privileges (read, write, execute) to be specified for each physical memory region.

PMP checks are applied to all accesses when the hart is running in U modes, and for loads and stores when the MPRV bit is set in the m-status register and the MPP field in the m-status register contains U. Optionally, PMP checks may additionally apply to M-mode accesses, in which case the PMP registers themselves are locked, so that even M-mode software cannot change them without a system reset. PMP violations are always trapped precisely at the processor.

PMP entries are described by an 8-bit configuration register and one 32 bit address register.

8 PMP entries are supported. PMP CSRs are only accessible to M-mode.

NOTE: *The abbreviations for the Type column of register table are summarized below:*

- *RO: read only*
- *WO: write only*
- *R/W: readable and writable*
- *W1C: write 1 to clear*
- *W1S: write 1 to set*
- *Volatile: can be modified unexpectedly*

Table 4-2 PMP Configuration Registers

CSR Address	CSR Name	Bit	Description
0x3A0	pmpcfg0	[31:24]	PMP3CFG
		[23:16]	PMP2CFG
		[15:8]	PMP1CFG
		[7:0]	PMP0CFG
0x3A1	pmpcfg1	[31:24]	PMP7CFG
		[23:16]	PMP6CFG
		[15:8]	PMP5CFG
		[7:0]	PMP4CFG

The 8-bit PMPiCFG is described as below.

Table 4-3 PMPiCFG Description

Field Name	Bit	Description	Type	Reset
L	[7]	Write lock and permission enforcement bit for Machine mode. 0: Machine mode writes to PMP entry registers are allowed. R/W/X permissions apply to U modes 1: For PMP entry i, writes to PMPiCFG and PMPADDRi are ignored. Additionally, if PMPiCFG.A is set to TOR, writes to pmpaddri-1 are ignored as well. As for Permission enforcement, R/W/X permissions apply to all modes. This bit can only be cleared to 0 with a system reset	W1S	0
Reserved	[6:5]	Reserved	-	-
A	[4:3]	Address matching mode. 0: OFF: Null region. 1: TOR: Top of range. For PMP entry 0, it matches any address A < pmpaddr0. For PMP entry i, it matches any address A such that pmpaddri > A >=pmpaddri-1. But the 4-byte range is not supported. 2: Reserved. 3: NAPOT: Naturally aligned power-of-2 region, >= 8 bytes. This mode makes use of the low-order bits of the associated address register to encode the size of the range. See Table 4-5 for range encoding from the value of a PMP address register.	RW	0

Field Name	Bit	Description	Type	Reset
X	[2]	Instruction execution control. 0: Instruction execution is not allowed. 1: Instruction execution is allowed	RW	0
W	[1]	Write access control. 0: Write accesses are not allowed. 1: Write accesses are allowed.	RW	0
R	[0]	Read access control. 0:Read accesses are not allowed 1:Read accesses are allowed.	RW	0

Table 4-4 PMP Address Registers

CSR Address	CSR Name	Bit	Description	Type	Reset
0x3B0	pmpaddr0	[31:0]	PMP entry 0 address register	RW	0
0x3B1	pmpaddr1	[31:0]	PMP entry 1 address register	RW	0
0x3B2	pmpaddr2	[31:0]	PMP entry 2 address register	RW	0
0x3B3	pmpaddr3	[31:0]	PMP entry 3 address register	RW	0
0x3B4	pmpaddr4	[31:0]	PMP entry 4 address register	RW	0
0x3B5	pmpaddr5	[31:0]	PMP entry 5 address register	RW	0
0x3B6	pmpaddr6	[31:0]	PMP entry 6 address register	RW	0
0x3B7	pmpaddr7	[31:0]	PMP entry 7 address register	RW	0

Each PMP address register encodes bits 33–2 of a 34-bit physical address, as shown in the register format. The encoding is described in Table 4-5. The "a" in the table represents one bit address, with arbitrary values.

Table 4-5 D25 NAPOT Range Encoding in PMP Address and Configuration Registers

Register Content	Match Size (Byte)
aaaa...aaa0	8 (2^3)
aaaa...aa01	16 (2^4)
aaaa...a011	32 (2^5)
.....	.....
aa01...1111	2^{32}

Register Content	Match Size (Byte)
a011...1111	2^{33}
0111...1111	2^{34}
1111...1111	2^{35}

4.3 Working Mode

The SoC supports five working modes, including Active, Idle, Suspend, Deep Sleep with SRAM retention, Deep Sleep without SRAM retention.

- The Power Management (PM) module is always active in all working modes.
- For modules such as MCU, RF transceiver (Radio), and SRAM, the state depends on working mode, as shown below.

Table 4-6 Working Mode

Mode	Active	Idle	Suspend	Deep Sleep With SRAM Retention	Deep Sleep without SRAM Retention
MCU	active	stall	stall	off	off
Radio	available	available	stall/off	off	off
USB	available	available	stall/off	off	off
Audio	available	available	stall/off	off	off
Wakeup Time to Active Mode in LDO Mode	-	0 μs	100 μs	Shorter than Deep sleep without retention, almost same as Suspend	1 ms
Wakeup Time to Active Mode in DCDC Mode	-	-	-	-	-
Retention SRAMs (with retention in deep sleep)	full	full	full	full	off
Wakeup on RTC (32K Timer wakeup)	-	-	available	available	available
Wakeup on pin (IO wakeup)	-	-	available	available	available

Mode	Active	Idle	Suspend	Deep Sleep With SRAM Retention	Deep Sleep without SRAM Retention
Wakeup on interrupt	-	available	-	-	-
Wakeup on reset pin (POR)	-	available	available	available	available

NOTE:

- active: MCU is at working state.
- stall: In Idle and Suspend mode, MCU does not work, while its clock is still running.
- available for modules: It's selectable to be at working state, or stall/be powered down if it does not need to work.
- available for wakeup: Corresponding wakeup method is supported.
- full/off for SRAMs:
  - full: In Active, Idle mode, the retention SRAMs are powered on and work normally (can be accessed); in Deep sleep with SRAM retention and Suspend mode, the retention SRAMs are powered on, however, the contents of the retention SRAMs can be retained and cannot be accessed.
  - off: The retention SRAMs are powered down in Deep sleep without SRAM retention mode.

Analog registers (0x35 ~ 0x3c) as shown in below table are retained in deep sleep mode and can be used to store program state information across deep sleep cycles.

- Analog registers 0x3a~0x3c are non-volatile even when chip enters deep sleep or chip is reset by watchdog or software, i.e. the contents of these registers won't be changed by deep sleep or watchdog reset or chip software reset.
- Analog registers 0x35~0x39 are non-volatile in deep sleep, but are cleared by watchdog reset or chip software reset.
- After POR (Power-On-Reset), all registers are cleared to their default values, including these analog registers.

User can set flag in these analog registers correspondingly, so as to check the booting source by reading the flag.

Table 4-7 Retention Analog Registers in Deep Sleep

Address	Type	Description	Reset Value
afe_0x35	R/W	buffer clean at power_on/32K_watchdog/ watchdog/reboot	11111111
afe_0x36	R/W	buffer clean at power_on/32K_watchdog/ watchdog/reboot	00000000

Address	Type	Description	Reset Value
afe_0x37	R/W	buffer clean at power_on/32K_watchdog/ watchdog/reboot	00000000
afe_0x38	R/W	buffer clean at power_on/32K_watchdog/ watchdog/reboot	00000000
afe_0x39	R/W	buffer clean at power_on/32K_watchdog/ watchdog/reboot	00000000
afe_0x3a	R/W	buffer clean at power_on/32K_watchdog	00000000
afe_0x3b	R/W	buffer clean at power_on/32K_watchdog	00000000
afe_0x3c	R/W	buffer clean at power_on/32K_watchdog	11111111

4.4 Reset

The chip supports three types of reset methods, including POR (Power-On-Reset), watchdog reset and software reset.

1. POR: After power on, the whole chip is reset, and all registers are cleared to their default values.
2. Watchdog reset: A programmable watchdog is supported to monitor the system. If watchdog reset is triggered, registers except for the retention analog registers 0x3a~0x3c are cleared.
3. Software reset: It is also feasible to carry out software reset for the whole chip or some modules.
   - Setting address 0x2f[5] as 1b'1 is to reset the whole chip. Similar to watchdog reset, the retention analog registers 0x3a~0x3c are non-volatile, while other registers including 0x35~0x39 are cleared by chip software reset.
   - Addresses 0x20~0x23 and 0x40~0x43 serve to reset individual modules: if some bit is set to logic "0", the corresponding module is reset.

The base address of the following reset related registers is 0x80140800.

Table 4-8 Register Configuration for Software Reset

Address Offset	Name	Type	Description	Reset Value
0x20	RST0	R/W	[0]: LSPI, reset active low, 0 for reset, 1 for disable reset [1]: I2C [2]: UART0 [3]: USB [4]: PWM [5]: rsvd [6]: UART1 [7]: Swires	0x80

Address Offset	Name	Type	Description	Reset Value
0x21	RST1	R/W	[0]: rsvd [1]: System Timer [2]: DMA [3]: ALGM [4]: PKE [5]: rsvd [6]: GSPI [7]: SPISLV, spi slave	0x80
0x22	RST2	R/W	[0]: Timer [1]: Audio [2]: I2C1 [3]: MCU reset disable [4]: MCU reset enable, when this bit set 1, enable power on reset to reset mcu (reset all) [5]: LM [6]: TRNG [7]: DPR	0x38
0x23	RST3	R/W	[0]: rsvd [1]: TRACE [2]: BROM [3]: rsvd [4]: MSPI [5]: QDEC [6]: SARADC [7]: ALG, analog module reset	0x96
0x2f	PWDNEN	R/W	[0]: suspend enable (RW) [4]: ramcrc_clren_tgl (W) [5]: rst_all (act as watchdog reset) (VOLATILE) [7]: stall_en_trg (stall mcu trig) (W)	0x00
0x40	RST4	R/W	[0]: rsvd [1]: rsvd [2]: rsvd [3]: rsvd [4]: SKE [5]: HASH [6]: rsvd [7]: ZB	0x04

Address Offset	Name	Type	Description	Reset Value
0x41	RST5	R/W	[0]: rsvd [1]: UART2 [2]: rsvd [3]: rsvd [4]: IR_LEARN [5]: rsvd [6]: PEM [7]: CHACHA20	0x00
0x42	RST6	R/W	[0]: RZ [7:1]: rsvd	0x04
0x43	RST7	R/W	[7:0]: rsvd	0x04

4.5 Power Management

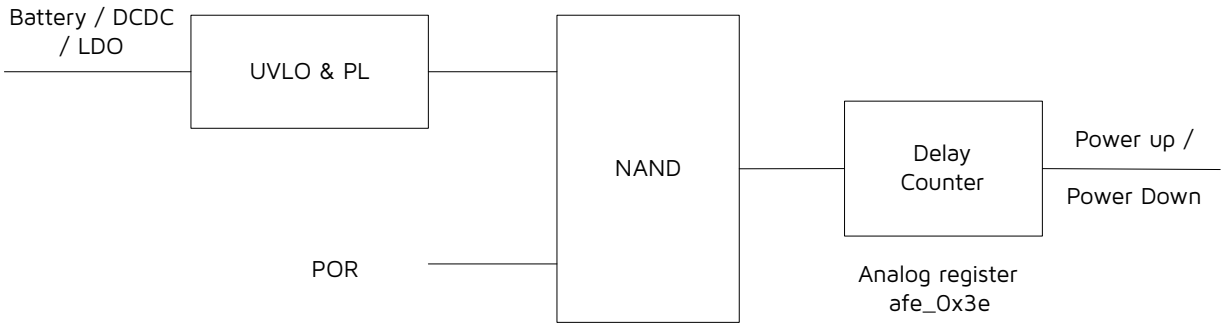
The multiple-stage Power Management (PM) module is flexible to control power state of the whole chip or individual functional blocks such as MCU, RF Transceiver, and peripherals by the following methods:

1. Power-On-Reset (POR) and Brown-out detect
2. Working Mode Switch
3. LDO and DCDC
4. VBAT and VANT Power-Supply Mode

4.5.1 Power-on-Reset (POR) and Brown-out Detect

Figure below shows the control logic of power up/down.

Figure 4-2 Control Logic of Power up/down



As shown in the above figure, the entire power-up and power-down process is controlled by the UVLO (Ultra-low Voltage Lockout) & PL (Power Logic) module, and the external POR pin, as illustrated in the diagram. The UVLO module takes the external power supply as input and only releases the lock when the power supply voltage exceeds a predetermined threshold.

After the UVLO and POR signals are released, there is an additional configurable delay before the system reset signal ("Sysrst") is released. This delay can be adjusted using the analog register afe_0x3e. It is worth noting

that the content of afe_0x3e is reset to its default value only after a power cycle, watchdog reset, or software reset. Therefore, any changes made to the delay using afe_0x3e only take effect if the chip has not undergone these reset conditions. For example, after waking up from deep sleep, the setting in afe_0x3e is effective.

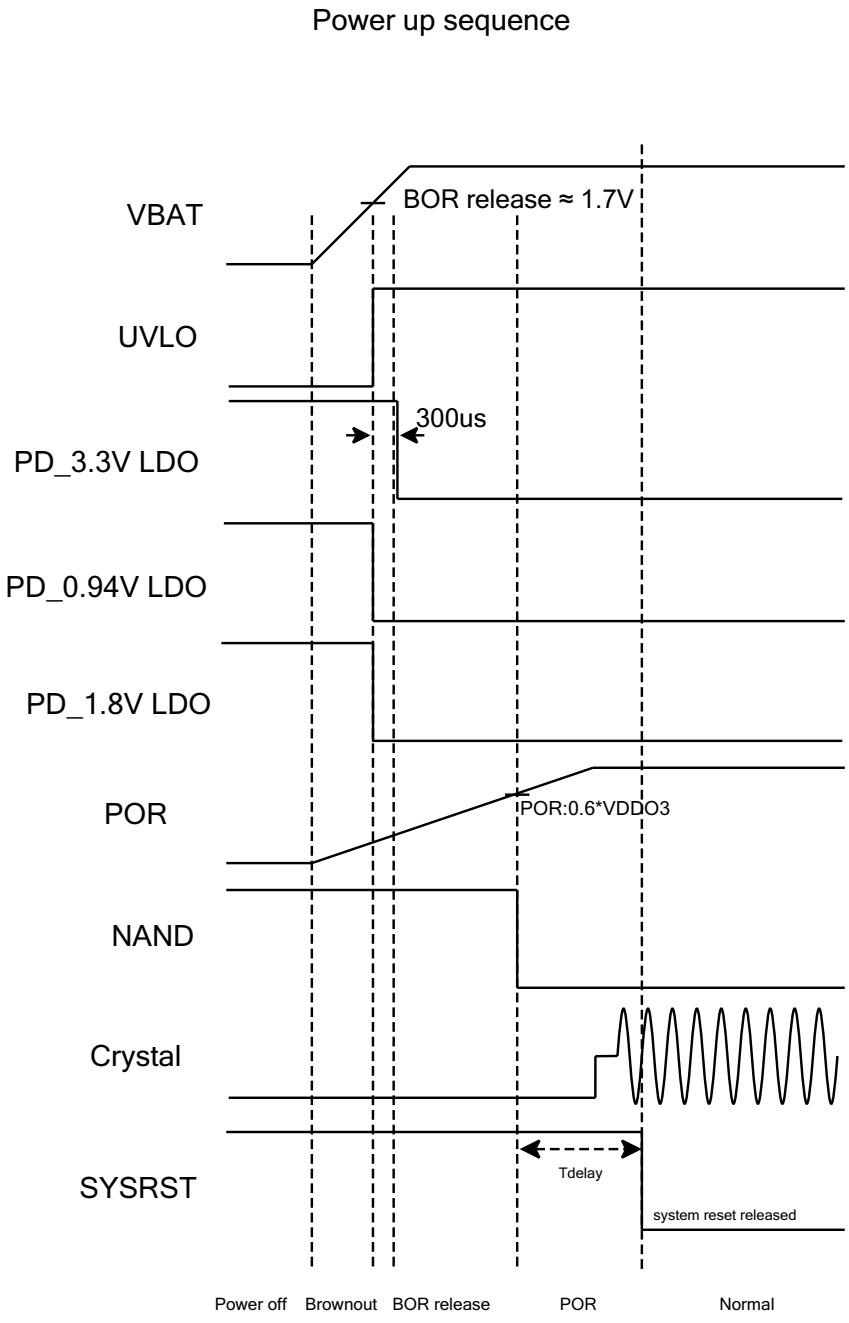
The related analog registers are described in table below.

Table 4-9 Analog Register to Control Logic of Power Up/Down

Address	Name	R/W	Description	Default Value
afe_0x3e	r_dly	R/W	base on 16KHz frequency increase counter (8ms)	10000000

Power up and power down sequences are shown in figures below.

Figure 4-3 Initial Power-up Sequence



NOTE:

- *PD_XXX LDO indicates the power down signal of the LDO voltage, high level means disable LDO, low level means enable LDO.*

Figure 4-4 Initial Power-down Sequence

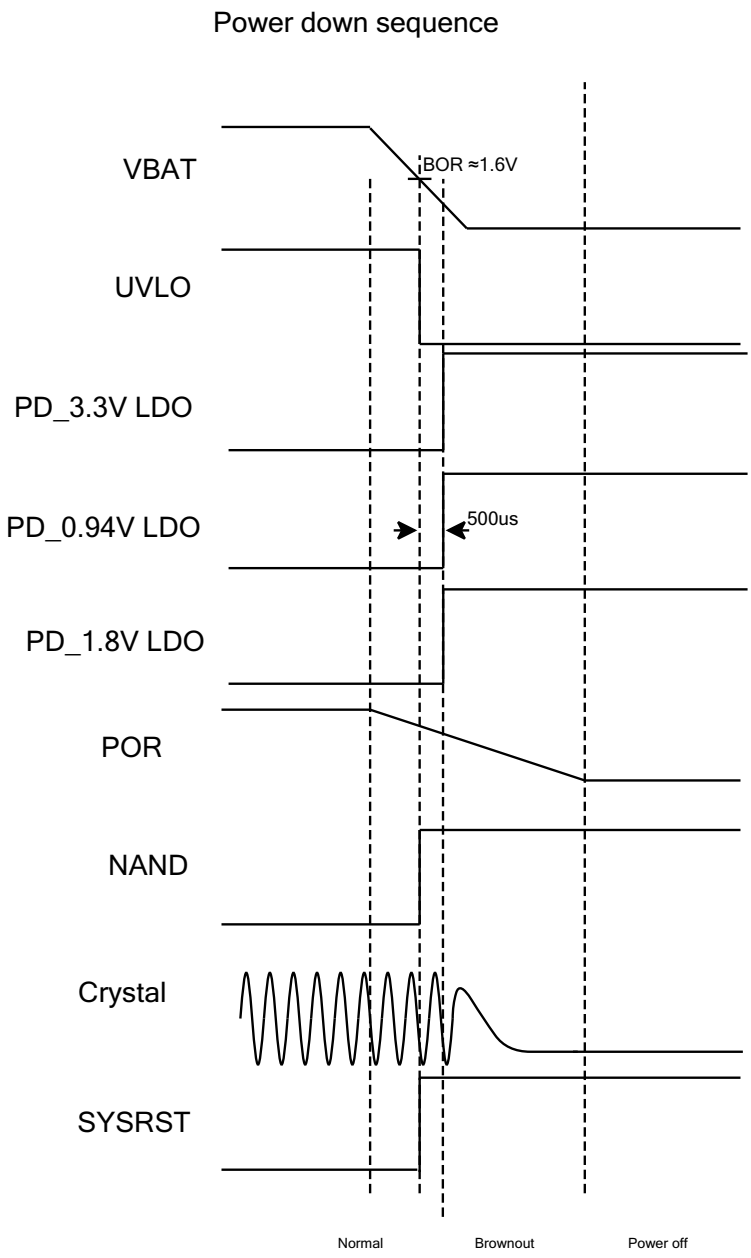


Table 4-10 Characteristics of Initial Power-up/Power-down Sequence

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{POR}	Reset trigger level	-	0.6*VDDO3	-	V
V_{BOR_high}	VDD voltage when V_{UVLO} turns to high level	-	1.7	-	V
V_{BOR_low}	VDD voltage when V_{UVLO} turns to low level	-	1.6	-	V
T_{Delay}	Delay counter value	Configurable via analog register afe_0x3e			

4.5.2 Working Mode Switch

In Active mode, MCU is active, all SRAMs are accessible, and other modules are selectable whether to be at working state.

The chip can switch to Idle mode to stall the MCU. In this mode, all SRAMs are still accessible, modules such as RF transceiver, USB are still selectable whether to be at working state. The chip can be triggered to Active mode by interrupt or POR pin, and the time to switch to Active mode is negligible.

To decrease power consumption to different levels, the chip can switch to power saving mode (Suspend, Deep sleep with SRAM retention, Deep sleep without SRAM retention) correspondingly.

- In Suspend mode, MCU stalls, all SRAMs are still accessible, the PM module is active, and modules such as RF transceiver, USB are powered down. The chip can be triggered to Active mode by 32K Timer, IO pin or POR pin. It takes 100 μs or so to switch from Suspend mode to Active mode.
- In Deep sleep with SRAM retention, the PM module is active, analog and digital modules except for the retention SRAMs are powered down, while the retention SRAMs can be retained and not accessible. The chip can be triggered to Active mode by 32K Timer, IO pin or POR pin. The time to switch to Active mode is shorter than Deep sleep without SRAM retention and close to Suspend.
- In Deep sleep without SRAM retention, only the PM module is active, while analog and digital modules including the retention SRAMs are powered down. The chip can be triggered to Active mode by 32K Timer, IO pin or POR pin. The time to switch to Active mode is 1 ms or so.

User can directly invoke corresponding library function to switch working mode of the chip.

If certain module doesn't need to work, user can power down this module in order to save power.

Table 4-11 3.3 V Analog Register for Module Power up/down Control

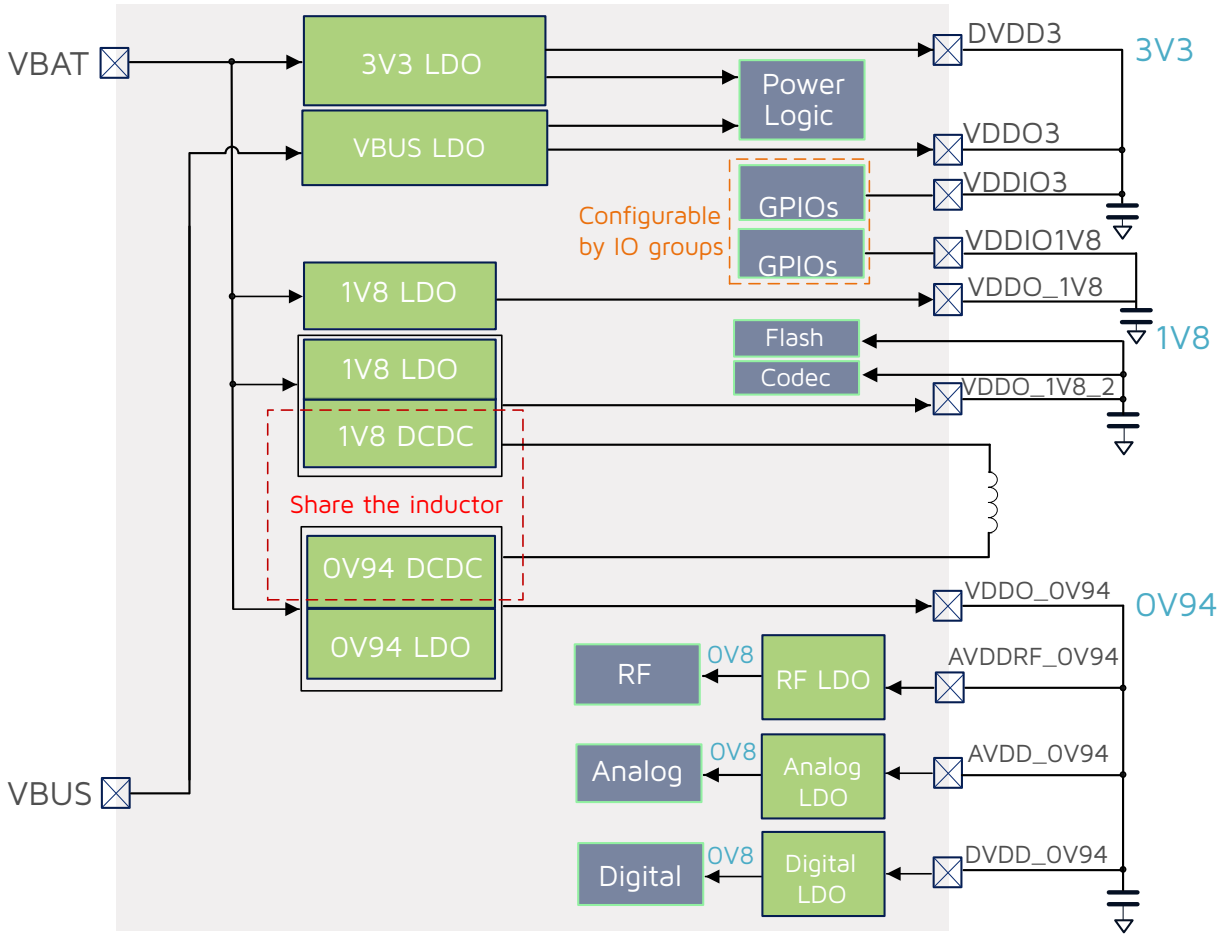
Address	Type	Description	Reset Value
afe_0x4c	R/W	[0] pd_rc32k_auto, 1: auto power down 32KHz RC [1] pd_xtal32k_auto, 1: auto power down 32KHz xtal [2] rsvd [3] pd_xtal24m_auto, 1: auto power down 24MHz xtal [4] pd_pl_all_auto, 1: auto power power logic [5] pd_dcdc auto, 1: auto power down dcdc [6] pd_vbus_ldo_auto, 1: auto power down vbus LDO [7] pd_ana_ldo/pd_bbpll/temp_sens auto, 1: auto power down ana/BBPLL/temp_sensor LDO	0x0

Address	Type	Description	Reset Value
afe_0x4d	R/W	[0] pd_lc_comp auto, 1: auto power down low power comparator [1] pd_ldo_dcore_auto/pd_ldo_sram_auto, 1: auto power down dcore/sram LDO [2] pd_uvlo_ib_auto, 1: auto power down UVLO ib [3] pd_vbus_sw_auto, 1: auto power down vbus switch [4] rsvd [5] rsvd [6] pwdn_en, 1: power down sequence enable [7] iso_en, 1: enable isolation	0x0

4.5.3 LDO and DCDC

The diagram of LDO and DCDC module is shown as following.

Figure 4-5 LDO and DCDC



As shown in figure above, the SoC operates with two power supply modes: VBAT and VBUS. Upon power up, the 3.3 V LDO and VBUS LDO generate 3.3 V voltage output and supply power for Power logic module and 3.3 V GPIO group; one channel of 1.8 V LDO produces 1.8 V voltage for 1.8 V GPIO group, the other channel of 1.8 V LDO and 1.8 V DCDC produce 1.8 V voltage output and supply power for Flash and CODEC modules; the 0.94 V LDO/DCDC generates 0.94 V voltage output that serves as input for the internal RF LDO, analog LDO and digital LDO; the three LDOs are responsible for supplying power to the RF, Analog, and Digital modules respectively.

4.5.4 VBAT and VANT Power-Supply Mode

The RF PA module has two power-supply modes including VBAT mode and VANT mode.

- In VBAT mode, the RF PA module is supplied by 3.3 V voltage regulated from 4.2V lithium battery or directly from two AA/AAA batteries in series. The maximum output power is related to power supply voltage of RF PA, for example, the maximum output power is 10 dBm at 3.3 V power supply.
- In VANT mode, the RF PA module is supplied with 0.94 V voltage by the embedded DCDC and LDO. In this mode, the output power won't change with AVDD3 which is converted from VBAT voltage, and the maximum output power is 5 dBm or less.

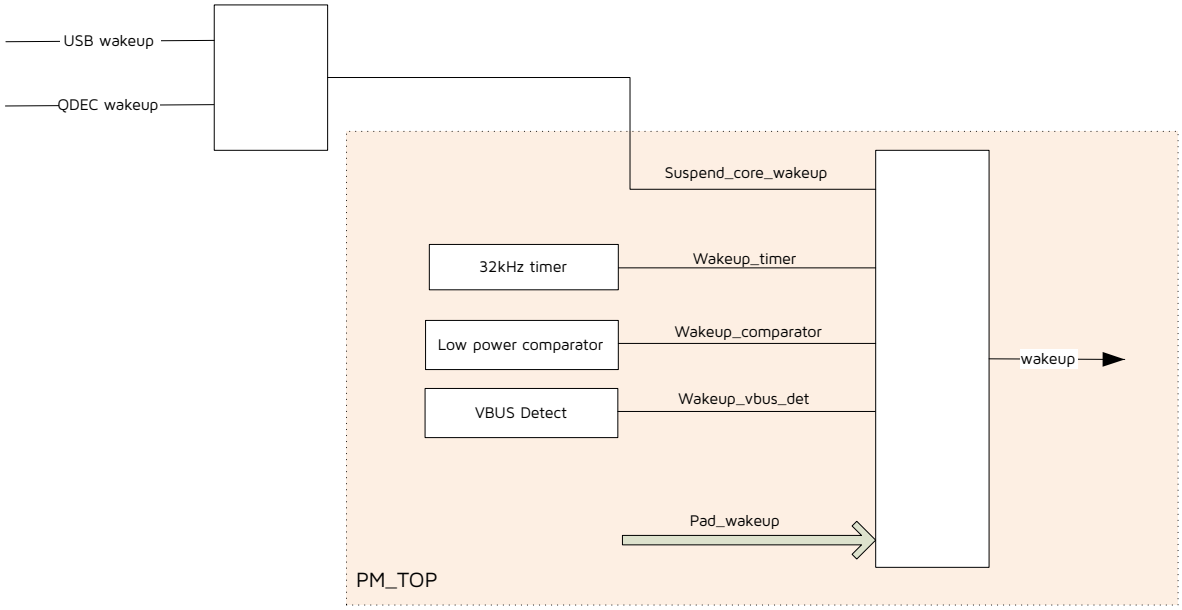
Comparing to the VBAT mode, the VANT mode is more power-saving at the same TX power.

When the chip works in VBAT mode, it can be configured to the maximum output power. However, as the VBAT/VDD supply decreases below 3.0 V, the maximum transmit power of TX is then slightly attenuated. The detailed RF transmit power level refers to the code comments in the corresponding driver SDK, in which the RF transmit power level under VBAT mode is the result tested in 3.3 V VBAT voltage.

4.6 Wakeup Source

The figure below shows wake up sources of the SoC.

Figure 4-6 Wake up Sources



Each wake up source is detailed below:

USB & QDEC

This wakeup source can only wake up the system from suspend mode.

For USB wakeup, once USB host sends out resuming signal, the system is woke up.

For QDEC wakeup, it is mainly used in mouse applications.

32 kHz Timer

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes.

Low Power Comparator

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes.

VBUS Detect

This wakeup source is able to wake up the system from suspend mode or two deep sleep modes.

Pad wakeup

This wakeup source is from IO signals and able to wake up the system from suspend mode or two deep sleep modes.

NOTE:

- *Tested in TL7218H chip, when GPIO is set to be powered by 1.8V and configuring pad low-level wakeup, pull up PE7 or other GPIO to 1.8V internally or externally, there is 12uA current leakage during deep sleep. To solve this issue, the GPIO can only be configured as high-level wakeup in deep sleep, which requires pulling down the GPIO to 0V internally or externally.*

Table 4-12 Analog Register for Wakeup

Address	Type	Description	Default Value
afe_0x3f	R/W	PA_polarity wakeup polarity 0: high level wakeup, 1: low level wakeup	0x0
afe_0x40	R/W	PB_polarity wakeup polarity 0: high level wakeup, 1: low level wakeup	0x0
afe_0x41	R/W	PC_polarity wakeup polarity 0: high level wakeup, 1: low level wakeup	0x0
afe_0x42	R/W	PD_polarity wakeup polarity 0: high level wakeup, 1: low level wakeup	0x0
afe_0x43	R/W	PE_polarity wakeup polarity 0: high level wakeup, 1: low level wakeup	0x0
afe_0x44	R/W	PF_polarity wakeup polarity 0: high level wakeup, 1: low level wakeup	0x0
afe_0x45	R/W	PA wakeup enable	0x0
afe_0x46	R/W	PB wakeup enable	0x0
afe_0x47	R/W	PC wakeup enable	0x0
afe_0x48	R/W	PD wakeup enable	0x0
afe_0x49	R/W	PE wakeup enable	0x0
afe_0x4a	R/W	PF wakeup enable	0x0
afe_0x4b	R/W	[0] pad wakeup enable [1] dig wakeup enable [2] timer wakeup enable [3] comparator wakeup enable [4] rsvd [5] rsvd [6] rsvd [7] rsvd	0x0

Address	Type	Description	Default Value
afe_0x64	R	write 1 to clean the status: [0]: wkup pad [1]: wkup dig [2]: wkup timer [3]: wkup cmp [4]: rsvd [5]: rsvd [6]: rsvd [7]: vbus on	-
afe_0x7f	R/W	[3]: vbus_detect_pol, vbus detect wakup polarity, 1: low level active, 0: high level active [4]: wkup_vbus_en, vbus detect wakeup enable	0x0

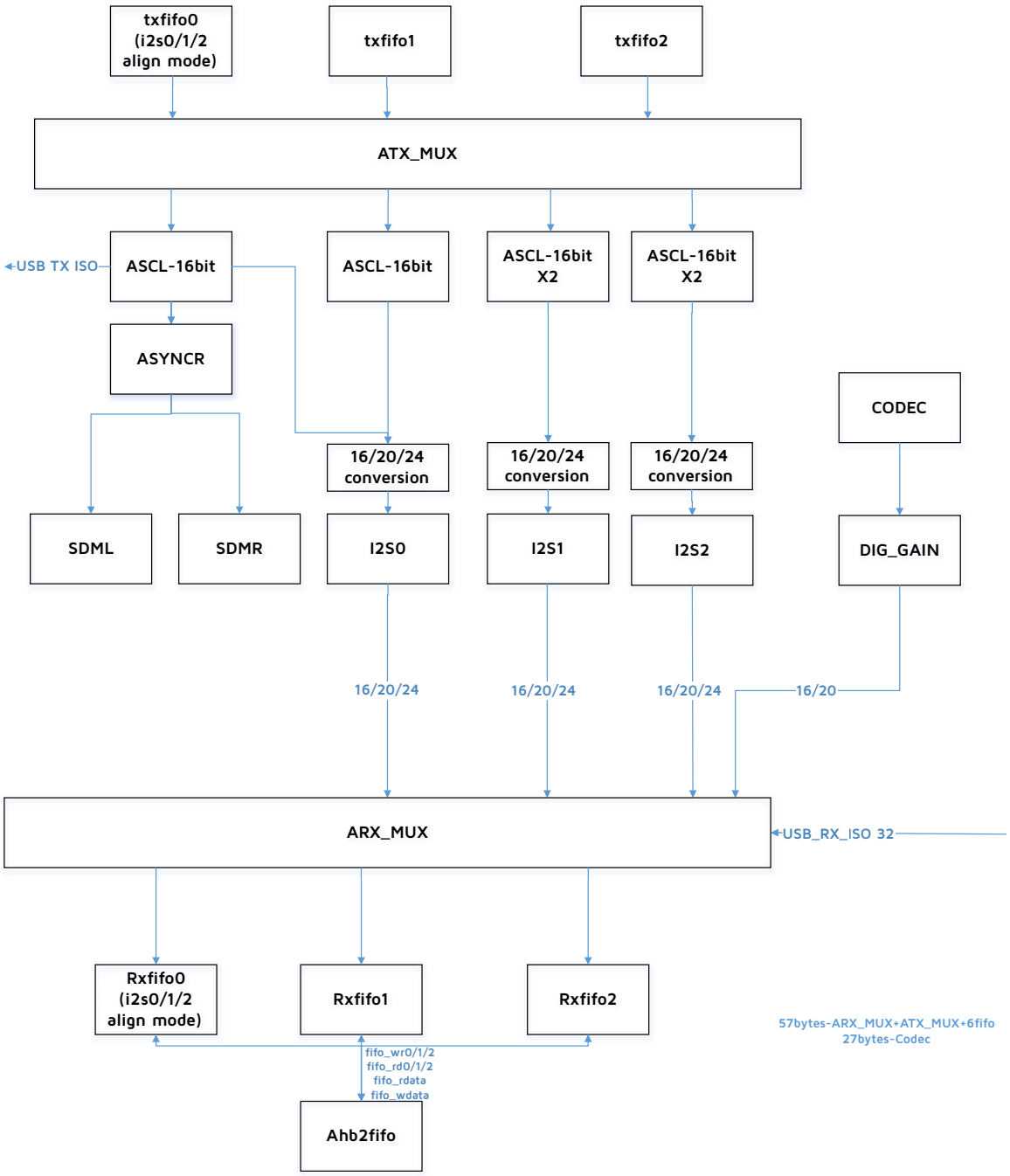
5 Audio

This chapter includes audio architecture, system clock, SDM (Sigma-Delta Modulation), ASCL (Asynchronous Sample rate Conversion with Linear interpolation), CODEC, FIFO (First-In-First-Out), MUX, and Interrupt.

5.1 Introduction

The figure below shows the audio architecture.

Figure 5-1 Audio Architecture

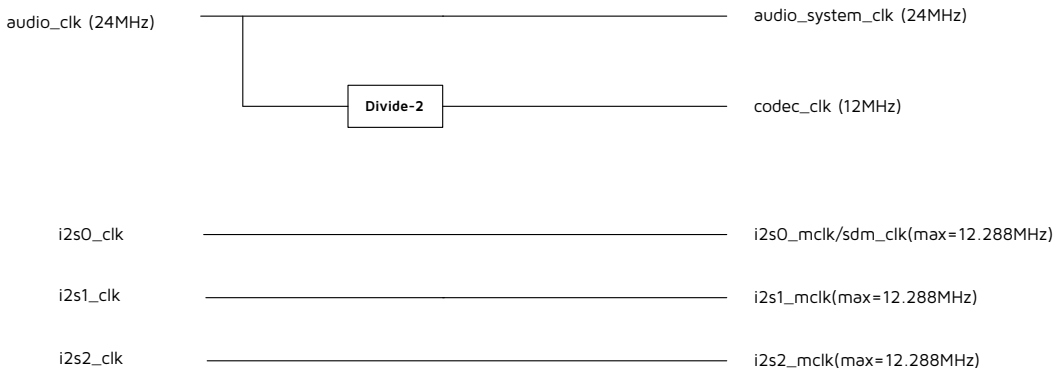


The audio features include:

- Supports 3-channel I2S (in which I2S2 supports TDM, Time Division Multiplexing mode)
- Supports 1-group stereo SDM
- The internal CODEC supports 1-channel ADC or 1-group stereo DMIC
- Supports 6-channel DMA transmission (3-channel RX + 3-channel TX)
- 3-channel I2S and TX of SDM support adjusting transmitting sample rate via ASCL

5.2 Audio System Clock

Figure 5-2 Audio System Clock



There are 3 types of clocks in this audio system: audio_clk, i2s_clk, hclk, pclk.

audio_clk: this clock is the master clock of the audio system, the modules of MUX, ASCL, CODEC, DIG_GAIN in the audio_arch architecture diagram is using this clock. The audio_clk needs to be fixed to 24 MHz (22.5792 MHz, 24.576 MHz, 33.8688 MHz, 36.864 MHz in some special cases), and in the Codec, 2 divisions are performed to get 12 MHz (11.2896 MHz, 12.288 MHz, 16.9344 MHz, 18.432 MHz in some special cases) as the master clock of the codec.

- i2s_clk: this clock is used as the main clock of SDM and the interface clock of i2s0/1/2.
- SDM_clk: the clock output from SDM shares i2s0_clk with i2s0.
- hclk: clock for DMA and usb_iso interfaces.
- pclk: clock for configuration registers.

5.3 I2S0/1/2

5.3.1 Introduction

There are three completely independent I2S modules in this chip, which can be connected to the peripheral CODEC through GPIO configuration and support I2S0, I2S1 dual module synchronization and I2S synchronization mode between multiple chips, Master/Slave mode.

This section briefly introduces some I2S protocols, as well as I2S module clock and some functional register configuration. As I2S0 and I2S1 are completely independent and the same in function, this chapter takes I2S0 as an example to introduce.

I2S features include:

- Supports master and slave mode
- I2S2 supports TDM mode, TDM mode supports FIFO2 channel only and supports 4/6/8 channel, the slot of TDM supports 16/24/32 BCLK wide
- Supports 16 bits / 20 bits / 24 bits data width
- Supports sampling rate of less than or equal to 192 kHz
- TDM supports sampling rate of less than or equal to 48 kHz (8 channels on simultaneously)
- Supports 2-line mode, two data lines in or out at the same time
- I2S0/1/2 supports align mode

5.3.2 I2S Protocol

The I2S protocols include: I2S format, Left Justified (LJ) format, Right Justified (RJ) format, DSP format (mode A and mode B), and TDM mode for I2S2. The frame clock and MSB position of each format are listed as below.

Table 5-1 Frame Clock and MSB Position

Format	Frame Clock Mode	MSB Position from Start of Frame Clock
I2S	50% duty cycle	One bit clock delay
Left Justified	50% duty cycle	No delay
Right Justified	50% duty cycle	16bit mode (Delay by 16bit clocks) 20bit mode (Delay by 12bit clocks) 24bit mode (Delay by 8bit clocks) Note: config 32bit bclks for fclk
DSP mode A	Single bit clock wide pulse	No delay
DSP mode B	Single bit clock wide pulse	One bit clock delay
TDM mode 1 (I2S2 only)	Single bit clock wide pulse	No delay
TDM mode 2 (I2S2 only)	Single bit clock wide pulse	One bit clock delay
TDM mode 3 (I2S2 only)	50% duty cycle	No delay

Since the function and register offset address of I2S0, I2S1 and I2S2 are the same except TDM, here describes the function configuration of I2S taking I2S0 as an example.

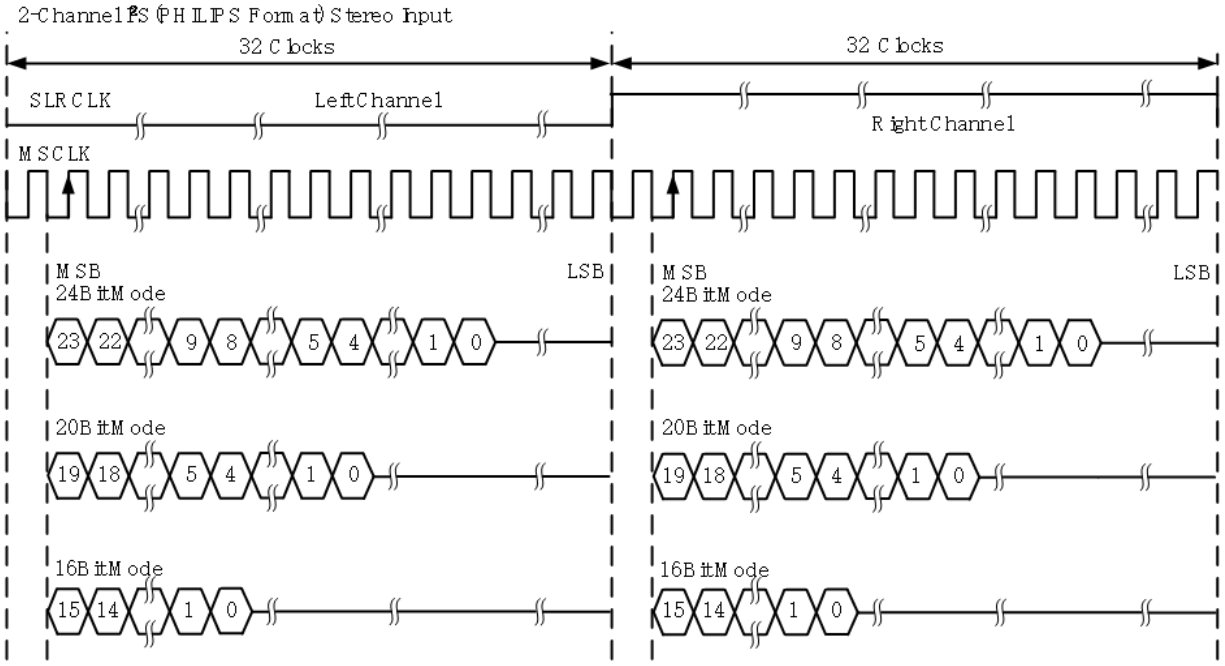
- The supported I2S, LJ, RJ and DSP format is set by configuring register i2s0_format (I2S0_BASE+0x01[6:5]);
- The data bit width of 16 bits, 20 bits, 24 bits is selected by configuring register i2s_wl (I2S0_BASE+0x01[4:3]);
- The register i2s0_lrp (I2S0_BASE+0x02[6]) is used to switch mode A and mode B in DSP format, and to reverse SLRCLK in other format;
- The register i2s0_lrswap (I2S0_BASE+0x02[0]) is used to reverse the data of left and right channels;

- The I2S module supports slave and master mode, by configuring register i2s0_adc_dci_ms and i2s0_dac_dci_ms(I2S0_BASE+0x00[6:5]) these two bits to control the ADC/DAC master or slave mode;
- The I2S supports four/five wire mode, four wire refers to the ADC and DAC channel share a SLRCLK, configure i2s0_mode(I2S0_BASE+0x03[1:0]) to 2'b01 to set ADC and DAC paths to share the SLRCLK of DAC, configure i2s0_mode(I2S0_BASE+0x03[1:0]) for 2'b10 to set ADC and DAC paths to share the SLRCLK of ADC.
- The I2S supports dual-line mode which shares bclk and lrc. Configure i2s0_rx_2line_enable(I2S0_BASE+0x02[1]) to enable rx_2line, configure i2s0_tx_2line_enable(I2S0_BASE+0x02[2]) to enable tx_2line, however, rx_2line and tx_2line should not be enabled at the same time.

5.3.2.1 I2S mode

The timing sequence of I2S mode is shown as below.

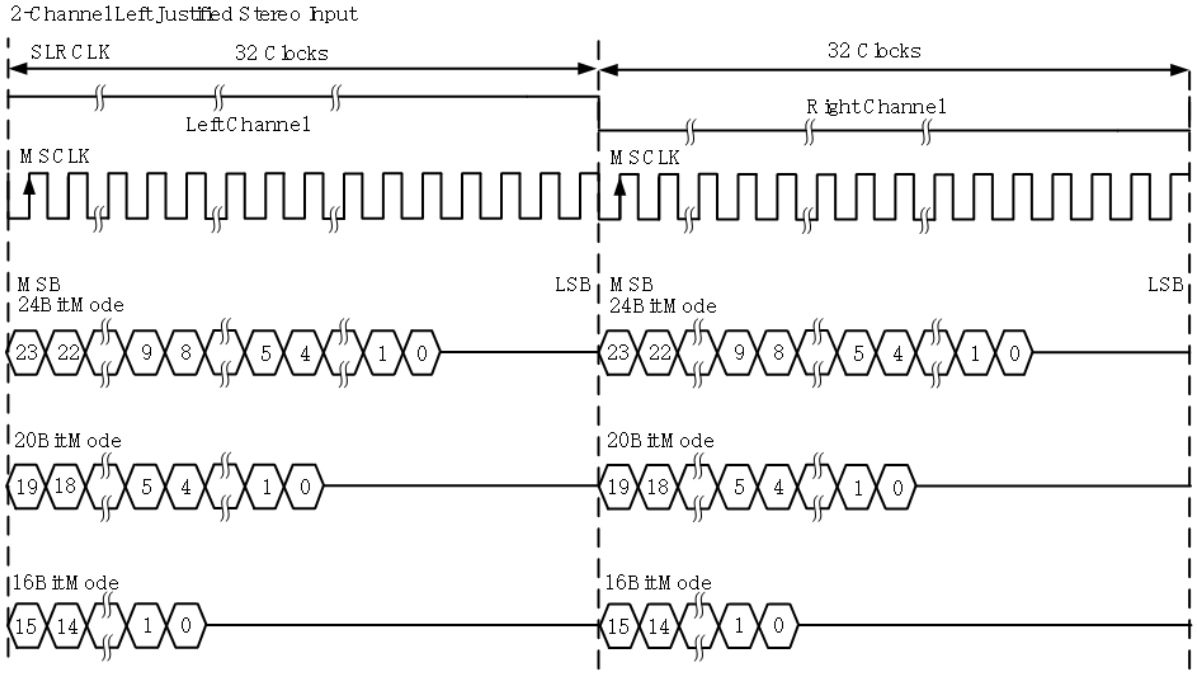
Figure 5-3 Timing Diagram of I2S Mode



5.3.2.2 Left Justified mode

The timing sequence of LJ mode is shown as below.

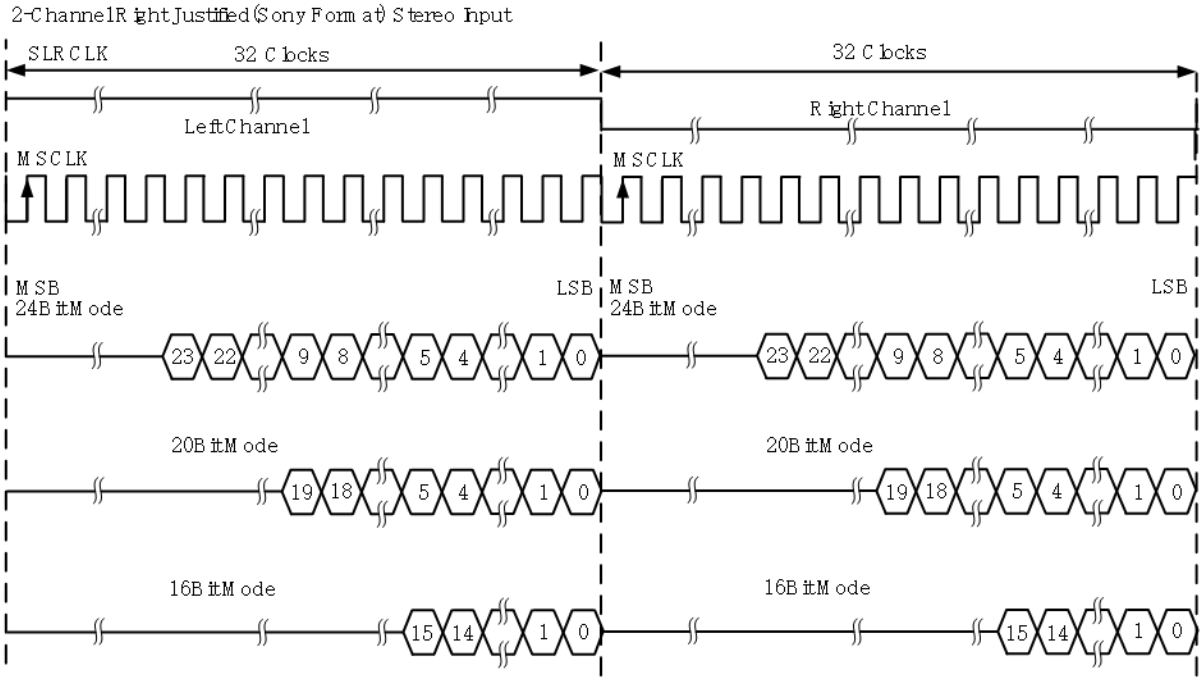
Figure 5-4 Timing Diagram of LJ Mode



5.3.2.3 Right Justified mode

The timing sequence of RJ mode is shown as below.

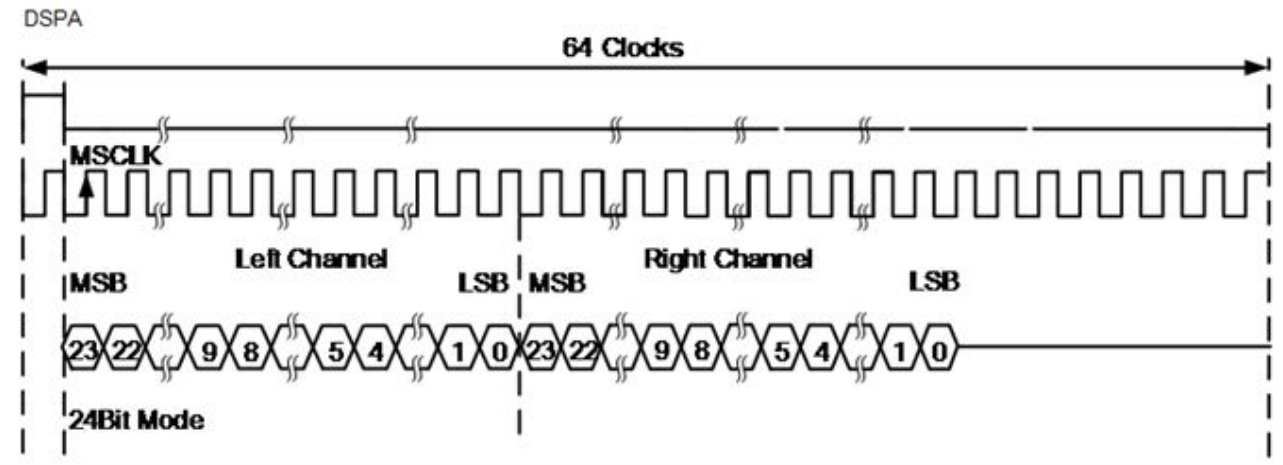
Figure 5-5 Timing Diagram of RJ Mode



5.3.2.4 DSP mode A

The timing sequence of DSP mode A is shown as below.

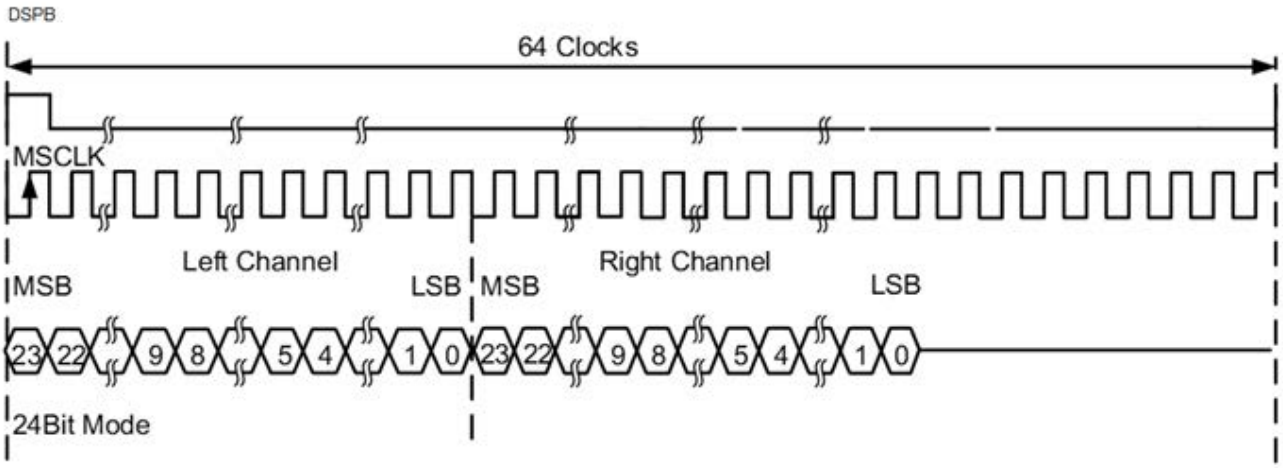
Figure 5-6 Timing Diagram of DSP Mode A



5.3.2.5 DSP mode B

The timing sequence of DSP mode B is shown as below.

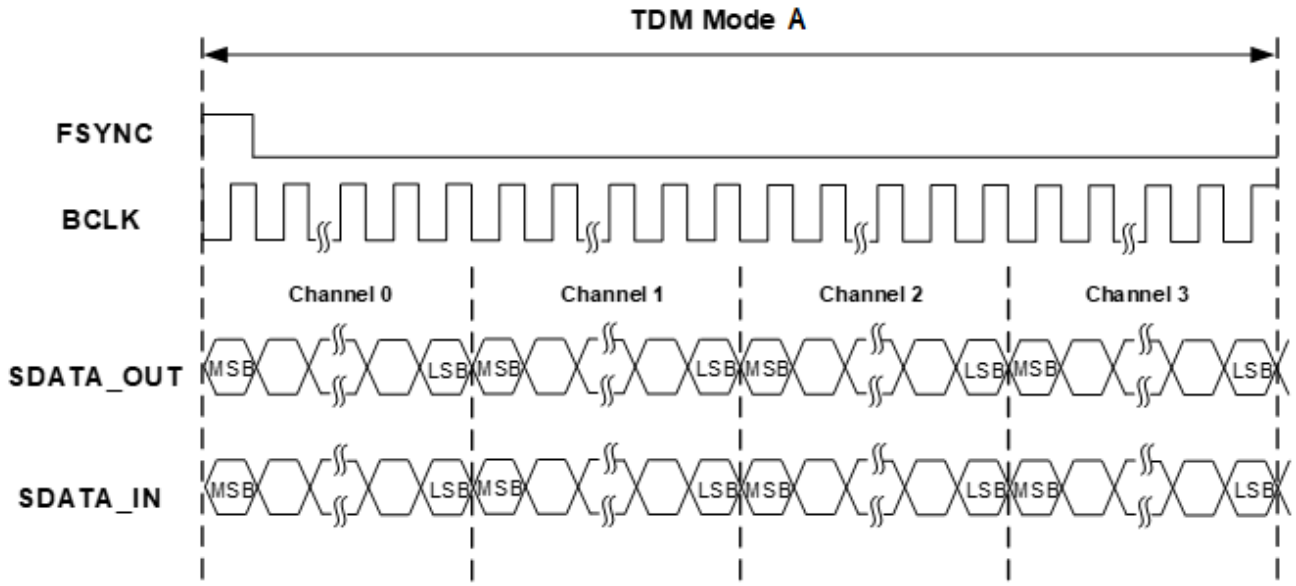
Figure 5-7 Timing Diagram of DSP Mode B



5.3.2.6 TDM mode A

The timing sequence of TDM mode A is shown as below.

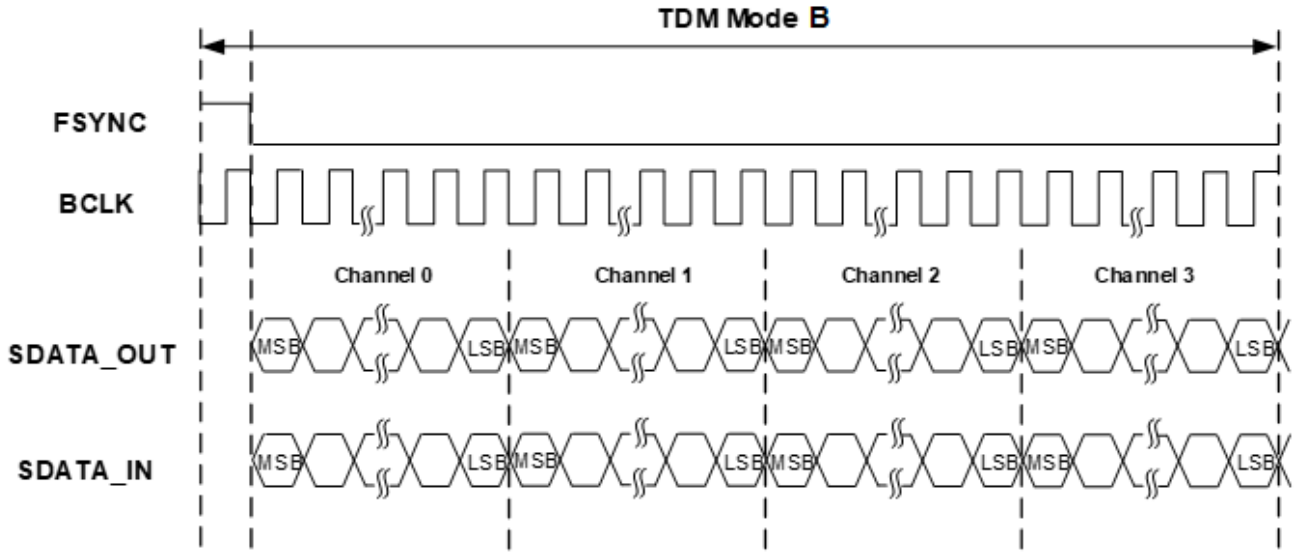
Figure 5-8 Timing Diagram of TDM Mode A



5.3.2.7 TDM mode B

The timing sequence of TDM mode B is shown as below.

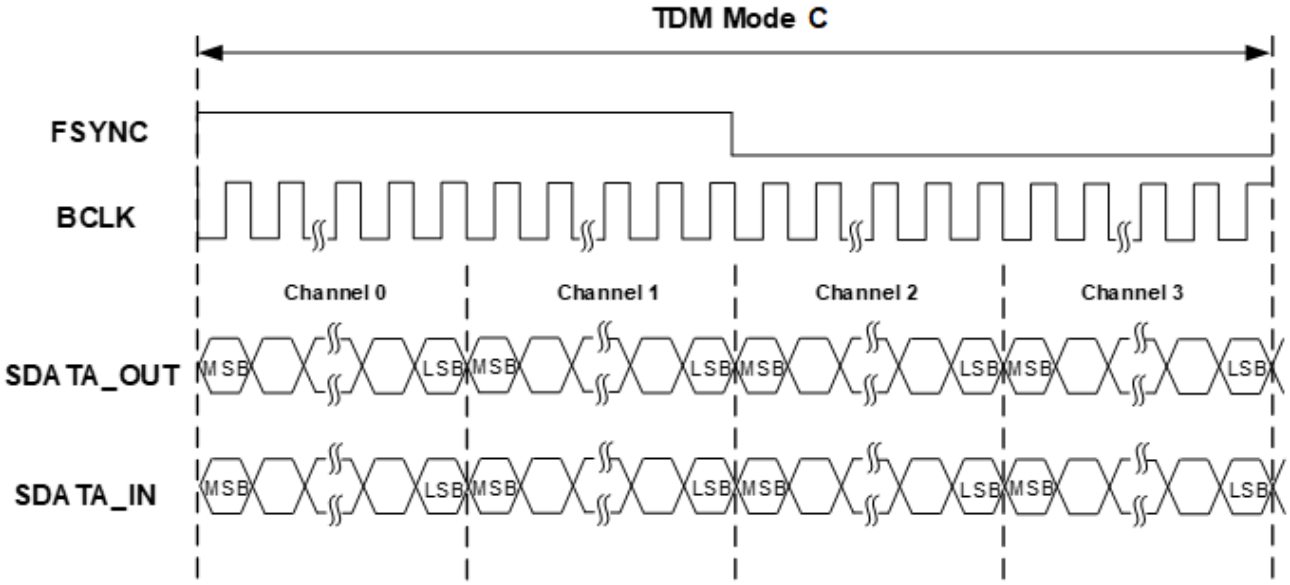
Figure 5-9 Timing Diagram of TDM Mode B



5.3.2.8 TDM mode C

The timing sequence of TDM mode C is shown as below.

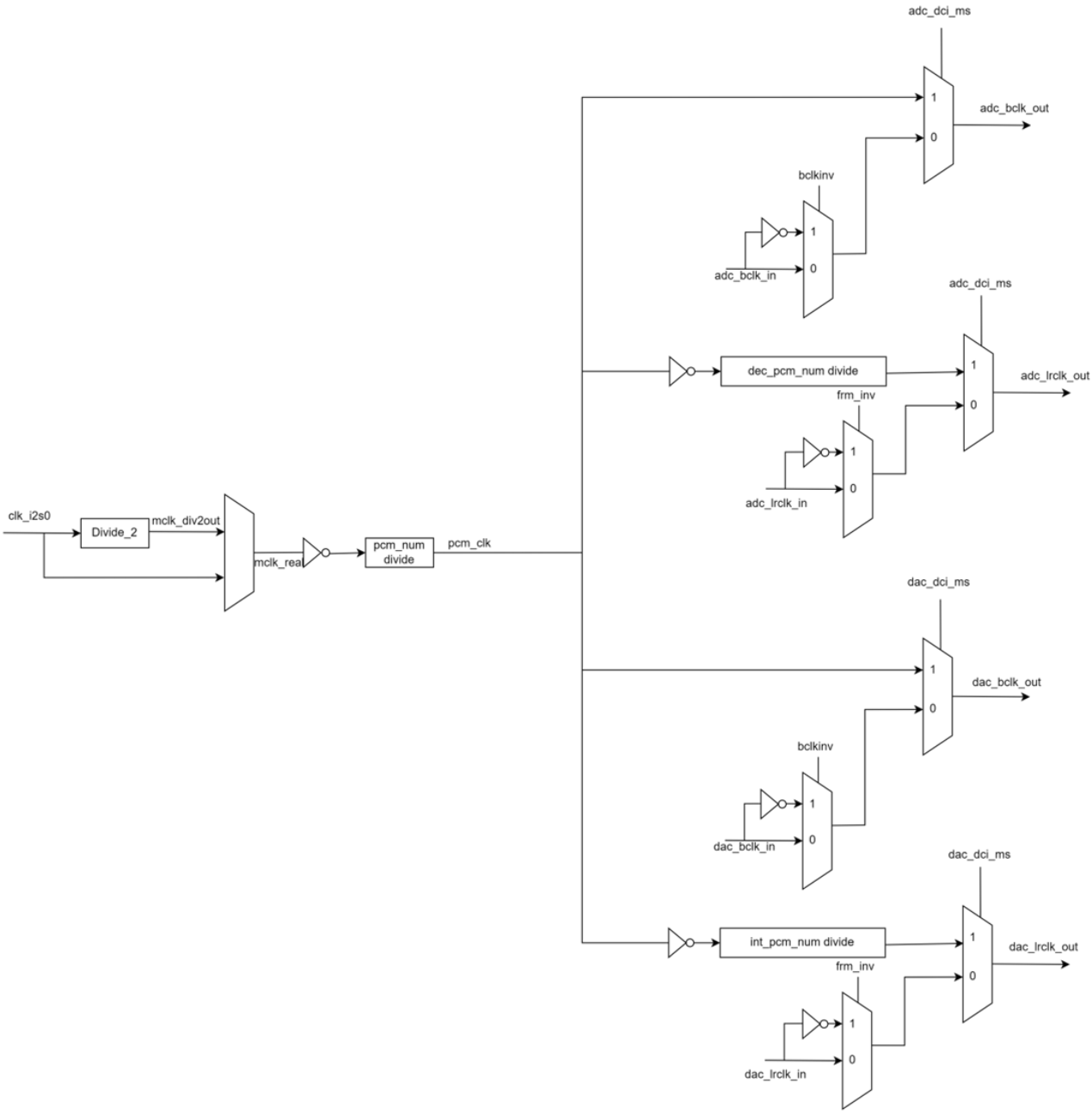
Figure 5-10 Timing Diagram of TDM Mode C



5.3.3 I2S Clock

Take the I2S0 module as an example, the clock tree is shown in the figure below, in which clk_i2s0 is divided from pll (Phase Locked Loop).

Figure 5-11 Clock Tree of I2S0 Module



- The i2s0_clk_en(I2S0_BASE+0x00[2]) is the I2S clock switch;
- We can choose whether to divide frequency for clk_i2s0 by configuring i2s0_clk_div2(I2S0_BASE+0x00[3]);
- When I2S module works as master, we can configure i2s0_pcm_clk_num(I2S0_BASE+0x08) to divide the frequency of clk_i2s0 to get bclk, which currently only supports even divisions (0, 2, 4...);
- By configuring i2s0_int_pcm_num(I2S0_BASE+0x04~0x05) and i2s0_dec_pcm_num(I2S0_BASE+0x06~0x07), we can get the dac_lrclk and adc_lrclk which are divided down by the bclk. For example, if bclk is 12MHz, dac_lrclk and adc_lrclk are both 48KHz, then configure i2s0_int_pcm_num and i2s0_dec_pcm_num to 249, that is, 48K = 12M/(249+1);

5.3.4 I2S Align Mode

Because there are external dual CODECs and the need of phase to be the same, it supports multiple i2s align mode.

The supported i2s align combinations are i2s0/i2s1/i2s2, i2s0/i2s1, i2s1/i2s2.

- i2s_align_en(I2S2_TDM_BASE+0x10[0]) is the master switch for the align mode of the i2s module.
- i2s_align_ctrl(I2S2_TDM_BASE+0x10[3:1]) is the enable for controlling the align mode of the i2s0/ i2s1/i2s2 modules respectively.
- i2s_clk_sel(I2S2_TDM_BASE+0x10[5]) is to configure whether the i2s clocks in i2s align mode should use i2s_align_clk (unified as i2s1_clk) or their respective i2s_clk, and it is recommended that i2s_align_clk be used as the clock for each module in align mode.

The following is an example of i2s0/i2s1 synchronization to illustrate the use of align mode:

1. Configure the mode of i2s0 and i2s1, and i2s1_clk.
2. Write 1 for i2s_align_en and 3'b011 for i2s_align_ctrl.
3. Write 0 for i2s_align_mask(I2S2_TDM_BASE+0x10[4]).
4. Configure i2s_timer_th(I2S2_TDM_BASE+0x14~0x17) threshold value.
5. In align mode, when the sys_timer arrives i2s_timer_th threshold value, it triggers i2s0 and i2s1 at the same time.

5.3.5 I2S2 TDM Mode

Configuring I2S2 to TDM mode requires i2s2_format(I2S2_TDM_BASE+0x01[7:5]) to be configured to 3'b100.

The specific choice of which TDM format needs to be configured i2s2_tdm_mode(I2S2_TDM_BASE+0x11[5:4]); the tx and rx channels of TDM can be configured separately, the relevant registers are i2s2_tdm_rx_ch_num(I2S2_TDM_BASE+0x11[1:0]) and i2s2_tdm_tx_ch_num(I2S2_TDM_BASE+0x11[3:2]); the slot width of each channel can be selected by register i2s2_tdm_slot(I2S2_TDM_BASE+0x11[7:6]), and 16/24/ 32 bclk width is optional.

5.3.6 I2S Registers

The I2S0/I2S1 related registers are listed as following. For I2S0 related register, the base address is 0x801410b0; for I2S1 related register, the base address is 0x801410d0.

Table 5-2 I2S Related Registers

Address offset	Name	Type	Description	Default value
0x00	I2S_CFG1	RW	[1:0]: i2s_bcm_bits BCLK frequency: 2'd0:BCM function disabled; 2'd1:MCLK/4; 2'd2:MCLK/8; 4'd3:MCLK/16 [2]: i2s_clk_en, clk enable [3]: i2s_clk_div2, i2s clk divide2 enable: 1'b1: enable 1'b0: disable [4]: i2s_bclkinv_o, bclk invert [5]: i2s_adc_dci_ms, i2s adc as master [6]: i2s_dac_dci_ms, i2s dac as master [7]: i2s_adc_frm_loop, adc frm loop from pad	0x00
0x01	I2S_CFG2	RW	[0]: i2s_adc_mbclk_loop 1'b0: adc bclk from i2s clk 1'b1: adc bclk loop gpio [1]: i2s_dac_mbclk_loop 1'b0: dac bclk from i2s clk 1'b1: dac bclk loop gpio [2]: i2s_frm_inv 1'b0: frm; 1'b1 frm_inv [4:3]: i2s_wl, i2s word length: 2'b00: 16 2'b01: 20 2'b10: 24 [6:5]: i2s_format, i2s format: 2'b00: RJ 2'b01: LJ 2'b10: I2S 2'b11: DSP	0x00

Address offset	Name	Type	Description	Default value
0x02	I2S_CFG3	RW	[0]: i2s_lrswap, i2s data swap [1]: i2s_rx_2line_enable [2]: i2s_tx_2line_enable [3]: i2s_dac_frm_enable 0: dac_frm_clk disable 1: dac_frm_clk enable [4]: i2s_adc_frm_enable 0: adc_frm_clk disable 1: adc_frm_clk enable [5]: i2s_tx_dat_sel 0: sel i2s tx data low bits 1: sel i2s tx dat high bits [6]: i2s_lrp DSP mode select when i2s DSP format or LRCLK invert operation: 1'b1: dsp mode A 1'b0: dsp mode B 1'b1: LRCLK invert 1'b0: not invert [7]: i2s_dac_frm_loop, dac frm loop from pad	0x00
0x03	I2S_ROUTE	RW	[1:0]: i2s_mode [2]: i2s_pad_bclk_sel 0:adc bclk; 1: dac bclk [3]: i2s_rec_bit_sel 0: low byte; 1: high byte [4]: i2s_schedule_en, schedule enable [5]: i2s_ascl_bypass 0: enable ascl 1:ascl bypass [6]: i2s_pem_trig_en 1: trig i2s via pem [7]: i2s_pem_dis_en 1: disable i2s via pem	0x00

Address offset	Name	Type	Description	Default value
0x04	I2S_INT_PCM_NUM0	RW	[7:0]: i2s_int_pcm_num0 dac i2s LRCLK counter low byte.	0x00
0x05	I2S_INT_PCM_NUM1	RW	[4:0]: i2s_int_pcm_num1 dac i2s LRCLK counter high byte.	0x00
0x06	I2S_DEC_PCM_NUM0	RW	[7:0]: i2s_dec_pcm_num0 adc i2s LRCLK counter low byte.	0x00
0x07	I2S_DEC_PCM_NUM1	RW	[4:0]: i2s_dec_pcm_num1 dac i2s LRCLK counter high byte.	0x00
0x08	I2S_PCM_CLK_NUM	RW	[7:0]: i2s_pcm_clk_num bclk division factor, i2s_clk/(n*2).	0x00
0x0c	I2S_STIMER_TARGET0	RW	[7:0]: i2s_stimer_target[7:0] i2s stimer for schedule	0x00
0x0d	I2S_STIMER_TARGET1	RW	[7:0]: i2s_stimer_target[15:8] i2s stimer for schedule	0x00
0x0e	I2S_STIMER_TARGET2	RW	[7:0]: i2s_stimer_target[23:16] i2s stimer for schedule	0x00
0x0f	I2S_STIMER_TARGET3	RW	[7:0]: i2s_stimer_target[31:24] i2s stimer for schedule	0x00
0x10	RX_DSP_START_SEL	RW	[6]: rx_dsp_start_sel fix pad2reg timing in i2s_slv mode	0x00

The I2S2_TDM related registers are listed as following, the base address of the following registers is 0x80141000.

Table 5-3 I2S2_TDM Related Registers

Address offset	Name	Type	Description	Default value
0x00	I2S2_CFG1	RW	[1:0]: i2s2_bcm_bits, BCLK frequency: 2'd0:BCM function disabled; 2'd1:MCLK/4; 2'd2:MCLK/8; 4'd3:MCLK/16 [2]: i2s2_clk_en, i2s2 clk enable: 1'b1: enable 1'b0: disable [3]: i2s2_clk_div2, i2s clk divide2 enable: 1'b1: enable 1'b0: disable [4]: i2s2_bclkinv_o, bclk invert [5]: i2s2_adc_dci_ms, i2s2 adc as master [6]: i2s2_dac_dci_ms, i2s2 dac as master [7]: i2s2_adc_frm_loop, adc frm loop from pad	0x00
0x01	I2S2_CFG2	RW	[0]: i2s2_adc_mbclk_loop 1'b0: i2s2_adc_bclk 1'b1: i2s2_adc_bclk loop GPIO [1]: i2s2_dac_mbclk_loop 1'b0: i2s2_dac_bclk 1'b1: i2s2_dac_bclk loop GPIO [2]: i2s2_frm_inv 1'b0: frm; 1'b1 frm_inv [4:3]: i2s2_wl, i2s2 word length: 2'b00: 16 2'b01: 20 2'b10: 24 [6:5]: i2s2_format, i2s2 format: 3'b000: RJ 3'b001: LJ 3'b010: I2S 3'b011: DSP 3'b100 TDM	0x00

Address offset	Name	Type	Description	Default value
0x02	I2S2_CFG3	RW	[0]: i2s2_lrswap i2s2 l channel and r channel data swap. [1]: i2s2_rx_2line_enable 0: i2s2 2line rx disbale; 1: i2s2 2line rx enable [2]: i2s2_tx_2line_enable 0: i2s2 2line tx disable; 1: i2s2 2line tx enable [3]: i2s2_dac_frm_enable 0: i2s2 dac frm disable 1: i2s2 dac frm enable [4]: i2s2_adc_frm_enable 0: i2s2 adc frm disable 1: i2s2 adc frm enable [5]: i2s_tx_dat_sel 0: sel i2s2 tx data low bits 1: sel i2s2 tx dat high bits [6]: i2s_lrp DSP mode select when i2s DSP format or LRCLK invert operation: 1'b1: dsp mode A 1'b0: dsp mode B 1'b1: LRCLK invert 1'b0: not invert [7]: i2s2_dac_frm_loop, dac frm loop from pad	0x00

Address offset	Name	Type	Description	Default value
0x03	I2S2_ROUTE	RW	[1:0]: i2s2_mode 2'b00: i2s2 5line mode 2'b01: i2s2 4line dac mode 2'b10: i2s2 4line adc mode [2]: i2s2_pad_bclk_sel 0:adc bclk; 1: dac bclk [3]: i2s2_rec_bit_sel 0: low byte; 1: high byte [4]: i2s2_schedule_en, schedule enable [5]: i2s2_ascl_bypass 0: enable ascl 1:ascl bypass [6]: i2s2_pem_trig_en 1: trig i2s2 via pem [7]: i2s2_pem_dis_en 1: disable i2s2 via pem	0x00
0x04	I2S2_INT_PCM_NUM0	RW	[7:0]: i2s2_int_pcm_num0 dac i2s2 LRCLK counter low byte.	0x00
0x05	I2S_INT2_PCM_NUM1	RW	[4:0]: i2s2_int_pcm_num1 dac i2s2 LRCLK counter high byte.	0x00
0x06	I2S2_DEC_PCM_NUM0	RW	[7:0]: i2s2_dec_pcm_num0 adc i2s2 LRCLK counter low byte.	0x00
0x07	I2S2_DEC_PCM_NUM1	RW	[4:0]: i2s2_dec_pcm_num1 dac i2s2 LRCLK counter high byte.	0x00
0x08	I2S2_PCM_CLK_NUM	RW	[7:0]: i2s2_pcm_clk_num bclk division factor, num*2 div	0x00
0x09	I2S_DACTUNE	RW	[3:0]: i2s_dactune_l1 [7:4]: i2s_dactune_l2	0x00
0x0a	I2S_ADCTUNE	RW	[3:0]: i2s_adctune_l1 [7:4]: i2s_adctune_l2	0x00

Address offset	Name	Type	Description	Default value
0x0b	I2S_FIFO_CONFIG	RW	[0]: txfifo_less_l1 [1]: txfifo_less_l2 [2]: txfifo_more_l1 [3]: txfifo_more_l2 [4]: rxfifo_less_l1 [5]: rxfifo_less_l2 [6]: rxfifo_more_l1 [7]: rxfifo_more_l2	0x00
0x0c	I2S2_STIMER_TARGET0	RW	[7:0]: i2s2_stimer_target[7:0] i2s2 stimer for schedule	0x00
0x0d	I2S2_STIMER_TARGET1	RW	[7:0]: i2s2_stimer_target[15:8] i2s2 stimer for schedule	0x00
0x0e	I2S2_STIMER_TARGET2	RW	[7:0]: i2s2_stimer_target[23:16] i2s2 stimer for schedule	0x00
0x0f	I2S2_STIMER_TARGET3	RW	[7:0]: i2s2_stimer_target[31:24] i2s2 stimer for schedule	0x00
0x10	I2S_ALIGN_CFG	RW	[0]: i2s_align_en i2s align mode enable: 1'b0: disable 1'b1: enable [3:1]: i2s_align_ctrl i2s0&i2s1&i2s2 align enable independently [1] i2s0 [2] i2s1 [3]i2s2 [4]: i2s_align_mask i2s align mask 0: i2s align not mask 1: i2s align mask [5]: i2s_clk_sel 0: use self i2s clk 1: under align mode, use align i2s clk [6]: rx_dsp_start_sel fix pad2reg timing in i2s_slv mode	0x00

Address offset	Name	Type	Description	Default value
0x11	I2S2_TDM_CFG	RW	[1:0]: i2s2_tdm_rx_ch_num configure the number of i2s2 tdm rx channel 2'b00 2channel, 2'b01 4channel,2'b10 6channel,2'b11 8channel [3:2]: i2s2_tdm_tx_ch_num configure the number of i2s2 tdm tx channel 2'b00 2channel, 2'b01 4channel,2'b10 6channel,2'b11 8channel [5:4]: i2s2_tdm_mode i2s2 TDM mode: 2'b00: TDM_mode_a 2'b01: TDM_mode_b 2'b10: TDM_mode_c [7:6]: i2s2_tdm_slot 2'b00: 16 BCLK wide 2'b01: 24 BCLK wide 2'b10: 32 BCLK wide	0x00
0x14	I2S2_TIMER_TH0	RW	[7:0]: i2s2_timer_th0 i2s2 sys timer tick threshold 0 byte.	0xff
0x15	I2S2_TIMER_TH1	RW	[7:0]: i2s2_timer_th1 i2s2 sys timer tick threshold 1 byte.	0xff
0x16	I2S2_TIMER_TH2	RW	[7:0]: i2s2_timer_th2 i2s2 sys timer tick threshold 2 byte.	0xff
0x17	I2S2_TIMER_TH3	RW	[7:0]: i2s2_timer_th3 i2s2 sys timer tick threshold 3 byte.	0x03

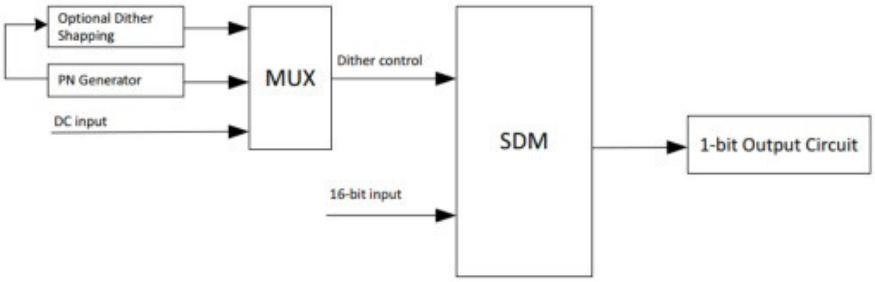
5.4 SDM (Sigma-Delta Modulation)

The SDM takes 16-bit audio data from SRAM and provides 1-bit modulated output. Only a simple passive filter network is needed to drive audio device directly.

Dither control can be added to the SDM to avoid spurs in output data. There are three dithering options: PN sequence, PN sequence with Shapping, and DC constant; only one type of input is allowed any time.

The following is the block diagram of SDM.

Figure 5-12 Audio SDM



5.4.1 Left Channel

For left channel,

(1) If selecting DC input for SDM, configure const_sel_l (ASCL0_SDM_BASE+0x0b[5]) to be 1, const_l (ASCL0_SDM_BASE+0xc~0xd) to configure constant value of input.

(2) If select PN generator for SDM, configuring const_sel_l to 0 means use PN generator, shap_l (ASCL0_SDM_BASE+0x03[5]) is the enable of dither shapping module. There are two PN generators to generate random number sequence, pn_sel_l (ASCL0_SDM_BASE+0x08[6:5]) is the enable for two PN generators.

- When the PN sequence is selected as input, const_sel_l and shap_l are configured as 0 and pn_sel_l is configured as 1.
- When PN sequence with Shapping is selected as input, const_sel_l is configured as 0 and shap_l and pn_sel_l are configured as 1.

When PN sequence or PN sequence with Shapping is selected, pn1_bits_l (ASCL0_SDM_BASE+0x08[4:0]) and pn2_bits_l (ASCL0_SDM_BASE+0x09[4:0]) determines the number of bits to be used in the PN1/PN2 generator (range 0-16).

5.4.2 Right Channel

For right channel,

(1) If selecting DC input for SDM, configure const_sel_r (ASCL0_SDM_BASE+0x0b[6]) as 1, const_r (ASCL0_SDM_BASE+0xe~0xf) configure input constant value;

(2) If selecting PN generator for SDM, configuring const_sel_r to 0 indicates that PN generator is used, shap_r (ASCL0_SDM_BASE+0x03[6]) is the enable of dither shapping module, there are two PN generators to generate random number sequence, pn_sel_r (ASCL0_SDM_BASE+0x09[6:5]) is the enable for two PN generators.

- When PN sequence is selected as input, const_sel_r and shap_r are configured as 0 and pn_sel_r is configured as 1.
- When PN sequence with Shapping is selected as input, const_sel_r is configured as 0 and shap_r and pn_sel_r are configured as 1.

When PN sequence or PN sequence with Shapping is selected, pn1_bits_r (ASCL0_SDM_BASE+0x0a[4:0]) and pn2_bits_r (ASCL0_SDM_BASE+0x0b[4:0]) determine the number of bits to be used in the PN1/PN2 generator (range 0-16).

5.5 ASCL

The ASCL (Asynchronous Sample rate Conversion with Linear interpolation) module performs sample rate conversion. It supports processing 16-bit data only. The input data is obtained from SRAM via DMA or MCU, and the output data is output to SDM/I2S at a specific sample rate.

For example, if the sample rate of input ASCL is SmpIn and the sample rate of output ASCL is SmpOut, ASCL can be configured according to the following formula:

$$\frac{SmpIn}{SmpOut} = \frac{step}{0x80000}$$

Linear interpolation or delay interpolation is used as shown below:

Figure 5-13 Linear interpolation

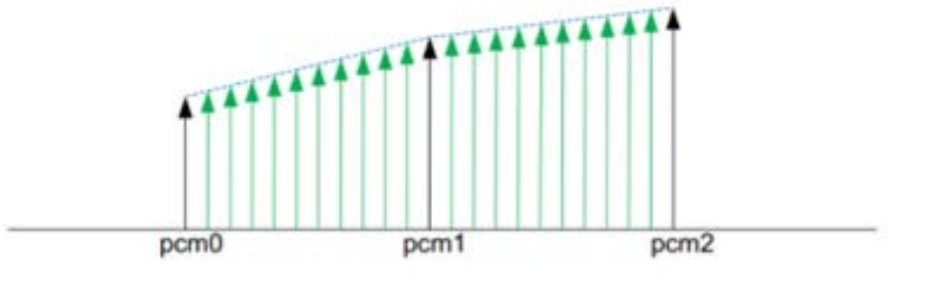
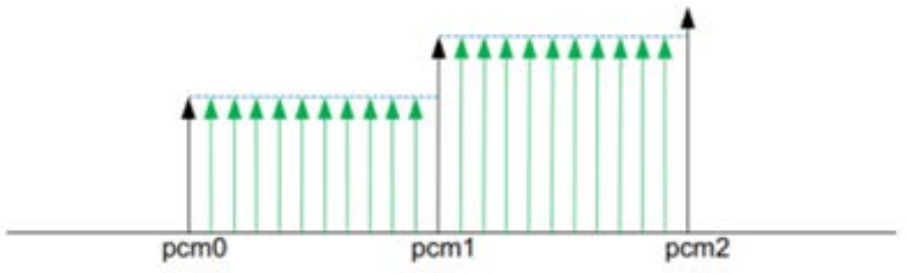


Figure 5-14 Delay interpolation



The Audio_arch has 3 ASCLs, ASCL0 non-SDM, ASCL1, and ASCL2, which have the same function and register offset address. The following is an example of ASCL0 non-SDM to illustrate the function configuration of ASCLs. The mono (ASCL0_SDM_BAEE+0x00[0]) is set to 1 to select the mono output, the hpf (ASCL0_SDM_BASE+0x00[7]) is HPF enable. The vol_ctrl (ASCL0_SDM_BAEE+0x02[6:0]) adjusts the volume. The line (ASCL0_SDM_BASE+0x03[2]) selects Linear interpolation or Delay interpolation. The step_i is configured through registers ASCL0_SDM_BASE+05[4:0]~07.

5.6 CODEC

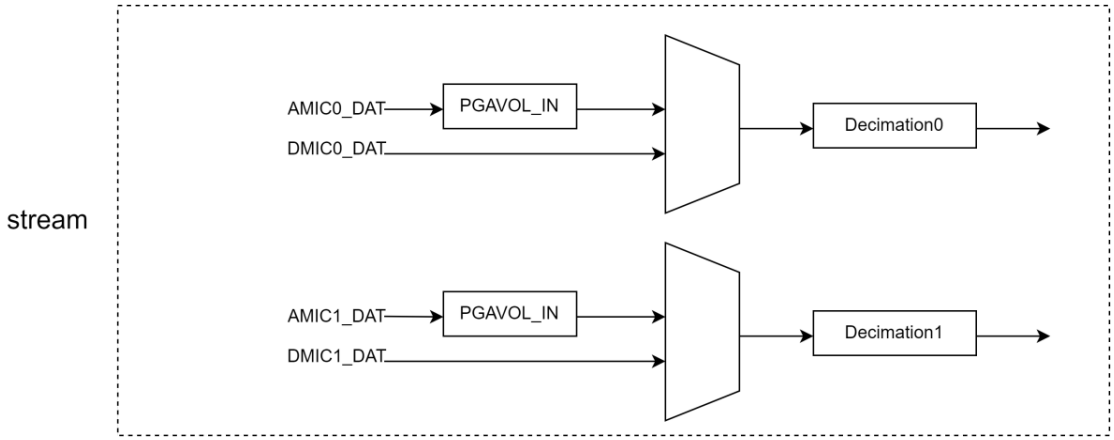
The CODEC features include:

- Supports 1-channel ADC
- Supports 2-mono or 1-stereo DMIC
- Supports sampling rate of 8k, 8.0214k, 11.0259k, 12k, 16k, 22.0588k, 24k, 32k, 44.118k, and 48k.
- MCLK=24MHz

5.6.1 CODEC Input Path

The CODEC input path is illustrated in figure below.

Figure 5-15 CODEC Input Path



The CODEC input consists of one group of stream (dec), and includes two data paths. The stream is utilized for dual AMIC input. To enable the AMIC input, the dec_en_ch (AUDIO_DFIFO_BASE+0x11[7:6]) should be set to 1 to enable the MIC interface module switch, and the mic_sel (AUDIO_DFIFO_BASE+0x12[7]) should be configured as 0 to select the input source as AMIC. The CODEC supports two DMIC inputs, configure mic_sel to 1.

The PGAVOL_IN is controlled via Ana_0x8d[6:4] with a range of 0 ~ 45.2 dB.

Table 5-4 PGA Gain for Different Configurations

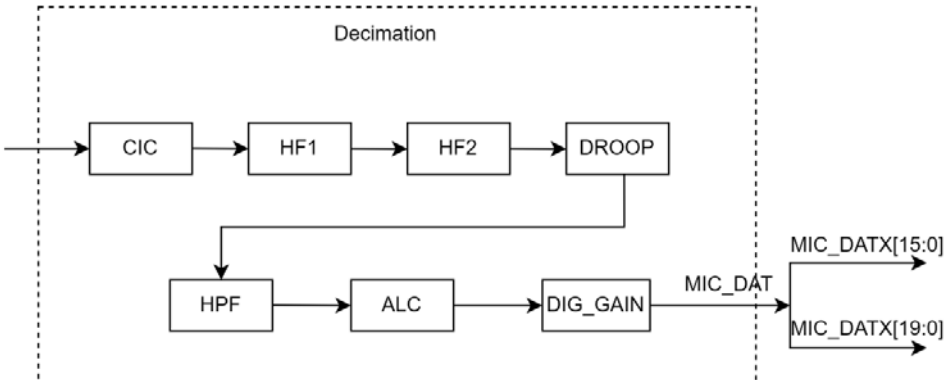
Code (4 bits)	PGA Gain (dB)
0000	45.2
0001	43.5
0010	42.1
0011	40.5
0100	39.1
0101	37.4
0110	36.0
0111	34.6
1000	33.0
1001	30.1
1010	27.0
1011	24.0
1100	21.0

Code (4 bits)	PGA Gain (dB)
1101	15.0
1110	9.0
1111	0

5.6.2 Decimation Data Path

The following figure shows the data path of the CODEC decimation of this chip. The sampling filter consists of a Cascaded Integrator-Comb (CIC) filter, two half-band filters, a compensation filter, and a High Pass Filter (HPF). The HPF serves the purpose of filtering out any DC bias generated by the input source. The switch is hpf_en(AUDIO_CODEC_BASE+0x00[0]). There are two formats for decimation output, one is 20-bit, the other is 16-bit, Set dec_ain0_mode(AUDIO_DFIFO_BASE+0x3b[0]), dec_ain1_mode(AUDIO_DFIFO_BASE+0x3b[2]) or dec_ain2_mode(AUDIO_DFIFO_BASE+0x3b[4]) to 0 to make output as 16-bit format.

Figure 5-16 Audio CODEC Decimation



The DIG_GAIN is a gain range of -48dB ~ +42dB, 6dB/step while controlling the digital gain of the left and right channels. The digital gain of the left and right channels of the dec is controlled by configuring dec_vol (AUDIO_DFIFO_BASE+0x12[5:0]).

Table 5-5 Digital Gain for Different Configurations

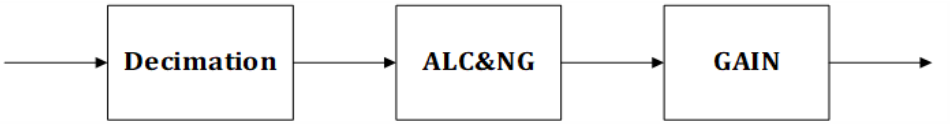
Gain	Coded
-48dB	0x00
-42dB	0x04
-36dB	0x08
-30dB	0x0c
-24dB	0x10
-18dB	0x14
-16dB	0x15
-12dB	0x18

Gain	Coded
-6dB	0x1c
0dB	0x20
+6dB	0x24
+12dB	0x28
+18dB	0x2c
+24dB	0x30
+30dB	0x34
+36dB	0x38
+42dB	0x3c

5.6.3 ALC

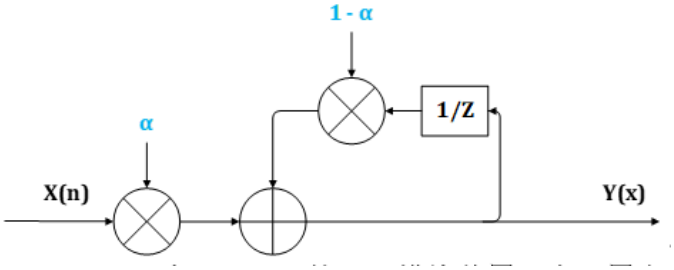
The Automatic Level Control (ALC) path is shown below and consists of a down-sampling filter, ALC & Noise Gate (NG), and adjustable gain.

Figure 5-17 Audio ALC Path



The structure of Average filter is shown below and is used to detect the envelope value of the input data. The parameter α=2^(-K1), K1 (AUDIO_CODEC_BASE+0x05[7:4]) can be adjusted to regulate the speed of detecting the following envelope.

Figure 5-18 Structure of Average Filter

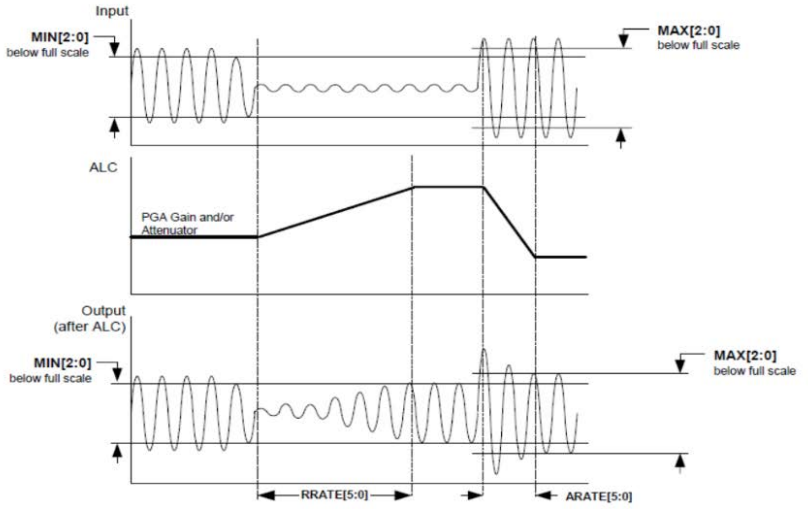


The ALC_SEL (AUDIO_CODEC_BASE+0x21[6:5]) switch is used to enable ALC for both the left and right paths. The ALC module compares the detected envelope value with the configured reference threshold ALCL (AUDIO_CODEC_BASE+0x20[3:0]). If the envelope value exceeds the reference threshold, the gain is reduced, and vice versa. To adjust the gain increase/decrease rate, the ATK (AUDIO_CODEC_BASE+0x22[3:0]) and DCY (AUDIO_CODEC_BASE+0x22[7:4]) parameters can be modified. These parameters control the rate at which the gain increases or decreases. Configure ALC_HLD (AUDIO_CODEC_BASE+0x21[3:0]) when the gain needs to be reduced to make the gain have a hysteresis time in reducing. For dynamic adjustable gain, if it exceeds

the MAXGAIN (AUDIO_CODEC_BASE+0x20[7:4]) value, it is capped at the maximum gain. Similarly, if the gain falls below MINGAIN (AUDIO_CODEC_BASE+0x24[2:0]), it is capped at the minimum gain.

Noise gating is used to prevent noise from entering during recording when there is no useful signal. The NGAT (AUDIO_CODEC_BASE+0x23[0]) switch controls the noise gating functionality, while the NGTH (AUDIO_CODEC_BASE+0x23[7:3]) parameter sets the noise threshold. When the detected input signal amplitude falls below the set threshold, it is considered noise. Configuring NGG (AUDIO_CODEC_BASE+0x23[2:1]) as 2'b01 activates signal soft mute in this case.

Figure 5-19 Signals before and after ALC



5.6.3.1 Sample Rate of CODEC Input

The master clock mclk of CODEC is divided from pll, the mclk frequency is 24 MHz (22.5792 MHz, 24.576 MHz, 33.8688 MHz, 36.864 MHz in some special cases). Configure codec clk div2 (AUDIO_CODEC_BASE+0x0a[6]) to 1 to make codec master clock 12 MHz (11.2896 MHz, 12.288 MHz, 16.9344 MHz, 18.432 MHz in some special cases). The input path of CODEC supports sampling rate of 8 kHz ~ 48 kHz, obtained by configuring dec_clk_sr (AUDIO_CODEC_BASE+0x0a[5:1]). Dmic_clk is dec_clk/2; amic clk clock can be selected by r_ck_sel(AUDIO_CODEC_BASE+0x00[1]), configure it to 1 to be dec_clk/2 in line with dmic, configure it to 0 to be dec_clk/12, configure r_if(AUDIO_DFIFO_BASE+0x11[1:0]) to 1, configure r_neg(AUDIO_DFIFO_BASE+0x11[5) to 1. When using DMIC, configure the sample rate according to Table 5-6 and Table 5-7 below. The sample rate of AMIC can be configured according to the four tables below, depending on the clock frequency required for the simulation.

NOTE:

- *Save the left channel data at DMIC rising edge, save the right channel data at DMIC falling edge.*

USB Mode

Configure (AUDIO_CODEC_BASE+0x0a[0]) to 1 to make the CODEC working in USB mode. The sample rate of the CODEC input is shown in the following two tables.

Table 5-6 Sample Rate of CODEC Input in USB mode - Part 1

MCLK (MHz)	Mclk_real (MHz)	CODEC input sample rate	Coded	dec_clk	dec_clk/2 (MHz)	dec_clk/12 (MHz)
24.000	12.000	8kHz (MCLK_real/(2*6*125))	00110, 00100	mclk/12	1	0.167
		11.025kHz (MCLK_real/(2*4*136))	11001	mclk/8	1.5	0.25
		12kHz (MCLK_real/(2*4*125))	01000	mclk/8	1.5	0.25
		16kHz (MCLK_real/(2*3*125))	01010	mclk/6	2	0.333
		22.05kHz (MCLK_real/(2*2*136))	11011	mclk/4	3	0.5
		24kHz (MCLK_real/(2*2*125))	11100	mclk/4	3	0.5
		32kHz (MCLK_real/(2*1.5*125))	01101	mclk/6	2	-
		44.1kHz (MCLK_real/(2*1*136))	11111	mclk/4	3	-
		48kHz (MCLK_real/(2*1*125))	11110	mclk/4	3	-

Table 5-7 Sample Rate of CODEC Input in USB mode - Part 2

MCLK (MHz)	Mclk_real (MHz)	CODEC input sample rate	Coded	dec_clk	dec_clk/2 (MHz)	dec_clk/12 (MHz)
24.000	12.000	32kHz (Mclk_real/(2*1.5*125))	01100	mclk/3	4	0.667
		44.1kHz (Mclk_real/(2*1*136))	10011 10001	mclk/2	6	1
		48kHz (Mclk_real/(2*1*125))	00000 00010	mclk/2	6	1

Normal Clock Mode

Configure (AUDIO_CODEC_BASE+0x0a[0]) to 0 to make the CODEC working in the normal clock mode. The sample rate of the CODEC input is shown in the following two tables.

Table 5-8 Sample Rate of CODEC Input in Normal mode - Part 1

MCLK (MHz)	Mclk_real (MHz)	CODEC input sample rate	Coded	dec_clk	dec_clk/2 (MHz)	dec_clk/12 (MHz)
24.000	12.000	32kHz (Mclk_real/(2*1.5*125))	01101	mclk/6	2	-
		44.1kHz (Mclk_real/(2*1*136))	11111	mclk/4	3	-
		48kHz (Mclk_real/(2*1*125))	11110	mclk/4	3	-

MCLK (MHz)	Mclk_real (MHz)	CODEC input sample rate	Coded	dec_clk	dec_clk/2 (MHz)	dec_clk/12 (MHz)
22.5792	11.2896	11.025kHz(Mclk_real/(2*4*128))	11000	mclk/8	1.4112	0.2352
		22.05kHz(Mclk_real/(2*2*128))	11010	mclk/4	2.8224	0.4704
24.576	12.288	8kHz(Mclk_real/(2*6*128))	00110, 00100	mclk/12	1.024	0.171
		12kHz(Mclk_real/(2*4*128))	01000	mclk/8	1.536	0.256
		16kHz(Mclk_real/(2*3*128))	01010	mclk/6	2.048	0.341
		24kHz(Mclk_real/(2*2*128))	11100	mclk/4	3.072	0.512
33.8688	16.9344	11.025kHz(Mclk_real/(2*6*128))	11001	mclk/8	2.1168	0.3528
		22.05kHz(Mclk_real/(2*3*128))	11011	mclk/4	4.2336	0.7056
36.864	18.432	8kHz(Mclk_real/(2*9*128))	00111, 00101	mclk/12	1.536	0.256
		12kHz(Mclk_real/(2*6*128))	01001	mclk/8	2.304	0.384
		16kHz(Mclk_real/(2*4.5*128))	01011	mclk/6	3.072	0.512
		24kHz(Mclk_real/(2*3*128))	11101	mclk/4	4.608	0.768

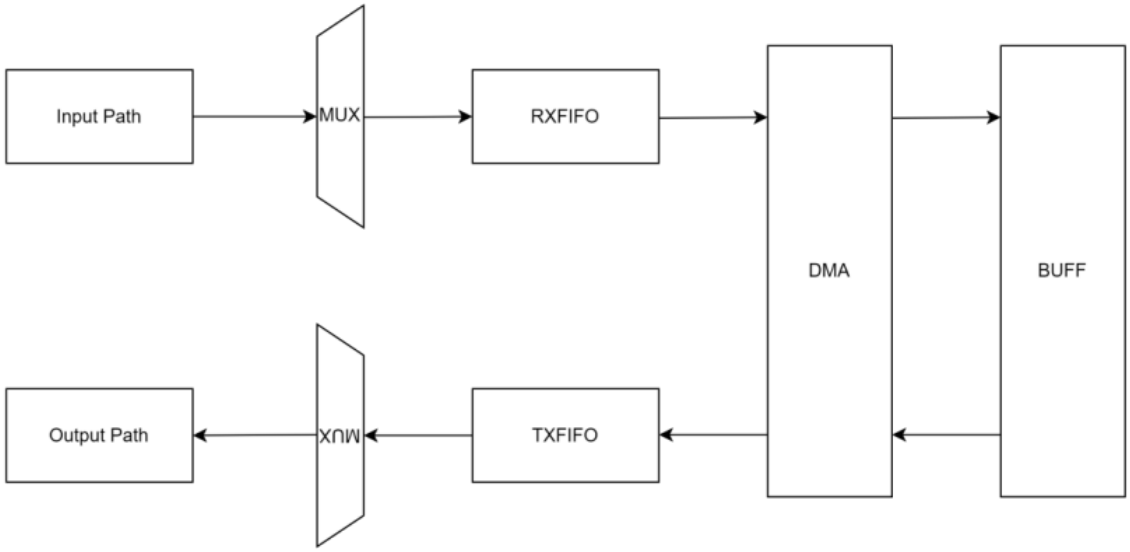
Table 5-9 Sample Rate of CODEC Input in Normal mode - Part 2

MCLK (MHz)	Mclk_real (MHz)	CODEC input sample rate	Coded	dec_clk	dec_clk/2 (MHz)	dec_clk/12 (MHz)
22.5792	11.2896	44.1kHz(Mclk_real/(2*1*128))	10010, 10000	mclk/2	5.6448	0.9408
24.576	12.288	48kHz (Mclk_real/(2*1*128))	00000 00010	mclk/2	6.144	1.024
33.8688	16.9344	44.1kHz(Mclk_real/(2*1.5*128))	10001	mclk/2	8.4672	1.4112
36.864	18.432	48kHz(Mclk_real/(2*1.5*128))	00001 00011	mclk/2	9.216	1.536

5.7 Audio FIFO

5.7.1 Introduction

Figure 5-20 Block Diagram of Audio FIFO

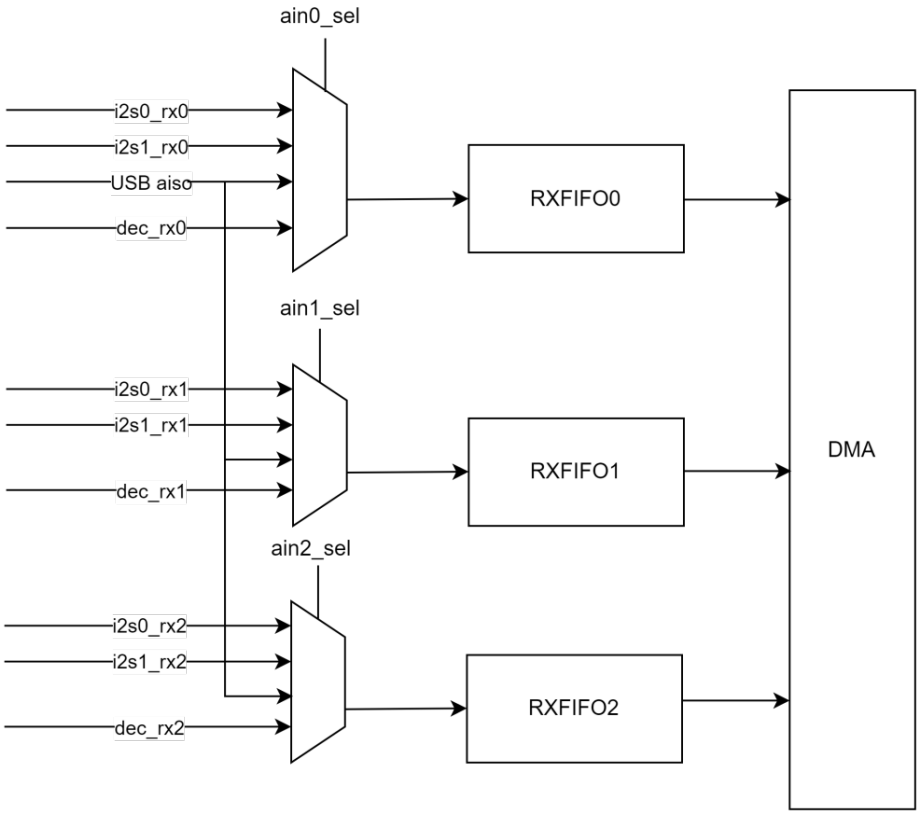


The above figure shows the general block diagram of Audio First-In-First-Out (FIFO), the input path includes I2S, USB, and CODEC. Data is written to the RX_FIFO buffer upon selection through a Multiplexer (MUX). When the source address is either 0x120000, 0x120040 or 0x120080, the DMA controller transfers data from FIFO to a user-specified SRAM location. Conversely, when the destination address is 0x120000, 0x120040 or 0x120080, the DMA controller moves data from SRAM to TXFIFO, then select output path according to the MUX. The output path includes I2S0, I2S1, I2S2 and USB. Taking 16bit stereo i2s0, FIFO0 as an example, configure ain0_sel (AUDIO_DFIFO_BASE+0x33[2:0]) to 3'b0, i2s0_ain0_mode (AUDIO_DFIFO_BASE+0x38[1:0]) to 2'b10, i2s0_aout_sel (AUDIO_DFIFO_BASE+0x42[1:0]) to 2'b10.

5.7.2 RXFIFO Selection

The following figure shows the input structure of RXFIFO. The RXFIFO input supports a variety of different source multiplexing channels, and the configuration registers can be used to select the channel and data format to meet different requirements.

Figure 5-21 RXFIFO Input



Select the input source for the RXFIFO0 path by configuring ain0_sel (AUDIO_DFIFO_BASE+0x33[2:0]). Select the input source for the RXFIFO1 path by configuring ain1_sel (AUDIO_DFIFO_BASE+0x34[2:0]). Select the input source for the RXFIFO2 path by configuring ain2_sel (AUDIO_DFIFO_BASE+0x34[6:4]).

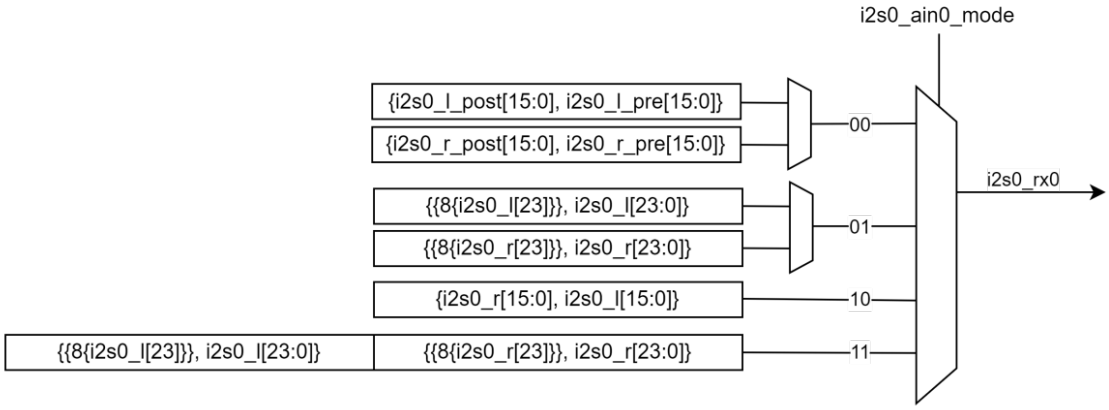
Table 5-10 Register of ain0_sel, ain1_sel and ain2_sel

Register	Description
ain0_sel	Rxfifo0 input source select: 3'b000: i2s0. 3'b001: i2s1. 3'b010: i2s2. 3'b011: dec. 3'b100: usb.
ain1_sel	Rxfifo1 input source select: 3'b000: i2s0. 3'b001: i2s1. 3'b010: i2s2. 3'b011: dec. 3'b100: usb.

Register	Description
ain2_sel	Rxfifo2 input source select: 3'b000: i2s0. 3'b001: i2s1. 3'b010: i2s2. 3'b011: dec. 3'b100: usb.

5.7.3 I2S Data Transfer in RXFIFO

Figure 5-22 I2S Data Transfer in RXFIFO



When I2S0 is used as the input source of RXFIFO 0, the data format of I2S0 is selected by configuring i2s0_ain0_mode (AUDIO_DFIFO_BASE+0x38[1:0]), as shown in the following table.

Table 5-11 Data format of i2s0_ain0_come

Register	Description
i2s0_ain0_come	Rxfifo0 input mode select: 2'b00: 16bit mono 2'b01: 20/24bit mono 2'b10: 16bit stereo 2'b11: 20/24bit stereo.

- When i2s0_ain0_mode selects 16bit mono, configure i2s0_leftdat_rxfifo_sel (AUDIO_DFIFO_BASE+0x35[1:0]) to 2'b0, it inputs two consecutive strokes of data from the left channel of i2s0 into RXFIFO0 as 32bit, and the first stroke is located at the lower 16bit of 32bit, and the second stroke is the higher 16bit of 32bit; configure i2s0_rightdat_rxfifo_sel (AUDIO_DFIFO_BASE+0x35[3:2]) to 2'b0, it inputs two consecutive strokes of data from the right channel of i2s0 into RXFIFO0.

- When i2s0_ain0_mode selects 20/24bit mono, at the same time configuring i2s0_leftdat_rxfifo_sel with 2'b0 expand the left channel data symbols into 32bit input into rxfifo0, similarly if i2s0_rightdat_rxfifo_sel is configured with 2'b0 expand the i2s0 right channel data symbols into 32bit input into rxfifo0;
- When i2s0_ain0_mode selects 16bit stereo, the 16bit data of the left and right channels are spliced into 32bit and input into FIFO, where the left channel data is the lower 16bit of 32bit;
- When i2s0_ain0_mode selects 20/24bit stereo, it inputs the data into FIFO in two strokes, first transmitting the 32bit data of the left channel symbol expansion, and then transmitting the 32bit data of the right channel symbol expansion.

When configuring i2s0_rx_2line_enable (I2S0_BASE+0x02[1]), i.e., rx 2line mode is selected, the data splicing of the input FIFO is equivalent to two consecutive data strokes. Taking 20/24bit stereo as an example, it is divided into four data strokes to be inputted into rxfifo0, which transmits the 32bit data of the symbol expansion of the left channel of i2s0, the 32bit data of the symbol expansion of the right channel of i2s0, then the 32bit data of the symbol expansion of the left channel of i2s1, and finally the 32bit data of the symbol expansion of the right channel of i2s1. See the following table for other types:

Table 5-12 RX 2line mode data type

Code	Mode	Data Type
i2s0_ain0_mode==2'b00 & i2s0_leftdat_rxfifo_sel==2'b00	16bit mono	{i2s0_l_post_line0[15:0],i2s0_l_pre_line0[15:0]} {i2s0_l_post_line1[15:0],i2s0_l_pre_line1[15:0]}
i2s0_ain0_mode==2'b00 & i2s0_rightdat_rxfifo_sel==2'b00	16bit mono	{i2s0_r_post_line0[15:0],i2s0_r_pre_line0[15:0]} {i2s0_r_post_line1[15:0],i2s0_r_pre_line1[15:0]}
i2s0_ain0_mode==2'b01 & i2s0_leftdat_rxfifo_sel==2'b00	20/24bit mono	{{8{i2s0_l_line0[23]}},i2s0_l_line0[23:0]} {{8{i2s0_l_line1[23]}},i2s0_l_line1[23:0]}
i2s0_ain0_mode==2'b01 & i2s0_rightdat_rxfifo_sel==2'b00	20/24bit mono	{{8{i2s0_r_line0[23]}},i2s0_r_line0[23:0]} {{8{i2s0_r_line1[23]}},i2s0_r_line1[23:0]}
i2s0_ain0_mode==2'b10	16bit stereo	{i2s0_r_line0[15:0],i2s0_l_line0[15:0]} {i2s0_r_line1[15:0],i2s0_l_line1[15:0]}
i2s0_ain0_mode==2'b11	20/24bit stereo	{{8{i2s0_l_line0[23]}},i2s0_l_line0[23:0]} {{8{i2s0_r_line0[23]}},i2s0_r_line0[23:0]} {{8{i2s0_l_line1[23]}},i2s0_l_line1[23:0]} {{8{i2s0_r_line1[23]}},i2s0_r_line1[23:0]}

When I2S1 is used as the input source of RXFIFO 0, the data format of I2S1 is selected by configuring i2s1_ain0_mode (AUDIO_DFIFO_BASE+0x39[1:0]), as shown in the above table.

When I2S2 is used as the input source of RXFIFO 0, the data format of I2S2 is selected by configuring i2s2_ain0_mode (AUDIO_DFIFO_BASE+0x3a[1:0]), as shown in the above table.

Table 5-13 Data format of i2s1_ain0_come and i2s2_ain0_come

Register	Description
i2s1_ain0_come	rxfifo0 input mode select: 2'b00: 16bit mono 2'b01: 20/24bit mono 2'b10: 16bit stereo 2'b11: 20/24bit stereo.
i2s2_ain0_come	rxfifo0 input mode select: 2'b00: 16bit mono 2'b01: 20/24bit mono 2'b10: 16bit stereo 2'b11: 20/24bit stereo.

In addition, when the I2S data is used as the input source of RXFIFO1 is basically the same as RXFIFO0. The difference is that rxfifo1 is fixedly selected when i2s1 and i2s2 sync mode, while rxfifo0 is fixedly selected when i2s0, i2s1 sync mode and i2s0, i2s1,i2s2 sync mode.

The data format in sync mode is shown in the table below. Take i2s1, i2s2 sync mode 20/24bit dual channel as an example, configure ain1_sel (AUDIO_DFIFO_BASE+0x34[2:0]) to 3'b001. When register i2s_sync_mode (AUDIO_DFIFO_BASE+0x37 [3:0]) is 4'b0111 i.e. i2s1, i2s2 sync mode 20/24bit dual channel mode, four write rxfifo behaviors are initiated, the first transmits 32bit data after i2s2 left channel symbol expansion, the second transmits 32bit data after i2s2 right channel symbol expansion, the third transfers the 32bit data after i2s1 left channel symbol expansion, and the fourth stroke transfers the 32bit data after i2s1 right channel symbol expansion.

Table 5-14 Data format in Sync Mode

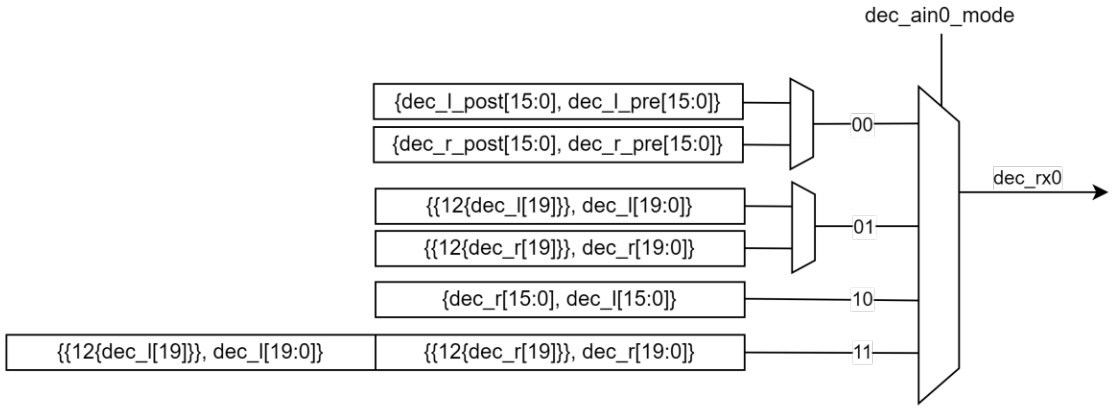
Code	Sync mode	Data Type
4'b0100	i2s1, i2s2 sync mode 16bit mono	{i2s1_l[15:0], i2s2_l[15:0]}
4'b0101	i2s1, i2s2 sync mode 20/24bit mono	i2s2_l[a] i2s1_l
4'b0110	i2s1, i2s2 sync mode 16bit stereo	{i2s2_r[15:0],i2s2_l[15:0]} {i2s1_r[15:0],i2s1_l[15:0]}
4'b0111	i2s1, i2s2 sync mode 20/24bit stereo	i2s2_l i2s2_r i2s1_l i2s1_r
4'b1000	i2s0, i2s1 sync mode 16bit mono	{i2s0_l[15:0], i2s1_l[15:0]}

Code	Sync mode	Data Type
4'b1001	i2s0, i2s1 sync mode 20/24bit mono	i2s1_l i2s0_l
4'b1010	i2s0, i2s1 sync mode 16bit stereo	{i2s1_r[15:0],i2s1_l[15:0]} {i2s0_r[15:0],i2s0_l[15:0]}
4'b1011	i2s0, i2s1 sync mode 20/24bit stereo	i2s1_l i2s1_r i2s0_l i2s0_r
4'b1101	i2s0, i2s1,i2s2 sync mode 20/24bit mono	i2s2_l i2s1_l i2s0_l
4'b1110	i2s0, i2s1,i2s2 sync mode16bit stereo	{i2s2_r[15:0],i2s2_l[15:0]} {i2s1_r[15:0],i2s1_l[15:0]} {i2s0_r[15:0],i2s0_l[15:0]}
4'b1111	i2s0, i2s1,i2s2 sync mode 20/24bit stereo	i2s2_l i2s2_r i2s1_l i2s1_r i2s0_l i2s0_r

a. In the data type column, one row of data means one stroke of data.

5.7.4 CODEC Data Transfer in RXFIFO

Figure 5-23 CODEC Data Transfer in RXFIFO



When dec is used as the input source of RXFIFO 0, the data format of stream0 is selected by configuring dec0_ain0_mode (AUDIO_DFIFO_BASE+0x3b[1:0]) as shown in the following table.

Table 5-15 Data format of dec0_ain0_come and dec1_ain0_come

Register	Description
dec0_ain0_come	Rxfifo0 input mode select: 2'b00: 16bit mono 2'b01: 20bit mono 2'b10: 16bit stereo 2'b11: 20bit stereo.

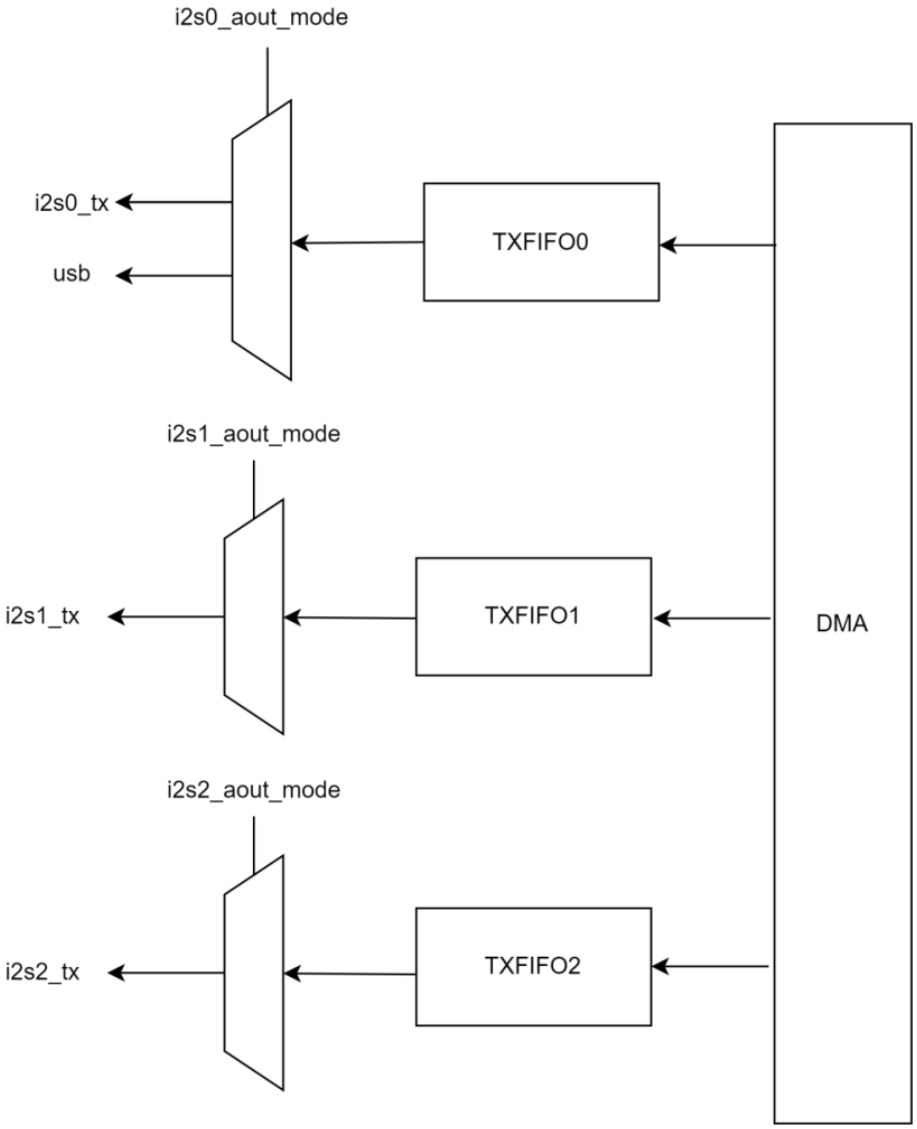
- When dec_ain0_mode selects 16bit mono, and configure the dec_leftdat_rxfifo_sel (AUDIO_DFIFO_BASE+0x35[5:4]) as 2'b0 inputs two consecutive strokes of data from the left channel of the dec as 32bit into rxfifo0 and the first stroke is in the lower 16bit of 32bit and the second stroke is the higher 16bit of 32bit. If dec_rightdat_rxfifo_sel (AUDIO_DFIFO_BASE+0x35[7:6]) is configured 2'b0 scrambles the two consecutive strokes of data from the dec right channel into the 32bit inputs into rxfifo0.
- When dec_ain0_mode selects 20/24bit mono, and configure dec_leftdat_rxfifo_sel as 2'b0 to expand the left channel data symbols into 32bit to input into rxfifo0, similarly if configuring dec_rightdat_rxfifo_sel as 2'b0 it expands the dec right channel data symbols into 32bit input to rxfifo0;
- When dec_ain0_mode selects 16bit stereo it splices the 16bit data of the left and right channels into 32bit and input it into rxfifo0, where the left channel data is the lower 16bit of 32bit;
- When dec_ain0_mode selects 20/24bit stereo, the data is written into rxfifo0 in two strokes, first transferring the 32bit data of the left channel's symbol expansion, and then transferring the 32bit data of the right channel's symbol expansion.

In addition, when the CODEC data is used as the input source for rxfifo1 and rxfifo2, it is consistent with rxfifo0.

5.7.5 TXFIFO

The following figure shows the output structure of txfifo in normal mode. The txfifo output in normal mode has fixed channels. The i2s0 and usb are fixed to output from txfifo0, i2s1 is fixed to output from txfifo1, and i2s2 is fixed to output from txfifo2.

Figure 5-24 TXFIFO Output Structure



When in i2s0,i2s1,i2s Sync mode and i2s0,i2s1 sync mode, the data is read from txfifo0 to i2s0,i2s1 and i2s2; when i2s1,i2s2 sync mode, the data is read from txfifo1 to i2s1 and i2s2.

When i2s0_2fifo mode (AUDIO_DFIFO_BASE+0x4b[0]) is configured, the left data of i2s0 is output from txfifo0 and the right data of i2s0 is output from txfifo1; when i2s1_2fifo mode (AUDIO_DFIFO_BASE+0x4b[1]) is configured, the left data of i2s1 is output from txfif01 and the right data of i2s1 is output from txfifo2; when i2s2_2fifo mode (AUDIO_DFIFO_BASE+0x4b[2]) is configured, the left data of i2s2 is output from txfif01 and the right data of i2s2 is output from txfifo2.

5.8 Audio MUX

The AUDIO_MUX is divided into ATX_MUX and ARX_MUX. In which, ATX_MUX is the route from txfifo to each module and ARX_MUX is the route from each module to rxfifo.

The output modules of ATX_MUX are SDML, SDMR, I2S0, I2S1, I2S2, USB_TX.

The ARX_MUX input modules are I2S0, I2S1, I2S2, CODEC, USB_RX

For the details on how to route and how to select data format, please contact Telink FAE.

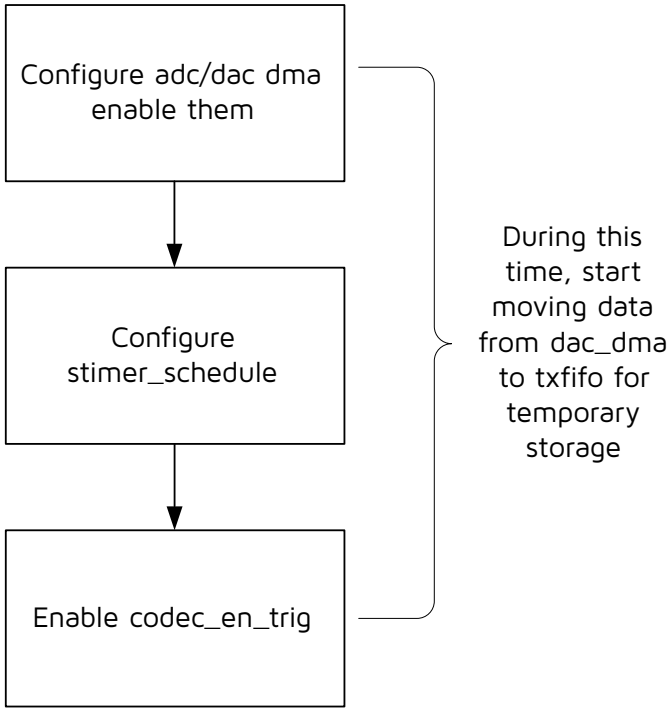
5.9 Schedule Mode

The schedule mode indicates that the relevant module can be enabled at a set point in time.

The modules that support schedule mode are Codec, I2S0, I2S1 and I2S2.

The software configuration sequence is shown as below.

Figure 5-25 CODEC Data Transfer in RXFIFO



Taking I2S0 as an example, schedule mode requires additional registers to be configured:

- system_timer needs to be enabled first.
- I2s0_schedule_en: I2S0_BASE+0x03[4]
- I2S0_stimer_target: I2S0_BASE+0x0c~0x0f (need to configure by word)

5.10 Audio Interrupt

There are two Audio interrupts: audio FIFO interrupt and audio FIFO DMA interrupt.

5.10.1 Audio FIFO Interrupt

Audio fifo interrupt has 6 interrupts: txfifo0_irq, txfifo1_irq, txfifo2_irq, rxfifo0_irq, rxfifo1_irq, rxfifo2_irq

For txfifo*_irq, the interrupt is triggered when the total number of dma write fifo's reaches the set threshold.

For rxfifo*_irq, the interrupt is triggered when the total number of dma read fifo's reaches the set threshold.

Take txfifo0_irq as an example, the write counter register of enable txfifo0 is AUDIO_DFIFO_BASE+0x3f[4], and you can set the initial value of the write counter (AUDIO_DFIFO_BASE+0x14~0x15), and the address of the txfifo0_threshold register is AUDIO_ DFIFO_BASE+0x62~0x63, the status register of txfifo0_irq is AUDIO_DFIFO_BASE+0x68[3], writing 1 to AUDIO_DFIFO_BASE+0x69[4] means clear txfifo0_irq, the interrupt enable register of txfifo0_irq is AUDIO_DFIFO_BASE+0x6a[4].

5.10.2 Audio FIFO DMA Interrupt

Audio fifo dma interrupt has 6 interrupts: txfifo0_dma_irq, txfifo1_dma_irq, txfifo2_dma_irq, rxfifo0_dma_irq, rxfifo1_dma_irq, rxfifo2_dma_irq.

For txfifo*_dma_irq, interrupt is triggered when txfifo_num < txfifo_threshold, then MCU can write data to txfifo.

For rxfifo*_dma_irq, interrupt is triggered when rxfifo_num > rxfifo_threshold, at this time MCU can read Data from rxfifo.

5.11 Audio Register Description

The audio FIFO related registers are listed in the table below. The base address for the following registers is 0x80141000.

Table 5-16 Audio FIFO Related Registers

Address Offset	Name	Type	Description	Reset Value
0x02	RXFIFO0_MAXL	RW	rxfifo0_max[7:0] [7:0] RXFIFO0 wptr max low byte	0x00
0x03	RXFIFO0_MAXH	RW	rxfifo0_max[15:8] [7:0] RXFIFO0 wptr max high byte	0x00
0x06	RXFIFO1_MAXL	RW	rxfifo1_max[7:0] [7:0] RXFIFO1 wptr max low byte	0x00
0x07	RXFIFO1_MAXH	RW	rxfifo1_max]15:8] [7:0] RXFIFO1 wptr max high byte	0x00
0x0a	RXFIFO2_MAXL	RW	rxfifo2_max[7:0] [7:0] RXFIFO2 wptr max low byte	0x00
0x0b	RXFIFO2_MAXH	RW	rxfifo2_max[15:8] [7:0] RXFIFO2 wptr max high byte	0x00

Address Offset	Name	Type	Description	Reset Value
0x10	RXFIFO_EN	RW	[0] rxfifo0_ainen, enable audio input of rxfifo0 [1] rxfifo1_ainen, enable audio input of rxfifo1 [2] rxfifo2_ainen, enable audio input of rxfifo2 [6:4] rxfifo_irq_en, fifo interrupt enable,[4]:fifo0 enable;[5]:fifo1 enable;[6]:fifo2 enable	0x00
0x11	CODEC_CFG	RW	[1:0] r_if, change sampling point, set 2'b01 [5] r_neg [7:6] r_ch_en, mic channel enable;[6]:mic l channel enable;[7]:mic r channel enable	0x01
0x12	CODEC_VOL	RW	[5:0] dec_vol, mic vol control: 6'h00:-48db 6'h04:-42db 6'h08:-36db 6'h0c:-30db 6'h10:-24db 6'h14:-18db 6'h18:-12db 6'h1c:-6db 6'h20:0db 6'h24:6db 6'h28:12db 6'h2c:18db 6'h30:24db 6'h34:30db 6'h38:36db 6'h3c:42db [7] mic_sel, 0:amic 1:dmic	0x20
0x14	TXFIFO0_RPTR_L	W	txfifo0_rptr[7:0] [7:0] txfifo0 read ptr low byte	0x00
0x15	TXFIFO0_RPTR_H	W	txfifo0_rptr[15:8] [7:0] txfifo0 read ptr high byte	0x00
0x16	RXFIFO0_WPTR_L	W	rxfifo0_wptr[7:0] [7:0] rxfifo0 write ptr low byte	0x00
0x17	RXFIFO0_WPTR_H	W	rxfifo0_wptr[15:8] [7:0] rxfifo0 write ptr high byte	0x00
0x18	TXFIFO1_RPTR_L	W	txfifo1_rptr[7:0] [7:0] txfifo1 read ptr low byte	0x00
0x19	TXFIFO1_RPTR_H	W	txfifo1_rptr[15:8] [7:0] txfifo1 read ptr high byte	0x00
0x1a	RXFIFO1_WPTR_L	W	rxfifo1_wptr[7:0] [7:0] rxfifo1 write ptr low byte	0x00
0x1b	RXFIFO1_WPTR_H	W	rxfifo1_wptr[15:8] [7:0] rxfifo1 write ptr high byte	0x00

Address Offset	Name	Type	Description	Reset Value
0x1c	TXFIFO2_RPTR_L	W	txfifo2_rptr[7:0] [7:0] txfifo2 read ptr low byte	0x00
0x1d	TXFIFO2_RPTR_H	W	txfifo2_rptr[15:8] [7:0] txfifo2 read ptr high byte	0x00
0x1e	RXFIFO2_WPTR_L	W	rxfifo2_wptr[7:0] [7:0] rxfifo2 write ptr low byte	0x00
0x1f	RXFIFO2_WPTR_H	W	rxfifo2_wptr[15:8] [7:0] rxfifo2 write ptr high byte	0x00
0x20	RXFIFO0_NUM	R	[3:0] rxfifo0_num, rxfifo0 data number	0x00
0x24	RXFIFO1_NUM	R	[3:0] rxfifo1_num, rxfifo1 data number	0x00
0x28	RXFIFO2_NUM	R	[3:0] rxfifo2_num, rxfifo2 data number	0x00
0x33	FIFO0IN_SEL	RW	[2:0] ain0_sel, rxfifo0 input sel;3'd0:i2s0;3'd1:i2s1;3'd2:i2s2;3'd3:codec;3'd4:usb	0x00
0x34	FIFO12IN_SEL	RW	[2:0] ain1_sel, rxfifo1 input sel;3'd0:i2s0;3'd1:i2s1;3'd2:i2s2;3'd3:codec;3'd4:usb [6:4] ain2_sel, rxfifo2 input sel;3'd0:i2s0;3'd1:i2s1;3'd2:i2s2;3'd3:codec;3'd4:usb	0x00
0x35	RXFIFO_SEL	RW	[1:0] i2s0_leftdat_rxfifo_sel, mono mode left fifo sel.2'b0:fifo0;2'd1:fifo1;2'd2:fifo2 [3:2] i2s0_rightdat_rxfifo_sel, mono mode right fifo sel.2'b0:fifo0;2'd1:fifo1;2'd2:fifo2 [5:4] dec_leftdat_rxfifo_sel, mono mode left fifo sel for dec.2'b0:fifo0;2'd1:fifo1;2'd2:fifo2 [7:6] dec_rightdat_rxfifo_sel, mono mode right fifo sel for dec.2'b0:fifo0;2'd1:fifo1;2'd2:fifo2	0x3
0x3b	DEC_AIN_MODE	RW	[1:0] dec_ain0_mode, dec fifo0 mode sel.2'd0:16bit mono;2'd1:20bit mono;2'd2:16bit stereo;2'd3:20bit stereo; [3:2] dec_ain1_mode, dec fifo1 mode sel.2'd0:16bit mono;2'd1:20bit mono;2'd2:16bit stereo;2'd3:20bit stereo; [5:4] dec_ain2_mode, dec fifo2 mode sel.2'd0:16bit mono;2'd1:20bit mono;2'd2:16bit stereo;2'd3:20bit stereo;	0x3

Address Offset	Name	Type	Description	Reset Value
0x3c	RXFIFO0_TRIG_NUM	RW	[3:0] rxfifo0_trig_num, rxfifo0 trig number	0x1
0x3d	RXFIFO1_TRIG_NUM	RW	[3:0] rxfifo1_trig_num, rxfifo1 trig number	0x1
0x3e	RXFIFO2_TRIG_NUM	RW	[3:0] rxfifo2_trig_num, rxfifo2 trig number	0x1
0x3f	RX_WPTR_EN	RW	[0] rx0_wptr_en, rx0 wptr enable [1] rx1_wptr_en, rx1 wptr enable [2] rx2_wptr_en, rx2 wptr enable [4] tx0_rptr_en, tx0 rptr enable [5] tx1_rptr_en, tx1 rptr enable [6] tx2_rptr_en, tx2 rptr enable	0x00
0x40	FIFO_CLR	RW	[0] rxfifo0_clr, write a rxfifo0 clr pulse in pclk domain, read rxfifo0 clr in hclk domain [1] rxfifo1_clr, write a rxfifo1 clr pulse in pclk domain, read rxfifo1 clr in hclk domain [2] rxfifo2_clr, write a rxfifo2 clr pulse in pclk domain, read rxfifo2 clr in hclk domain [4] txfifo0_clr, write a txfifo0 clr pulse in pclk domain, read txfifo0 clr in hclk domain [5] txfifo1_clr, write a txfifo1 clr pulse in pclk domain, read txfifo1 clr in hclk domain [6] txfifo2_clr, write a txfifo2 clr pulse in pclk domain, read txfifo2 clr in hclk domain	0x00
0x41	FIFO_OUTEN	RW	[0] txfifo0_aouten, enable audio output of rxfifo0 [1] txfifo1_aouten, enable audio output of rxfifo1 [2] txfifo2_aouten, enable audio output of rxfifo2 [6:4] txfifo_irq_en, txfifo interrupt enable.[4]:fifo0 enable;[5]:fifo1 enable;[6]:fifo2 enable	0x00
0x43	TXFIFO0_TRIG_NUM	RW	[3:0] txfifo0_trig_num, txfifo0 trig number	0x1
0x44	TXFIFO1_TRIG_NUM	RW	[3:0] txfifo1_trig_num, txfifo1 trig number	0x1

Address Offset	Name	Type	Description	Reset Value
0x45	TXFIFO2_TRIG_NUM	RW	[3:0] txfifo2_trig_num, txfifo2 trig number	0x1
0x46	FIFO_ST	R	[0] rxfifo0_overrun, write when rxfifo0 full [1] rxfifo1_overrun, write when rxfifo1 full [2] rxfifo2_overrun, write when rxfifo2 full [4] txfifo0_underrun, read when txfifo0 empty [5] txfifo1_underrun, read when txfifo1 empty [6] txfifo2_underrun, read when txfifo2 empty	0x0
0x47	I2S2_TDM_MODE_SEL	RW	[1:0] i2s2_tdm_ain_mode, [0]:16bit TDM enable; [1]:20/24bit TDM enable [3:2] i2s2_tdm_aout_mode, [2]:16bit TDM enable; [3]:20/24bit TDM enable	0x0
0x4b	I2S_2FIFO_MODE	RW	[0] i2s0_2fifo_mode, i2s0_2fifo_mode enable [1] i2s1_2fifo_mode, i2s1_2fifo_mode enable [2] i2s2_2fifo_mode, i2s2_2fifo_mode enable	0x0
0x4c	ascl0_aful_aemp	RW	[3:0] txfifo0_numl, txfifo0 min num for ascl empty [7:4] txfifo0_numh, txfifo0 max num for ascl empty	0x80
0x4d	ascl1_aful_aemp	RW	[3:0] txfifo1_numl, txfifo1 min num for ascl empty [7:4] txfifo1_numh, txfifo1 max num for ascl empty	0x80
0x4e	ascl2_aful_aemp	RW	[3:0] txfifo2_numl, txfifo2 min num for ascl empty [7:4] txfifo2_numh, txfifo2 max num for ascl empty	0x80
0x50	TXFIFO0_NUM	R	[3:0] txfifo0_num, txfifo0 data number	0x00
0x51	TXFIFO1_NUM	R	[3:0] txfifo1_num, txfifo1 data number	0x00
0x52	TXFIFO2_NUM	R	[3:0] txfifo2_num, txfifo2 data number	0x00
0x56	TXFIFO0_L	RW	txfifo0_h[7:0] [7:0] TXFIFO0 low level when irq	0x00
0x57	TXFIFO0_H	RW	txfifo0_h[15:8] [7:0] TXFIFO0 high level when irq	0x00
0x58	TXFIFO1_L	RW	txfifo1_h[7:0] [7:0] TXFIFO1 low level when irq	0x00

Address Offset	Name	Type	Description	Reset Value
0x59	TXFIFO1_H	RW	txfifo1_h[15:8] [7:0] TXFIFO1 high level when irq	0x00
0x5a	TXFIFO2_L	RW	txfifo2_h[7:0] [7:0] TXFIFO2 low level when irq	0x00
0x5b	TXFIFO2_H	RW	txfifo2_h[15:8] [7:0] TXFIFO2 high level when irq	0x00
0x5c	RXFIFO0_L	RW	rxfifo0_h[7:0] [7:0] RXFIFO0 low level when irq	0x00
0x5d	RXFIFO0_H	RW	rxfifo0_h[15:8] [7:0] RXFIFO0 high level when irq	0x00
0x5e	RXFIFO1_L	RW	rxfifo1_h[7:0] [7:0] RXFIFO1 low level when irq	0x00
0x5f	RXFIFO1_H	RW	rxfifo1_h[15:8] [7:0] RXFIFO1 high level when irq	0x00
0x60	RXFIFO2_L	RW	rxfifo2_h[7:0] [7:0] RXFIFO0 low level when irq	0x00
0x61	RXFIFO2_H	RW	rxfifo2_h[15:8] [7:0] RXFIFO0 high level when irq	0x00
0x62	TXFIFO0_MAXL	RW	txfifo0_max[7:0] [7:0] txfifo0 max num low byte for rptr	0x00
0x63	TXFIFO0_MAXH	RW	txfifo0_max[15:8] [7:0] txfifo0 max num high byte for rptr	0x00
0x64	TXFIFO1_MAXL	RW	txfifo1_max[7:0] [7:0] txfifo1 max num low byte for rptr	0x00
0x65	TXFIFO1_MAXH	RW	txfifo1_max[15:8] [7:0] txfifo1 max num high byte for rptr	0x00
0x66	TXFIFO2_MAXL	RW	txfifo2_max[7:0] [7:0] txfifo2 max num low byte for rptr	0x00
0x67	TXFIFO2_MAXH	RW	txfifo2_max[15:8] [7:0] txfifo2 max num high byte for rptr	0x00

Address Offset	Name	Type	Description	Reset Value
0x68	FIFO_IRQ	W1C	[0] rxfifo0_irq, rxfifo0_irq [1] rxfifo1_irq, rxfifo1_irq [2] rxfifo2_irq, rxfifo2_irq [3] txfifo0_irq, txfifo0_irq [4] txfifo1_irq, txfifo1_irq [5] txfifo2_irq, txfifo2_irq	0x00
0x69	FIFO_TH_IRQ	W1C	[0] rxfifo0_th_irq, rxfifo0_th_irq [1] rxfifo1_th_irq, rxfifo1_th_irq [2] rxfifo2_th_irq, rxfifo2_th_irq [3] txfifo0_th_irq, txfifo0_th_irq [4] txfifo1_th_irq, txfifo1_th_irq [5] txfifo2_th_irq, txfifo2_th_irq	0x38
0x6a	FIFO_TH_EN	RW	[0] rxfifo0_th_irq_en, rxfifo0_th_irq enable [1] rxfifo1_th_irq_en, rxfifo1_th_irq enable [2] rxfifo2_th_irq_en, rxfifo2_th_irq enable [3] txfifo0_th_irq_en, txfifo0_th_irq enable [4] txfifo1_th_irq_en, txfifo1_th_irq enable [5] txfifo2_th_irq_en, txfifo2_th_irq enable	0x00
0x6b	ACLK_DBG	RW	[0] aclk_dbg_open, 1'b1:dsm_output_clk=aclk.for debug. [1] i2s0_clk_dbg, 1'b1:dsm_output_clk=i2s0.for debug. [2] i2s1_clk_dbg, 1'b1:dsm_output_clk=i2s1.for debug. [3] i2s2_clk_dbg, 1'b1:dsm_output_clk=i2s2.for debug. [4] adc_clk6m_dbg, 1'b1:sdm output clk=6M.for debug [5] adc_clk1m_dbg, 1'b1:sdm output clk=1M.for debug [6] codec_clk_dbg, 1'b1:dsm_output_clk=codec clk.for debug.	0x00

The CODEC related registers are listed in the table below. The base address for the following registers is 0x80141080.

Table 5-17 CODEC Related Registers

Address Offset	Name	Type	Description	Reset Value
0X00	CODEC_CFG1	RW	[0] hpf_en, high-pass filter enable [1] r_ck_sel, 1'b1:dmic clk;1'b0:1M; [2] r_sft_zc, cic shift,1'b1:left shift:8;1'b0:left shift:9 [3] sram_ce_manual, for sram enable,1'b1:sram always active; [4] dati_soft_mute, mic input softmute enable [5] dato_soft_mute, alc output data softmute enable	0x05
0X05	CODEC_K1	RW	[3:0] reserved [7:4] k1, coef for ALC	0x5e
0x0a	CODEC_CLKCFG	RW	[0] clk_usb, 1'b1:clk usb mode. [5:1] clk_sr, sample rate.5'b00110:8k; 5'b10111:8.0214k; 5'b11001:11.0259; 5'b01000:12k; 5'b01010:16k; 5'b11011:22.0588; 5'b11100:24k; 5'b01100:32k; 5'b01101:32k@2Mhz;5'b10001:44.118; 5'b00000:48k;5'b11110:48k@3MHz;5'b11111:44.118k@3MHz. [6] clk_div2, clk div2 enable [7] clk_en, clk enable	0x00
0x0b	CODEC_RST	RW	[0] en_dec, codec rst disable	0x00
0x20	CODEC_ALC	RW	[3:0] alcl, decide the ref:1.5dB/step.4'b0000:28dBFS...4'b1011:-12dBFS(default)...4'b1111:-6dBFS. [6:4] maxgain, maxgain for ALC:6dB/step;3'b000:-72dB...3'b111:-30dB;	0x7b
0x21	CODEC_ALCSEL	RW	[3:0] hld, hold time:x2 step.4'b0000:0ms;4'b0001:2.67ms;4'b0010:5.33ms...4'b1111:43.691 [6:5] alcsel, alc select.2'b00:no alc;2'b01:right only;2'b10:left only;2'b11:stereo	0x00
0x22	CODEC_ATK	RW	[3:0] atk, Change the rate of gain increase,X2 step.4'b0000:6ms...4'b1111:6.144s [7:4] dcy, Change the gain reduction rate,X2 step.4'b0000:24ms...4'b1111:24.576s	0x32

Address Offset	Name	Type	Description	Reset Value
0x23	CODEC_NOISE	RW	[0] ngat, Noise gate enable [2:1] ngg, Noise gate type.2'b00:hold PGA gain constant;2'b01:mute output;2'b10:softmute/unmute output;2'b11:reserve [7:3] ngth, Noise gate threshold value,1.5dB step;5'b00000:-76.5dBFS...5'b11110:-31.5dBFS;5'b11111:-30dBFS	0x00
0x24	CODEC_MINGAIN	RW	[2:0] mingain, mingain,6dB step:3'b000:-72dB...3'b111:-30dB [4] dmic_clk_sel, dmic clk sel. [5] codec_trig_en, 1: trig codec via pem [6] codec_dis_en, 1: disable codec via pem [7] codec_schdl_en, codec schedule enable	0x02
0x28	CODEC_STIMER_TARGET0	RW	codec_stimer_target[7:0] [7:0] codec stimer for schedule	0x00
0x29	CODEC_STIMER_TARGET1	RW	codec_stimer_target[15:8] [7:0] codec stimer for schedule	0x00
0x2a	CODEC_STIMER_TARGET2	RW	codec_stimer_target[23:16] [7:0] codec stimer for schedule	0x00
0x2b	CODEC_STIMER_TARGET3	RW	codec_stimer_target[31:24] [7:0] codec stimer for schedule	0x00

6 BLE/802.15.4/2.4GHz RF Transceiver

6.1 Overview

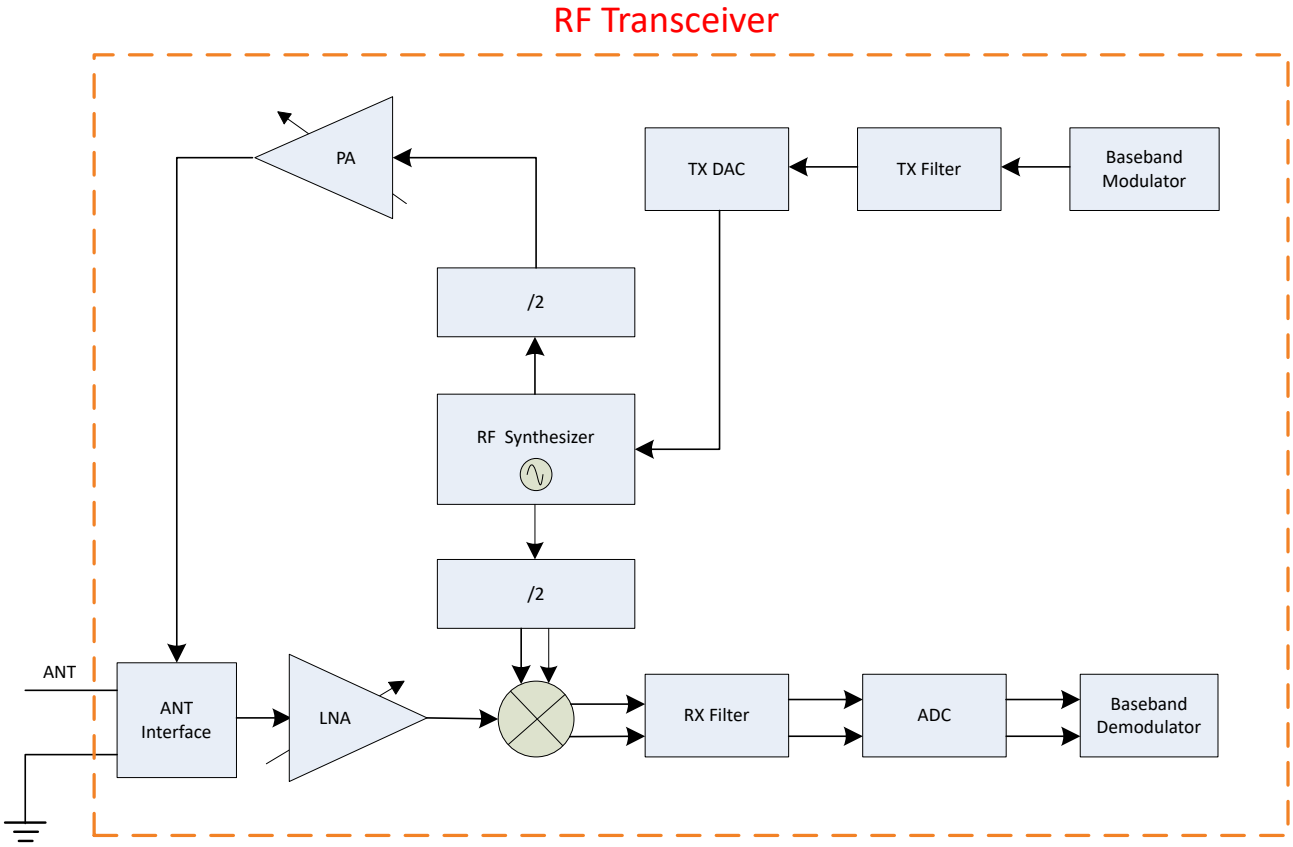
The SoC integrates an advanced RF transceiver for Bluetooth LE, 802.15.4 and 2.4 GHz application.

This RF transceiver works in the worldwide 2.4GHz ISM (Industrial Scientific Medical) band and it consists of a fully integrated RF synthesizer, a Power Amplifier (PA), a Low Noise Amplifier (LNA), a TX filter, a RX filter, a TX DAC, a RX ADC, a modulator and a demodulator.

The transceiver can be configured to work in standard-compliant 1 Mbps BLE mode, 2 Mbps enhancement BLE mode,125 kbps BLE long range mode (S8), 500 kbps BLE long range mode (S2), IEEE 802.15.4 standard-compliant 250 kbps mode and proprietary 1 Mbps, 2 Mbps, 250 kbps and 500 kbps mode.

The block digram of the transceiver is shown below.

Figure 6-1 Block Diagram of RF Transceiver



To control external PA and LNA, first follow the GPIO lookup table (see Section 11.1.2) to configure the specific two pins as TX_CYC2PA and RX_CYC2LNA function, respectively. Note: To use TX_CYC2PA and RX_CYC2LNA function for the two pins, other functions with higher polarity should be disabled at the same time. After the two pins are configured as TX_CYC2PA and RX_CYC2LNA function, the output function is enabled. Generally the two pins are high active: When both the two pins output low level, the external PA and LNA are disabled; when one of the two pins output high level, the external PA/LNA are enabled correspondingly; the two pins won't output high level simultaneously.

Table 6-1 External RF Transceiver Control Example

TX_CYC2PA	RX_CYC2LNA	External RF Transceiver
L	L	Both LNA and PA off
L	H	LNA on
H	L	PA on
H	H	N/A

6.2 Air Interface Data Rate and RF Channel Frequency

Air interface data rate, the modulated signaling rate for RF transceiver when transmitting and receiving data, is configurable via related register setting: 125 kbps, 250 kbps, 500 kbps, 1 Mbps, 2 Mbps.

RF transceiver can operate with frequency ranging from 2.400 GHz to 2.4835 GHz. The RF channel frequency setting determines the center of the channel.

6.3 Baseband

The baseband is disabled by default. The corresponding API is available for user to power on/down the baseband and enable/disable clock, so that the baseband can be turned on/off flexibly.

The baseband contains dedicated hardware logic to perform fast Automatic Gain Control (AGC), access code correlation, Cyclic Redundancy Check (CRC), data whitening, encryption/decryption and frequency hopping logic.

The baseband supports all features required by Bluetooth and 802.15.4 specification.

6.3.1 Packet Format

Packet format in standard 1 Mbps BLE mode is shown in table below.

Table 6-2 Packet Format in Standard 1 Mbps BLE Mode

LSB MSB

Preamble	Access Address	PDU	CRC
(1 octet)	(4 octets)	(2 ~ 257 octets)	(3 octets)

Packet length 80 bit ~ 2120 bit (80 ~ 2120 μs @ 1 Mbps).

Packet format in standard 2 Mbps BLE mode is shown in table below.

Table 6-3 Packet Format in Standard 2 Mbps BLE Mode

LSB MSB

Preamble	Access Address	PDU	CRC
(2 octet)	(4 octets)	(2 ~ 257 octets)	(3 octets)

Packet length 88 bit ~ 2028 bit (44 ~ 1064 μs @ 2 Mbps).

Packet format in standard 500kbps/125kbps BLE mode is shown in table below.

Table 6-4 Packet Format in Standard 500 kbps/125 kbps BLE Mode

LSB MSB

Preamble (10 octet)	Access Address (4 octets)	CI (2 bits)	TERM1 (3 bits)	PDU (2 ~ 257 octets)	CRC (3 octets)	TERM2 (3 bits)

Packet format in 250 kbps 802.15.4 mode is shown in table below.

Table 6-5 Packet Format in 802.15.4 Mode

LSB MSB

Preamble (4~16 octet)	SFD (1 octet)	Frame Length (1 octet)	PSDU (Variable 0~127 octets)	CRC (2 octets)
SHR		PHR	PHY Payload	

Packet format in 2.4 GHz proprietary mode is shown in table below:

Table 6-6 Packet Format in Proprietary Mode

LSB/MSB MSB/LSB

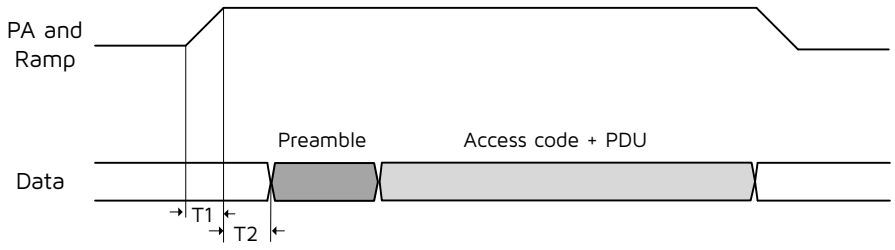
Preamble (configurable 8 octet)	Access Address (configurable 2~5 bytes)	Packet Controller + Payload (1~33 bytes)	CRC (1~2 bytes)

6.3.2 Modem

6.3.2.1 Power Amplifier (PA)

Power amplifiers are for radio frequency signal amplification through an antenna within a defined frequency range.

Figure 6-2 PA Ramp



As shown in the figure above, the time between PA starts from 0 and PA ramp up to steady is defined as PA ramp up step T1, which has four kinds listed below; the time between PA ramp up to steady and transmission data starts to preamble is defined as T2, which is adjustable.

PA ramp up step 1: Increment PA slices to programmed value from 0 in one shot.

PA ramp up step 2: Increment PA slices to programmed value from 0 using delay of 41.6ns between each step. (1 - 2 - 4 - 8 -16 - 32 - 48-63)

PA ramp up step 3: Increment PA slices to programmed value from 0 using delay of 83ns between each step. (1 - 2 - 4 - 8 -16 - 32 - 48-63)

PA ramp up step 4: Increment PA slices to programmed value from 0 using delay of 125ns between each step. (1 - 2 - 4 - 8 -16 - 32 -48-63)

PA ramp up step 5: Increment PA slices to programmed value from 0 using delay of 166ns between each step. (1 - 2 - 4 - 8 -16 - 32 -48-63)

PA ramp up step 6: Increment PA slices to programmed value from 0 using delay of 250ns between each step. (1 - 2 - 4 - 8 -16 - 32 -48-63)

PA ramp up step 7: Increment PA slices to programmed value from 0 using delay of 500ns between each step. (1 - 2 - 4 - 8 -16 - 32 -48-63)

PA ramp up step 8: Increment PA slices to programmed value from 0 using delay of 1000ns between each step. (1 - 2 - 4 - 8 -16 - 32 -48-63)

At any point, if the value programmed in TX_PA_PWR<5:0> is lower than the next step, then the ramp happens to the programmed value and stop.

Ramp down should also follow the same sequence and delay as ramp up. During ramp down, from the programmed value in TX_PA_PWR<5:0>, the sequence is followed down to 0 from the next lower step onwards.

Between turning on PA and tx_data(preamble), it supports undermodulation tone.

6.3.2.2 Fast Settle

Fast Settle is a feature improvement for timing sequence which include transmission (tx) and receiving (rx). For BLE, fast settle is realized by configuring registers manually and it needs regular calibration or manual value setting. It is recommended to do calibration at frequency hopping.

Configure register 0x170629[3] to manually enable rx sequence; configure register 0x170629[4] to manually enable tx sequence.

Set 0 for registers 0x170629[3] and 0x170629[4] to switch to normal situation.

NOTE: If using fast settle in normal registers is needed, enable: 0x17063f[5], NORM_PKT_FAST_STL_EN.

Fast Settle supports dual-channel detection. It has two antennas switch to select the most reliable RF signal path based on the build. This is done by the radio transceiver during preamble field search without the need for micro-controller interaction.

6.3.2.3 AoA/AoD

This chip supports AOD/AOD features defined in Bluetooth Core 5.1. Client device is hereinafter referred to as "the LE radio that we want to get direction information". Server device is hereinafter referred to as "the LE

radio that set as the basepoint of the direction information". Client device can get its direction information for a server device though AOA/AOD method. Using direction information from several server devices and profile-level information giving their locations, a client radio can calculate its own position. For the following AoA/AoD description, this chip supports up to 64 antennae of the antenna array for the data transmission and receiving.

(1) Angle of Arrival (AOA)

An LE device can make its direction available to a peer device by transmitting direction finding enabled packets using a single antenna.

The peer device, consisting of an RF switch and antenna array, switches antennae while receiving part of those packets and captures IQ samples. The IQ samples can be used to calculate the phase difference in the radio signal received using different elements of the antenna array, which in turn can be used to estimate the angle of arrival (AoA).

Consider a receiver device with an antenna array consisting of two antennae, separated by distance d. The transmitter device uses a single antenna to transmit a signal. As shown in Figure 1, a perpendicular line can be drawn from an incoming signal wave front extending to the furthest antenna (antenna2) at the point of intersection to the closest antenna (antenna 1). The adjacent side of that right triangle represents the path difference relative to the angle of incidence of that wave front between both antennae. The phase difference, ψ, in the signal arriving at the two antennae is then

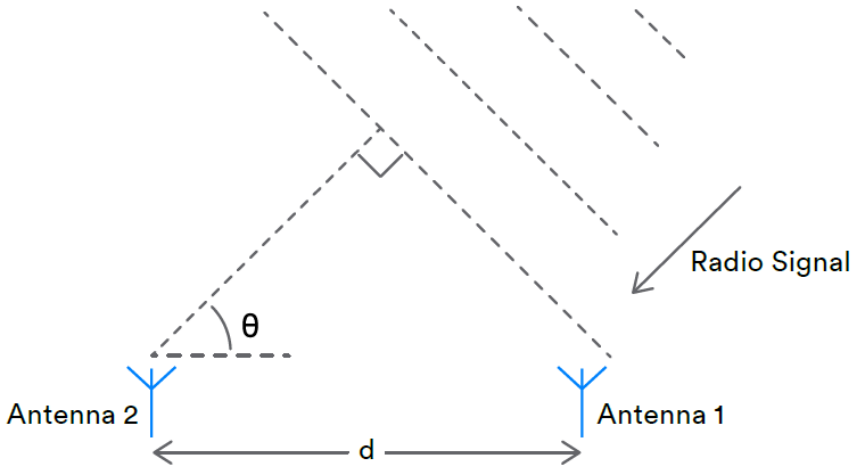
$$\psi = (2\pi d \cos(\theta))/\lambda$$

where λ is the wavelength of the signal and θ is the angle of arrival (measured from a line connecting the two antennae in the receiver), and so

$$\theta = \arccos((\psi\lambda)/(2\pi d))$$

Note: The distance d is profile-level information that is used by the receiving device to calculate the angle of arrival.

Figure 6-3 Measuring the angle of arrival



(2) Angle of Departure (AOD)

A device consisting of an RF switch and antenna array can make its angle of departure (AoD) detectable by transmitting direction finding enabled packets, switching antennae during transmission.

The peer device receives those packets using a single antenna and captures IQ samples during part of those packets. Determination of the direction is based on the different propagation delays of the LE radio signal

between the transmitting elements of the antenna array and a receiving single antenna. The propagation delays are detectable with IQ measurements. Any receiving LE radio with a single antenna that supports the AoD feature can capture IQ samples and, with the aid of profile-level information specifying the antenna layout of the transmitter, calculate the angle of incidence of the incoming radio signal.

Consider a transmitter device with an antenna array consisting of two antennae, separated by distance d. The receiver device uses a single antenna to receive the signals. The phase difference, ψ, in the signal from antenna 1 and the signal from antenna 2 arriving at the receiver is then

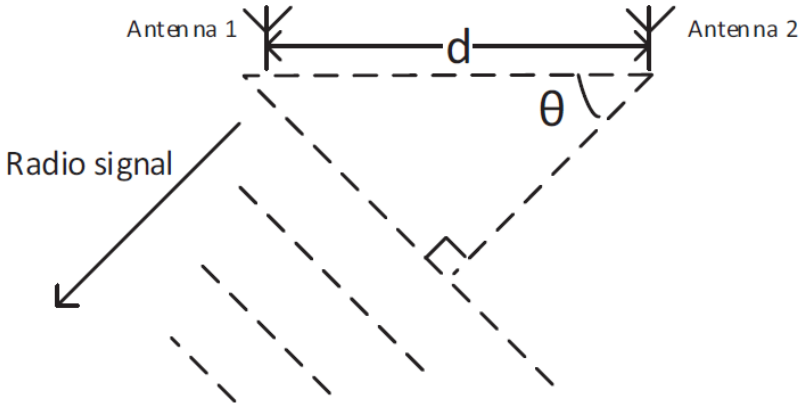
$$\psi = (2\pi d \cos(\theta))/\lambda$$

where λ is the wavelength of the signal and θ is the angle of departure (measured from a line connecting the two antennae in the transmitter), and so

$$\theta = \arccos((\psi\lambda)/(2\pi d))$$

Note: The distance d is profile-level information that a transmitting device exchanges with the receiving device in order for the receiving device to calculate the angle of departure.

Figure 6-4 Measuring the angle of departure



6.3.2.4 1-N Multi-Receiver

The chip modem supports 1-N multi-receiver on all modes including Bluetooth LE, Zigbee, and 2.4 GHz proprietary. The access code of expanded lower 16 bit is configurable for different channels.

6.3.2.5 Low-Speed Modulation Method

The low-speed modulation supports 0.25 kbps and 100 kbps. It supports all kinds of packet formats. The adjustable-rate configuration table per output bit speed is shown as below.

Table 6-7 Adjustable-rate configuration per output bit speed

Output bit speed (bps)	r_rate_adj(14bits) configuration value
100k	40
50k	80
25k	160

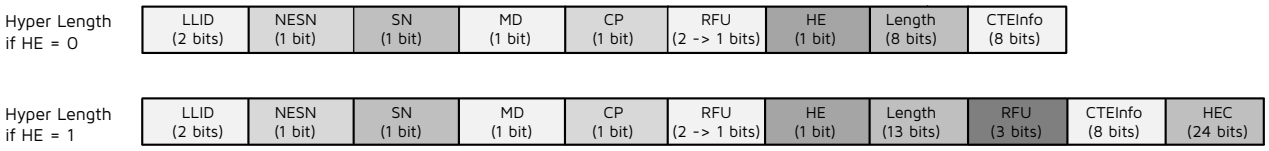
Output bit speed (bps)	r_rate_adj(14bits) configuration value
20k	200
10k	400
1k	4000
0.5k	8000
0.25k	16000

6.3.3 Linklayer

6.3.3.1 Hyper Length

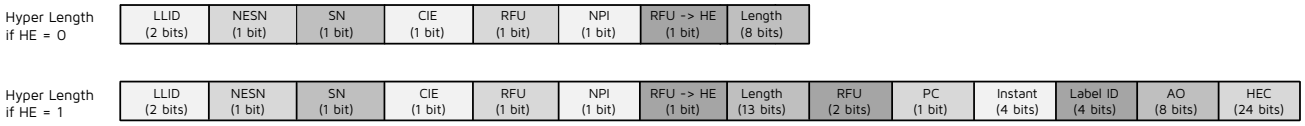
This chip supports LE Hyper Length Extensions. The structure of data physical channel PDU header is shown as below.

Figure 6-5 Data Physical Channel PDU Header Structure



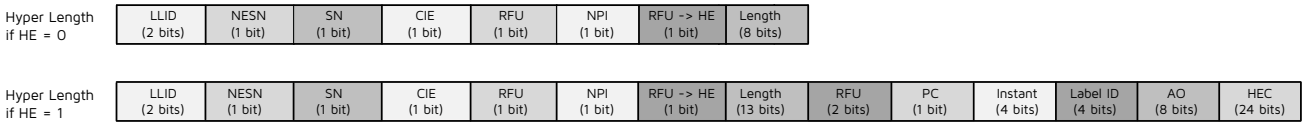
The structure of Connected Isochronous PDU header is shown as below.

Figure 6-6 Connected Isochronous PDU Header Structure



The structure of Broadcast Isochronous PDU header is shown as below.

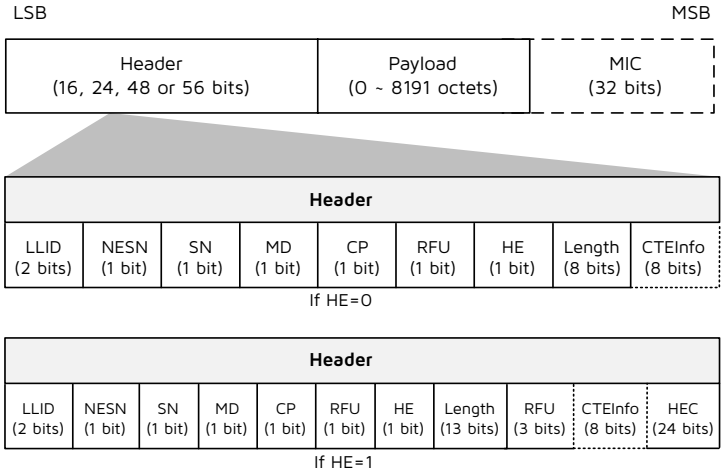
Figure 6-7 Broadcast Isochronous PDU Header Structure



Data Physical Channel PDU

The Data Physical Channel PDU has a 16 to 56-bit header, a variable size payload, and may include a Message Integrity Check (MIC) field.

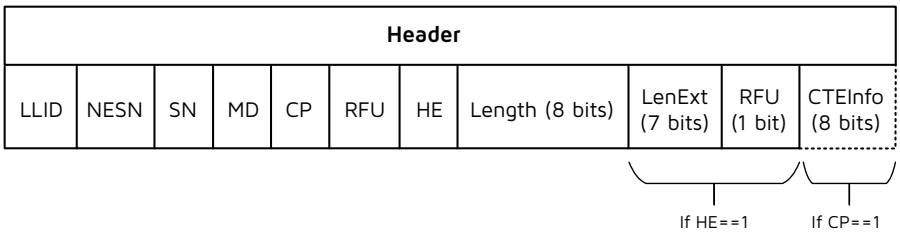
Figure 6-8 Data Physical Channel Format



- When HE bit is set to 0, the 8-bit length field has the range 0 to 255 octets and the Payload shall be less than or equal to 251 octets in length.
- When HE bit is set to 1, the 13-bit length field has the range 0 to 8191 octets and the Payload shall be less than or equal to 8187 octets in length.

The following figure shows how Telink implements this format.

Figure 6-9 Data Physical Channel Format



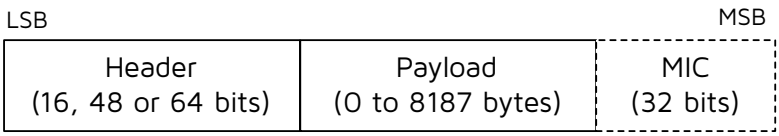
Isochronous Physical Channel PDU

(1) The Isochronous Physical Channel PDU has a 16 to 64-bit header, a variable size payload, and may include a Message Integrity Check (MIC) field.

(2) The format of the Header field and the payload depends on the type of the Isochronous Physical Channel PDU that is being used. (CIS or BIS PDU)

(3) The MIC field shall be included in all PDUs that contain a non-zero length Payload that are sent on an encrypted CIS or BIS.

Figure 6-10 Isochronous Physical Channel PDU Format



Connected Isochronous PDU (CIS)

(1) A Connected Isochronous PDU (CIS PDU) shall be either a CIS Data PDU or a CIS Null PDU.

(2) A CIS Data PDU is used to carry isochronous data.

(3) A CIS Null PDU is used when there is no data to be sent.

Figure 6-11 Connected Isochronous Format 1

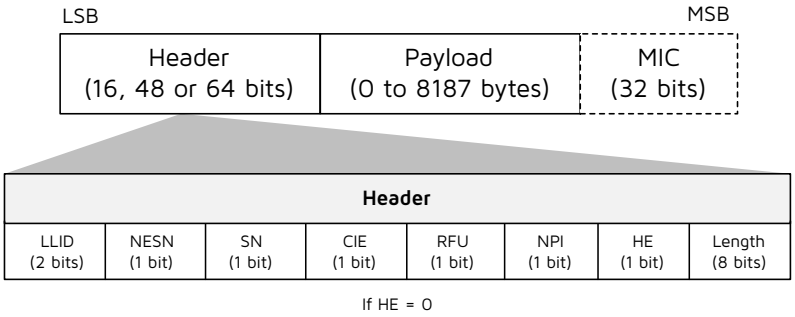
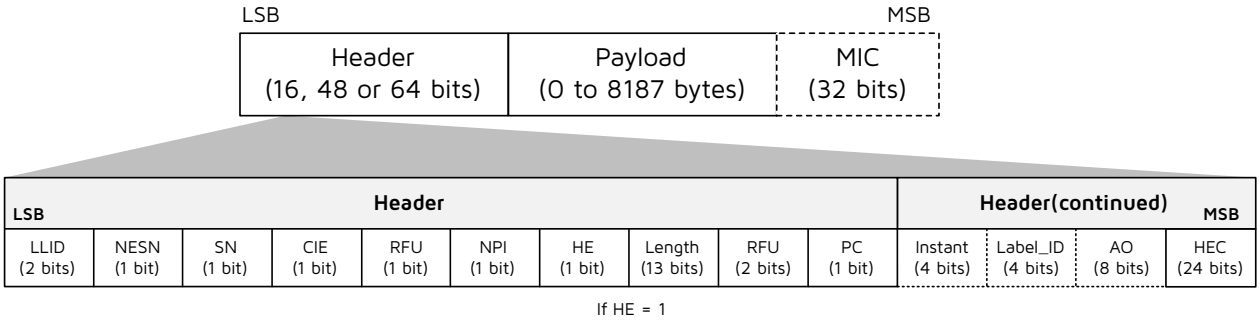


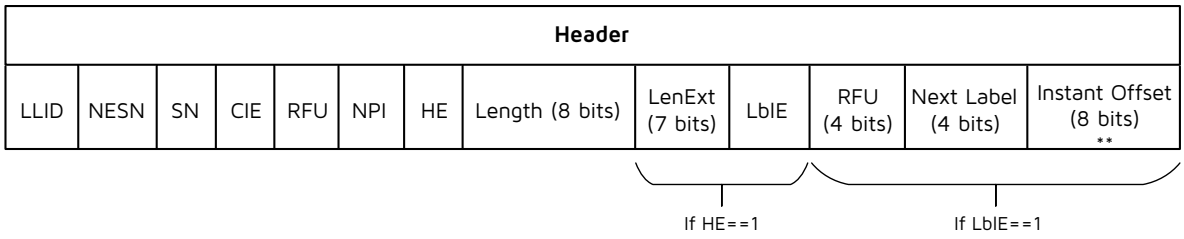
Figure 6-12 Connected Isochronous Format 2



- The Header Extension (HE) field shall be set "1" to enable header extension. When HE bit is set to 1, the Header includes at minimum an extended Length field, a PC bit and HEC field as described below. Otherwise these fields are absent.
- The Length field shall be set to 0 for a CIS Null PDU. When HE bit is set to 1, the Length field is a 13-bit field. Otherwise the Length field is an 8-bit field.
- Parameter Change (PC) field shall be set "1" to enable a Label based parameter update.
- Instant, Label_ID and AO fields shall be present when PC bit is set to 1. Otherwise these fields are absent.

The following figure shows how Telink implements this format.

Figure 6-13 Connected Isochronous Format



** Last 8 bits of CIS event count when update takes effect.

Broadcast Isochronous PDU (BIS)

(1) A Broadcast Isochronous PDU (BIS PDU) shall be either a BIS Data PDU or BIG Control PDU.

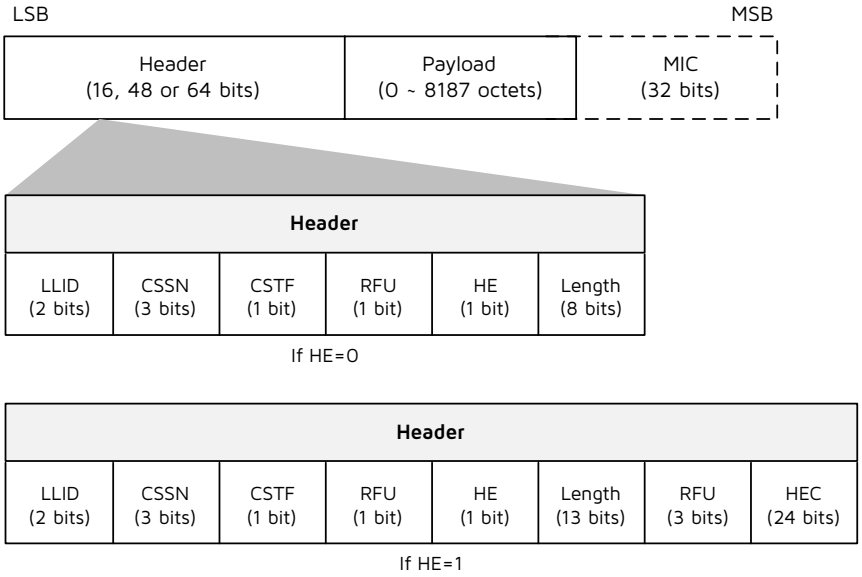
(2) A BIS Data PDU is used to carry isochronous data.

(3) A BIG Control PDU is used to send control information for a BIG.

(4) The Header Extension (HE) field shall be set to enable header extension. When HE bit is set to 1, the Length field is a 13-bit field. Otherwise the Length field is an 8-bit field.

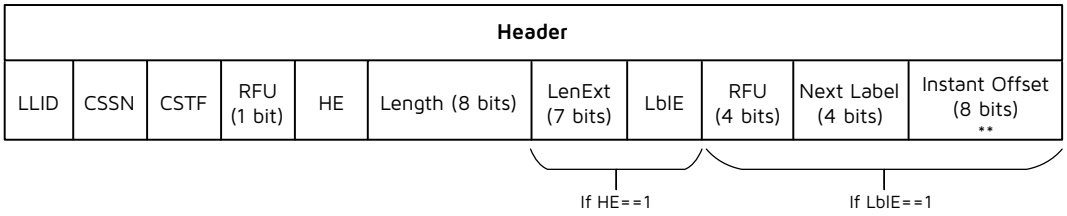
(5) The Header Error Checksum (HEC) field shall be set when HE bit is set to 1.

Figure 6-14 Broadcast Isochronous Format



The following figure shows how Telink implements this format.

Figure 6-15 Broadcast Isochronous Format

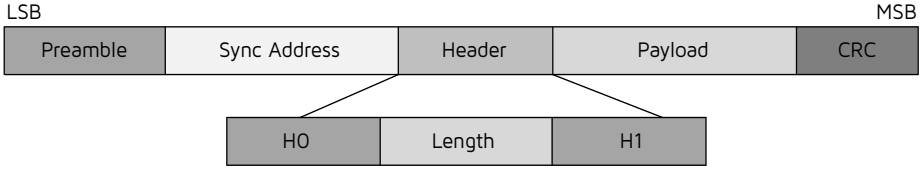


** Last 8 bits of BIS event count when update takes effect.

6.3.3.2 2.4GHz Generic Packet

The Generic FSK Link Layer Controller supports a packet structure based on the following template:

Figure 6-16 Generic FSK Packet Structure

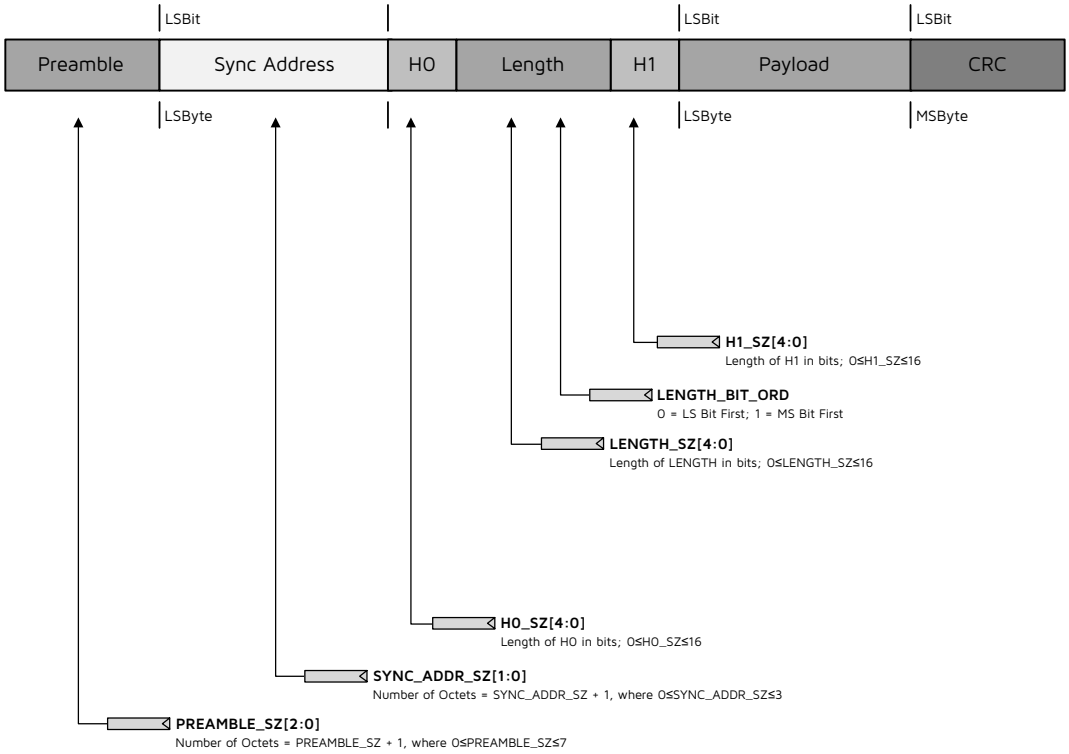


The controller also offers a highly generic packet format, which features programmability that allows for a highly configurable packet structure, defining the lengths, bit ordering, and contents of individual packet fields, defining the start and end points for whitening and CRC.

Most of the Generic Packet elements have associated configurability by way of a set of programmable registers, which allows software to pre-configure. The following diagram depicts the register configurability associated with the Preamble, Sync Address, H0, Length, and H1 elements of the packet structure.

The controller provides 4 bitrate options for Generic Packet, for both TX and RX. (2M/1M/500k/250k).

Figure 6-17 Packet Structure Definitions



Preamble

The preamble pattern is 0x55 or 0xAA, and preamble is transmitted LSB first. The controller hardware selects the preamble pattern based on the first transmitted bit of Sync Address, such that the last bit of preamble is the opposite polarity from the first bit of Sync Address.

Provide SZ register for Preamble fields:

- PREAMBLE_SZ: specify the number of preamble pattern in octets

Sync Address

Sync Address, also known as Sync word or Access Address, is the second packet element transmitted by the Link Layer Controller.

For reception, up to 8 unique Sync Addresses are supported. Any combination of the 8 Sync Addresses can be simultaneously searched for. Multiple Sync Address capability also enables multiceiver operation.

Provide SZ register for Sync Address fields:

- SYNC_ADDRESS_SZ: specify the number of Sync Address in octets

Header

The generic packet header is comprised of H0, LENGTH, and H1, in that order. Each of the 3

fields of the header has programmable length, from 0 to 16 bits. Although the size of the individual H0, LENGTH, and H1 components need not be aligned to a byte boundary, the overall header must be byte-aligned. That is, the sum of the sizes (in bits) of H0, LENGTH, and H1, must be a integer multiple of 8 bits.

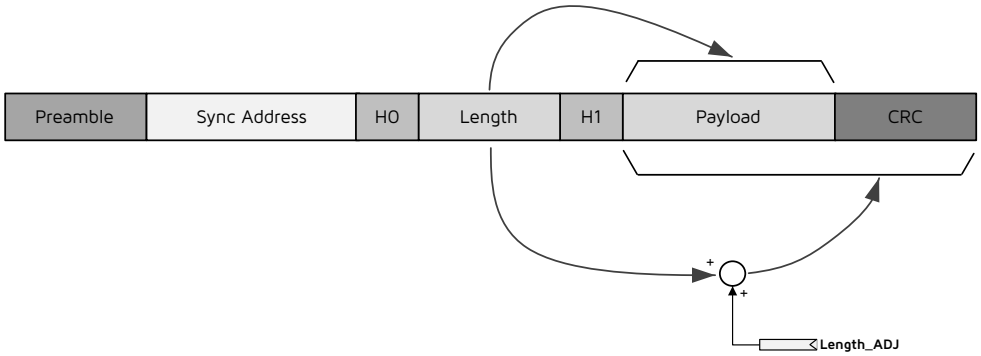
Provide SZ register for H0 and H1 fields of header:

- H0/H1_SZ: specify the length of H0 / H1 field in bits.

Provide SZ | ORD | ADJ register for LENGTH field of header.

- LENGTH_SZ: specify the length of LENGTH field in bits.
- LENGTH_BIT_ORD: specify the bit order of LENGTH field.
- LENGTH_ADJ: specify the calculation method of LENGTH, PAYLOAD_ LEN+ADJ (offset) [-31<=LENGTH_ADJ<=31].

Figure 6-18 Length Filed Interpretation with Length_ADJ



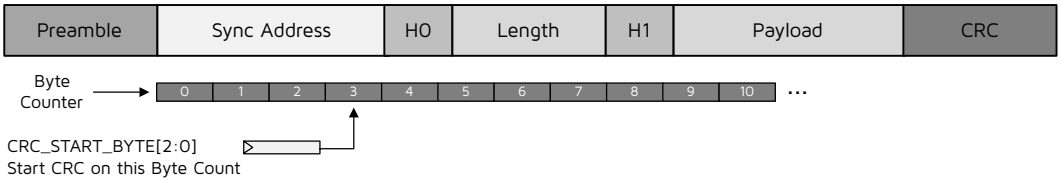
CRC configurability

Provide CRC_ SZ | CRC_ INIT | CRC_ POLY register:

- CRC_ SZ: CRC length.
- CRC_ INIT: CRC initial value.
- CRC_ POLY: Numerical sequence corresponding to CRC polynomial.

Provide the starting position for configuring CRC calculation CRC_START_BYTE register.

Figure 6-19 CRC Start Byte



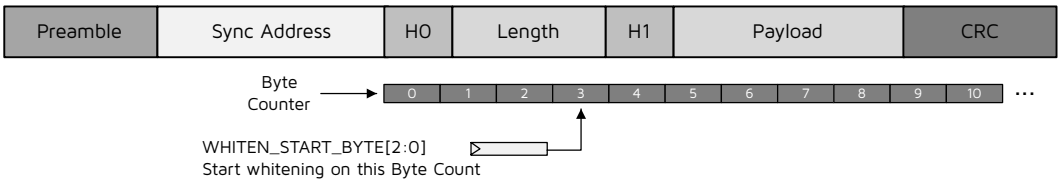
Whitening configurability

Provide WHITEN_ SZ | WHITEN_ INIT | WHITEN_ POLY register:

- WHITEN_ SZ: Whitening length.
- WHITEN_ INIT: Whitening polynomial initial value.
- WHITEN_ POLY: Numerical sequence corresponding to CRC polynomial.

Provide the starting position for configuring Whitening calculation WHITEN_START_BYTE register.

Figure 6-20 Whitening Start Byte



6.3.3.3 Zigbee Extension Packet

The format of Preamble, SFD and PHY header are old mode, after PHY Header the Zigbee extension packet becomes 2Mbps.

Figure 6-21 Zigbee Private Packet Format

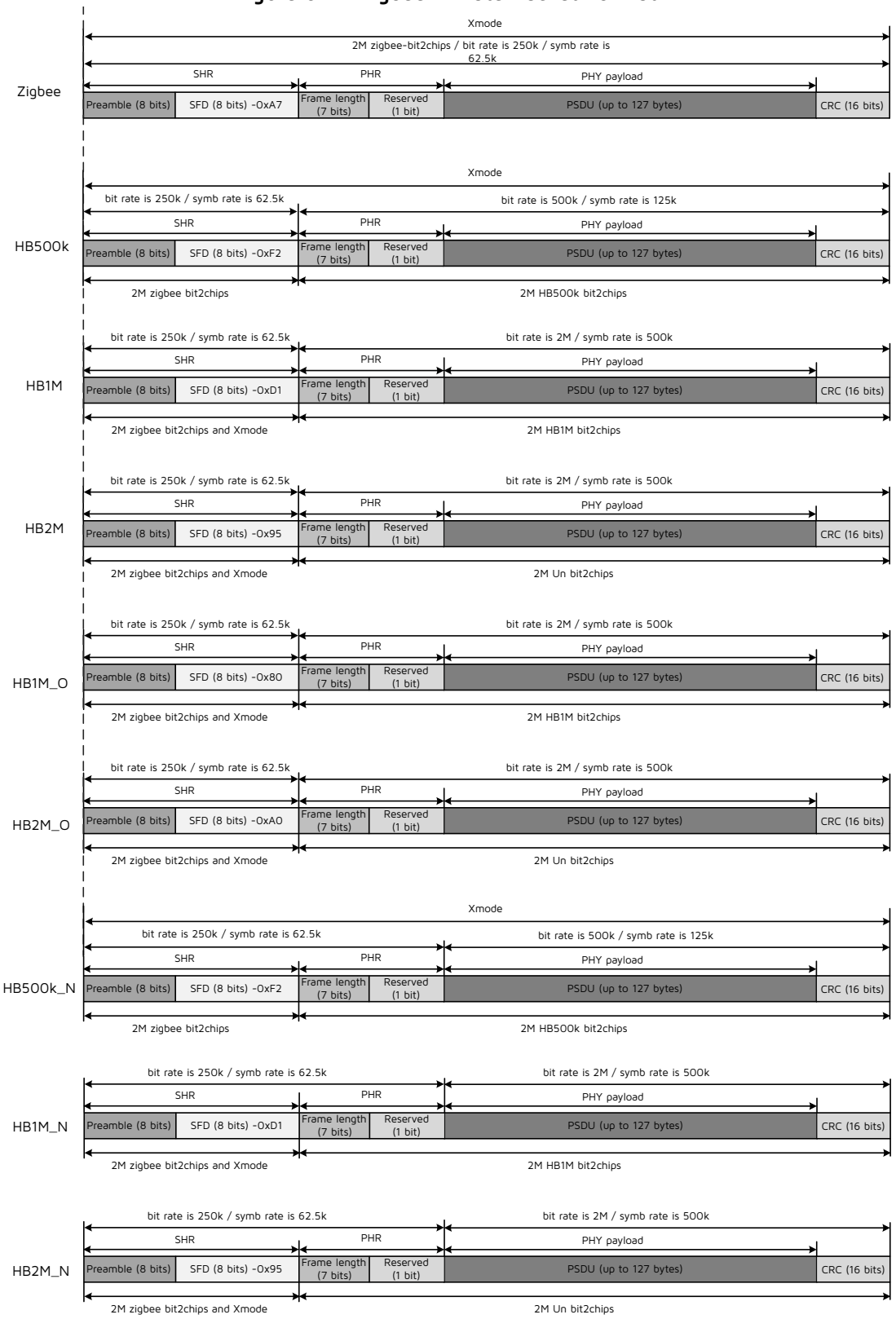
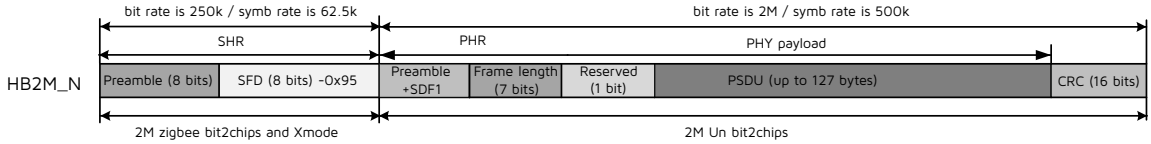


Figure 6-22 Zigbee Private Packet Format 2



The length of Preamble+SDF1 is adjustable, and the maximum is 32.

6.3.3.4 Packet Filter

The packet filter is for Bluetooth LE and 2.4GHz proprietary applications. In this chip, it adds the match window of ADV filter HEADER interrupt or hardware processing payload, supports single bit/byte/multi-byte mode, and continues to receive or close the match.

The most important thing is to process the BLE advertising packet, which has a fixed MAC address. The main purpose is to directly target the BLE packet format. In fact, it is easier to match data according to a given location, and it is easier to be compatible with other situations that we may not consider (such as Mesh, Zigbee or Thread broadcast channels).

Regardless of the upper application, it can be processed as long as it can confirm that a certain byte and a certain field can be identified.

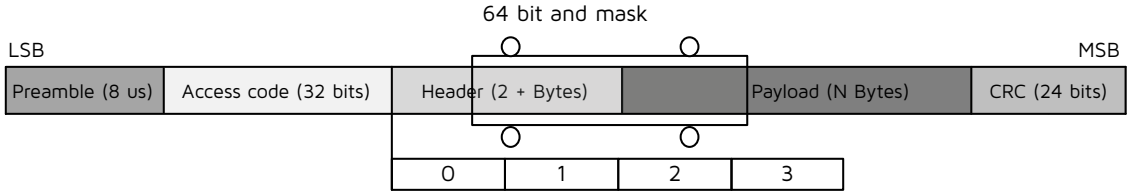
(1) It can analysis to a certain length, and stop.

(2) It can analysis to the Nth byte, which is the same as a certain value (configurable) and can continue or stop.

(3) It can analysis to several consecutive bytes, which are the same as some values (configurable) and can continue or stop.

(4) In addition to bit mask configuration, you can only compare the specific bits in the mask, and do not compare the bits which are masked out, so that you can achieve bit by bit comparison. In this way, all the requirements I mentioned earlier can be realized.

Figure 6-23 Packet Filter



NOTE: Normal mask window supports single bit; AES encryption mode can only support byte at least.

6.3.3.5 Hardware Frequency Hopping

The hardware frequency hopping integrates BLE hopping accelerator. The frequency selection module automatically calculates the frequency to be used for the packet, depending on the parameters given by the software in the CS-FORMAT field. The channel map is split in between 3 parts:

(1) 37 data channels (index in [0:36] range) dynamically selected, and two possible hopping schemes.

(2) 3 primary advertising channels (index in [37:39] range) dynamically selected.

(3) 1 secondary advertising channel (index in [0:36] range).

The table below shows how to select in between the BLE 40 channels split and usage.

Table 6-8 Frequency Hopping Scheme Selection

CS-Format	Channel Type
Master Connect	Data Channel
Slave Connect	Data Channel
ISO Mode 0 Master Connect	Data Channel
ISO Mode 0 Slave Connect	Data Channel
Advertiser	Primary Advertising Channels
Extended Advertiser	Primary and/or Secondary Advertising Channels
Passive Scanner	Primary Advertising Channels
Extended Passive Scanner	Primary and/or Secondary Advertising Channels
Active Scanner	Primary Advertising Channels
Extended Active Scanner	Primary and/or Secondary Advertising Channels
Initiator	Primary Advertising Channels
Extended Initiator	Primary and/or Secondary Advertising Channels
Tx Test Mode	n/a[a]
Rx Test Mode	n/a[a]
Tx/Rx Test Mode	n/a[a]

a. In test mode, the CS-CH_IDX is used blindly till abort is required.

The selected frequency is used to program the radio before any transmission or reception. Depending on the operating mode, the hopping scheme is adapted as described in the Bluetooth Low Energy specifications: advertising, scanning request / response, connection indication, and master or slave connection are supported, as described in [a].

The following table provides correspondence between channel type, channel index, and frequency.

Table 6-9 Bluetooth Low Energy Channel Index and Physical Channel Correspondence

Frequency (MHz)	Link Type	Channel Index	Channel Number
2402	Primary Advertising Channel	37	0
2404	Data Channel / ISO Channel / Secondary Advertising Channel	0	1

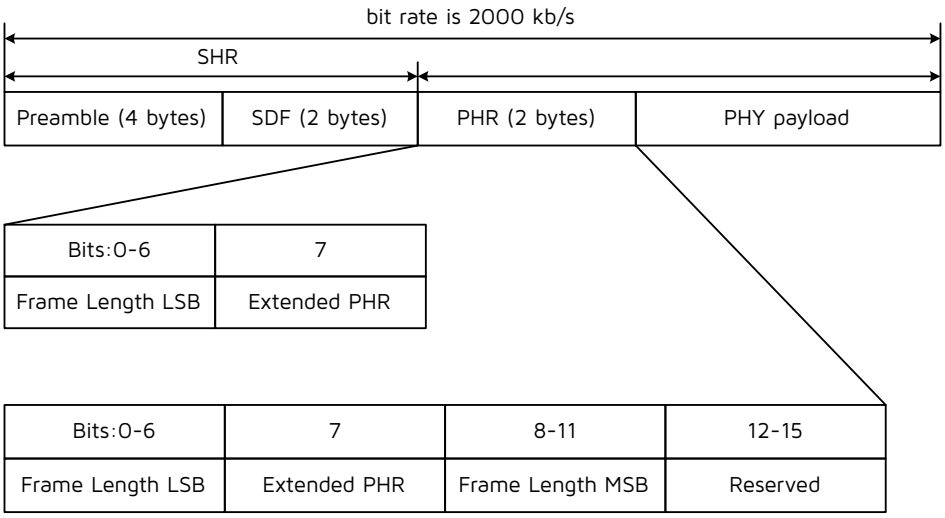
Frequency (MHz)	Link Type	Channel Index	Channel Number
2406	Data Channel / ISO Channel / Secondary Advertising Channel	1	2
...	Data Channel / ISO Channel / Secondary Advertising Channel	...	...
2424	Data Channel / ISO Channel / Secondary Advertising Channel	10	11
2426	Primary Advertising Channel	38	12
2428	Data Channel / ISO Channel / Secondary Advertising Channel	11	13
...	Data Channel / ISO Channel / Secondary Advertising Channel	...	...
2478	Data Channel / ISO Channel / Secondary Advertising Channel	36	38
2480	Primary Advertising Channel	39	39

6.3.3.6 802.15.4

The 802.15.4t is IEEE Standard for Low-Rate Wireless Networks–Amendment 4: Higher Rate (2 Mb/s) Physical (PHY) Layer. It supports 2 Mb/s data rates, utilizing the 2400-2483.5 MHz band.

The packet format of 802.15.4t is shown as below.

Figure 6-24 802.15.4 Packet Format

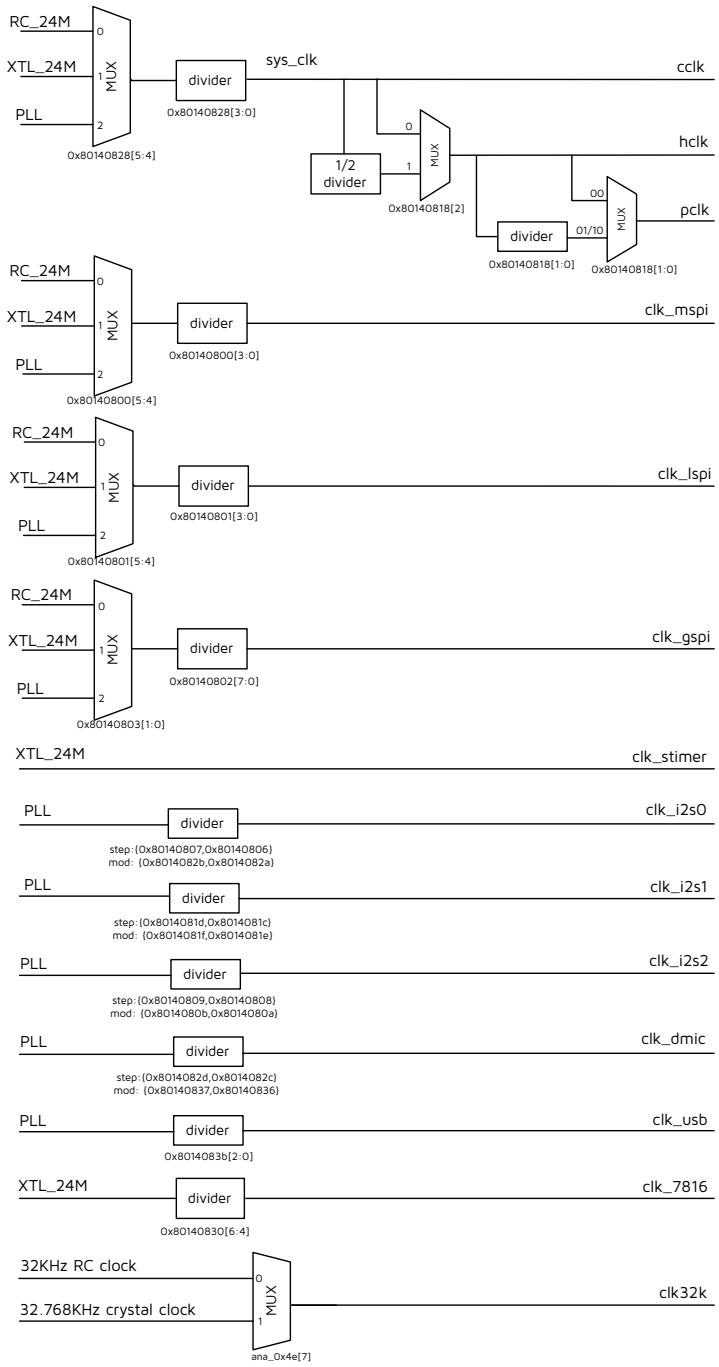


7 Clock

7.1 Clock Sources

The SoC's clock sources are a 24 MHz RC oscillator and an external 24 MHz crystal, as shown below.

Figure 7-1 Clock Sources



NOTE: *The maximum frequency supported by cclk is 240 MHz.*

The clock sources of each module are shown in table below.

Table 7-1 Clock Sources of Each Module

Module	Clock Source(s)
PLDM	hclk
PLIC_SW	hclk
PLMT	clk32k, hclk
PLIC	hclk
MCU	cclk
BROM	hclk
ZB	clk32k, hclk
AUDIO	hclk, pclk, clk_i2s, clk_dmic
USB	hclk, clk_usb
PKE	hclk
CHACHA20	hclk
SKE	hclk
TRNG	hclk, clk_ro
HASH	hclk
SWIRE	hclk
OSR_REG	hclk
DMA	hclk
BMC	hclk
MSPI	hclk, clk_mspi
GSPI	hclk, clk_gspi
LSPI	hclk, clk_lspi
ILM	cclk
DLM	cclk
PEM	cclk, hclk, pclk
RZ	pclk
IR_LEARN	pclk
I2C1	pclk

Module	Clock Source(s)
PWM	pclk, clk32k
EOTP	pclk
UART2	pclk
I2C	pclk
QDEC	pclk, clk32k
STIMER	pclk, clk32k, clk_stimer
SAR_ADC_DIG	pclk, xtl_24m
ALGM	pclk
TIMER	pclk
UART1	pclk
UART0	pclk
SPI_SLV	hclk

7.2 System Clock

There are three selectable clock sources for MCU system clock (named cclk): RC_24M derived from 24 MHz RC oscillator, 24 MHz crystal, and pll clk. The sources are selectable via register CCLK_SET.cclk_sel[1:0].

And system clock can be reduced in frequency via frequency divider which is controlled via register CCLK_SET.cclk_div[3:0]. Assuming cclk_pre_div is the clock before frequency division. There is the following relationship expression:

Fcclk = Fcclk_pre_div/n (n=CCLK_SET.cclk_div[3:0], n=1~15)

7.3 Module Clock

Registers CLKEN0~CLKEN7 are used to enable or disable clock for various modules. By disable the clocks of unused modules, current consumption could be reduced.

7.3.1 clk_mspi

The clk_mspi is the system clock of MSPI module.

There are three selectable clock sources for clk_mspi: RC_24M derived from 24 MHz RC oscillator, 24 MHz crystal, and pll clk. The sources are selectable via register MSPI_MODE.mspi_sel[1:0].

And clk_mspi can be reduced in frequency via frequency divider which is controlled via register MSPI_MODE.mspi_div[3:0]. Assuming clkmspi_pre_div is the clock before frequency division. Then there is the following relationship expression:

Fclk_mspi = Fclkmspi_pre_div/n (n=MSPI_MODE.mspi_div[3:0],n=1~15)

7.3.2 clk_lspi

The clk_lspi is the clock of LSPI module.

There are three selectable clock sources for clk_lspi: RC_24M derived from 24 MHz RC oscillator, 24 MHz crystal, and pll clk. The sources are selectable via register LSPI_MODE.lspi_sel[1:0].

And clk_lspi can be lowered in frequency via frequency divider which is controlled via register LSPI_MODE.lspi_div[3:0]. Assuming clklspi_pre_div is the clock before frequency division. Then there is the following relationship expression:

Fclk_lspi = Fclklspi_pre_div/n (n=LSPI_MODE.lspi_div[3:0],n=1~15)

7.3.3 clk_gspi

The clk_gspi is the clock of GSPI module.

There are three selectable clock sources for clk_gspi: RC_24M derived from 24 MHz RC oscillator, 24 MHz crystal, and pll clk. The sources are selectable via register GSPI_MODE_H.gspi_sel[1:0].

And clk_gspi can be lowered in frequency via frequency divider which is controlled via register GSPI_MODE_L.gspi_div[7:0]. Assuming clkgspi_pre_div is the clock before frequency division. Then there is the following relationship expression:

Fclk_gspi = Fclkgspi_pre_div/n (n=GSPI_MODE_L.gspi_div[7:0],n=1~255)

7.3.4 System Timer Clock

System timer clock is derived from 24 MHz crystal oscillator.

7.3.5 I2S0 Clock

I2S0 clock is generated by pll_clk via frequency divider. Register I2S0_STEP_H[7] should be set as 1'b1 to enable I2S0 clock. I2S0 clock frequency dividing factor contains step and mod.

Registers I2S0_STEP_H[6:0], I2S0_STEP_L[7:0], I2S0_MOD_H[7:0] and I2S0_MOD_L[7:0] serve to set I2S0 clock step[14:0] and mod[15:0] respectively, and mod should be no less than 2*step.

I2S0 clock frequency, Fi2s0_clock, equals to pllclk*I2S0_step[14:0]/I2S0_mod[15:0].

7.3.6 I2S1 Clock

I2S1 clock is generated by pll_clk via frequency divider. Register I2S1_STEP_H[7] should be set as 1'b1 to enable I2S1 clock. I2S1 clock frequency dividing factor contains step and mod.

Registers I2S1_STEP_H[6:0], I2S1_STEP_L[7:0], I2S1_MOD_H[7:0] and I2S1_MOD_L[7:0] serve to set

I2S1 clock step[14:0] and mod[15:0] respectively, and mod should be no less than 2*step.

I2S1 clock frequency, Fi2s1_clock, equals to pllclk*I2S1_step[14:0]/I2S1_mod[15:0].

7.3.7 I2S2 Clock

I2S2 clock is generated by pll_clk via frequency divider. Register I2S2_STEP_H[7] should be set as 1'b1 to enable I2S2 clock. I2S2 clock frequency dividing factor contains step and mod.

Registers I2S2_STEP_H[6:0], I2S2_STEP_L[7:0], I2S2_MOD_H[7:0] and I2S2_MOD_L[7:0] serve to set I2S2 clock step[14:0] and mod[15:0] respectively, and mod should be no less than 2*step.

I2S2 clock frequency, Fi2s2_clock, equals to pllclk*I2S2_step[14:0]/I2S2_mod[15:0].

7.3.8 DMIC Clock

DMIC clock is generated by pll_clk via frequency divider. Register DMIC_STEP_H[7] should be set as 1'b1 to enable DMIC clock. DMIC clock frequency dividing factor contains step and mod.

Registers DMIC_STEP_H[6:0], DMIC_STEP_L[7:0], DMIC_MOD_H[7:0] and DMIC_MOD_L[7:0] serve to set DMIC clock step[14:0] and mod[15:0] respectively, and mod should be no less than 2*step.

DMIC clock frequency, Fdmic_clock, equals to pllclk*DMIC_step[14:0]/DMIC_mod[15:0].

7.3.9 USB Clock

USB clock is generated by pll_clk via frequency divider, the frequency is calculated with the following equations:

Fclk_usb = Fpllclk/n (n=0x801401fb[2:0], n=1~7)

7.3.10 clk_7816

Fclk_7816 = Fpad_24m/n (n=CLK_DIV[6:4], n=2~7)

Fclk_7816 = Fpad_24m/16 (n=CLK_DIV[6:4], n=0)

7.4 Clock Register Description

The Clock related registers are listed in table below. The base address of the following registers is 0x80140800.

Table 7-2 Clock Related Registers

Address Offset	Name	Type	Description	Reset Value
0x00	MSPI_MODE	R/W	[3:0]: mspi_mod [5:4]: mspi_div_in_sel	0x01
0x01	LSPI_MODE	R/W	[3:0]: lspi_mod [5:4]: lspi_div_in_sel	0x01
0x02	GSPI_MODE_L	R/W	[7:0]: gspi_mod_l	0x01
0x03	GSPI_MODE_H	R/W	[1:0]: gspi_div_in_sel	0x00
0x06	I2S0_STEP_L	R/W	[7:0]: i2s0_step_l	0x01
0x07	I2S0_STEP_H	R/W	[6:0]: i2s0_step_h [7]: i2s0_clk_en	0x00

Address Offset	Name	Type	Description	Reset Value
0x08	I2S2_STEP_L	R/W	[7:0]: i2s2_step_l	0x01
0x09	I2S2_STEP_H	R/W	[6:0]: i2s2_step_h [7]: i2s2_clk_en	0x00
0x0a	I2S2_MOD_L	R/W	[7:0]: i2s2_mod_l	0x02
0x0b	I2S2_MOD_H	R/W	[7:0]: i2s2_mod_h	0x00
0x10	CACHE_INIT	R/W	[5:0]: rsvd [6]: icache_disable_init [7]: dcache_disable_init	0x00
0x11	EOC_TS_VALUE	R/W	[7:0]: eoc_ts_value	0x80
0x14	MCU_CTRL0	W	[7]: mcu_reboot	0x00
0x15	MCU_RESET_VECTOR_R1	R/W	[7:0]: mcu_rst_vector_r1	0x00
0x16	MCU_RESET_VECTOR_R2	R/W	[7:0]: mcu_rst_vector_r2	0x00
0x17	MCU_RESET_VECTOR_R3	R/W	[7:0]: mcu_rst_vector_r3	0x00
0x18	BUSCLK_RATIO	R/W	[2:0]: busclk_ratio [2]0:mcu clk is the same as hclk , 1: mcu clk is 2 times of hclk [1:0]:0: hclk is the same as pclk, 1:hclk is 2 times of pclk, 2: hclk is 4 times of pclk cclk : hclk : pclk period ratio 3'b000 : 1:1:1; 3'b001 : 1:1:2; 3'b010 : 1:1:4; 3'b100 : 1:2:2; 3'b101 : 1:2:4;	0x00
0x1a	PROBE_CLK_SEL	R/W	[4:0]: probe_clk_sel	0x00
0x1c	I2S1_STEP_L	R/W	[7:0]: i2s1_step_l	0x01
0x1d	I2S1_STEP_H	R/W	[6:0]: i2s1_step_h [7]: i2s1_clk_en	0x00
0x1e	I2S1_MOD_L	R/W	[7:0]: i2s1_mod_l	0x01
0x1f	I2S1_MOD_H	R/W	[7:0]: i2s1_mod_h	0x00

Address Offset	Name	Type	Description	Reset Value
0x24	CLKEN0	R/W	[0]: lspi [1]: i2c [2]: uart0 [3]: usb [4]: pwm [5]: rsvd [6]: uart1 [7]: swires	0xa0
0x25	CLKEN1	R/W	[0]: rsvd [1]: stimer [2]: dma [3]: algm [4]: pke [5]: plmt [6]: gspi [7]: spislv	0xa0
0x26	CLKEN2	R/W	[0]: timer [1]: audio [2]: i2c1 [3]: rsvd [4]: mcu [5]: lm [6]: trng [7]: dpr	0x30
0x27	CLKEN3	R/W	[0]: rsvd [1]: trace [2]: brom [3]: rsvd [4]: mspi [5]: qdec [6]: saradc_dig [7]: rsvd	0x16

Address Offset	Name	Type	Description	Reset Value
0x28	CCLK_SET	R/W	[3:0]: cclk_div [5:4]: cclk_sel	0x01
0x2a	I2S0_MOD_L	R/W	[7:0]: i2s0_mod_l	0x02
0x2b	I2S0_MOD_H	R/W	[7:0]: i2s0_mod_h	0x00
0x2c	DMIC_STEP_L	R/W	[7:0]:dmic_step_l	0x01
0x2d	DMIC_STEP_H	R/W	[6:0]:dmic_step_h [7]: dmic_clk_sel	0x00
0x2e	WAKEUPEN	R/W	[0]: usb_pwdn_i, enable wakeup from usb [1]: gpio_wakeup_i, enable wakeup from gpio [2]: qdec_wakeup_i, enable wakeup from qdec [3]: reserved [4]: usb resume, enable remote wakeup from USB [5]: standby ex [7:6]: reserved	0x00
0x2f	PWDNEN	R/W	[0]: suspend_en_o [4]: ramcrc_clren_tgl [5]: rst_all [7]: stall_en_trg	0x00
0x30	CLK_DIV	R/W	[6:4]: r_7816_mod, 7816 clk div [7]: r_7816_clk_en, 7816 clk enable	0x60
0x32	RAM_CRC	R/W	[0]: ram_crc_err [1]: rsvd4 [2]: watchdog_rst_status [3]: jtag_rst_status	0x02
0x33	SEL	R/W	[0]: jtag_sel [1]: rsvd	0x00
0x36	DMIC_MOD_L	R/W	[7:0]: dmic_mod_l	0x02
0x37	DMIC_MOD_H	R/W	[7:0]: dmic_mod_h	0x00
0x3b	USB_DIV	R/W	[2:0]:usb_div	0x05

Address Offset	Name	Type	Description	Reset Value
0x44	CLKEN4	R/W	[0]: rsvd [1]: rsvd [2]: rsvd [3]: rsvd [4]: ske [5]: hash [6]: cclk [7]: zb	0x44
0x45	CLKEN5	R/W	[0]: rsvd [1]: uart2 [2]: rsvd [3]: rsvd [4]: ir_learn [5]: rsvd [6]: pem [7]: chacha20	0x00
0x46	CLKEN6	R/W	[0]: rz [7:1]: rsvd	0x00
0x47	CLKEN7	R/W	[0]: rz [7:1]: rsvd	0x00

8 Timer

8.1 Timer0/1 and Watchdog

The SoC supports two timers: Timer0 ~ Timer1. Timer0 and Timer1 support four modes: Mode 0 (System Clock Mode), Mode 1 (GPIO Trigger Mode), Mode 2 (GPIO Pulse Width Mode) and Mode 3 (Tick Mode), which are selectable via the register TMR_CTRL0 (address 0x80140140). In addition, the Timer0 and Timer1 support Input Capture function. The Input Capture function can be used with Mode 0 and Mode 3.

Watchdog could reset chip from unexpected hang up or malfunction.

8.1.1 Timer0/1

8.1.1.1 Mode

(1) Mode 0 (System Clock Mode)

In Mode 0, pclk is employed as clock source.

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive edge of pclk from preset initial Tick value. Generally the initial Tick value is set to 0.

Once current Timer Tick value matches the preset Timer Compare (i.e. timing value), an interrupt is generated, Timer re-starts counting from 0 or continues counting and Timer status is updated.

Steps of setting Timer0 for Mode 0 is taken as an example.

Step 1 Set initial Tick value of Timer0

Set Initial value of Tick via registers TMR_TICK0_0~TMR_TICK0_3, from lowest byte to highest byte respectively. It's recommended to clear initial Timer Tick value to 0.

Step 2 Set Compare value of Timer0

Set registers TMR_COMP0_0~TMR_COMP0_3, from lowest byte to highest byte respectively.

Step 3 Set Timer0 to Mode 0 and enable Timer0

Set register TMR_CTRL0 [1:0] to 2'b00 to select Mode 0; set register TMR_CTRL0[3] to 0 to wrap the tick value; set register TMR_CTRL3[0] to 1 to enable timer0 mode interrupt mask. Meanwhile set TMR_CTRL0 [2] to 1'b1 to enable Timer0. Timer0 starts counting upward, and Tick value is increased by 1 on each positive edge of pclk. When the Tick value is equal to the target Tick value set by registers TMR_COMP0_0~TMR_COMP0_3, it generates interrupt signal and Tick re-starts counting from 0; if the register TMR_CTRL0[3] is set to 1, the Tick continues counting.

Step 4 Interrupt processing

After entering the interrupt, if Timer0 is not needed to continue working, TMR_CTRL0 [2] can be set to 1'b0 to disable Timer0, and the interrupt status bit can be cleared by writing a 1 to register TMR_STATS1[0].

(2) Mode 1 (GPIO Trigger Mode)

In Mode 1, GPIO is employed as clock source. The "**M0**"/"**M1**"" register specifies the GPIO which generates counting signal for Timer0/Timer1 (details refer to the note on Polarity above Table 11-5).

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive/negative edge of GPIO (The "**Polarity**" register specifies the GPIO edge, details refer to the note on Polarity above Table 11-5) from preset initial Tick value. Generally the initial Tick value is set to 0.

Once current Timer Tick value matches the preset Timer Compare (i.e. timing value), an interrupt is generated. Timer re-starts counting from 0 or continues counting and Timer status is updated.

Steps of setting Timer1 for Mode 1 is taken as an example.

Step 1 Set initial Tick value of Timer1

Set Initial value of Tick via registers TMR_TICK1_0~TMR_TICK1_3, from lowest byte to highest byte respectively. It's recommended to clear initial Timer Tick value to 0.

Step 2 Set Compare value of Timer1

Set registers TMR_COMP1_0~TMR_COMP1_3, from lowest byte to highest byte respectively.

Step 3 Select GPIO source and edge for Timer1

Select certain GPIO to be the clock source via setting "M1" register (details refer to the note on Polarity above Table 11-5).

Select positive edge or negative edge of GPIO input to trigger Timer1 Tick increment via setting "Polarity" register (details refer to the note on Polarity above Table 11-5).

Step 4 Set Timer1 to Mode 1 and enable Timer1

Set TMR_CTRL0 [5:4] to 2'b01 to select Mode 1; set register TMR_CTRL0[7] to 0 to wrap the tick value; set register TMR_CTRL3[3] to 1 to enable timer1 mode interrupt mask. Meanwhile set TMR_CTRL0 [6] to 1'b1 to enable Timer1. Timer1 starts counting upward, and Timer1 Tick value is increased by 1 on each positive/ negative (specified during the 3rd step) edge of GPIO. When the Tick value is equal to the Tick target value set in the TMR_COMP1_0~TMR_COMP1_3 registers, an interrupt signal is generated and the Tick starts counting again from 0. The Tick continues counting if register TMR_CTRL0[7] is set to 1.

Step 5 Interrupt processing

After entering the interrupt, if Timer1 is not needed to continue working, TMR_CTRL0 [6] can be set to 1'b0 to disable Timer1, and the interrupt status bit can be cleared by writing a 1 to register TMR_STATS1[3].

(3) Mode 2 (GPIO Pulse Width Mode)

In Mode 2, pclk is employed as the unit to measure the width of GPIO pulse. The "**M0**"/"**M1**" register specifies the GPIO which generates control signal for Timer0/Timer1 (details refer to the note on Polarity above Table 11-5).

After Timer is enabled, Timer Tick is triggered by a positive/negative edge (The "**Polarity**" register specifies the GPIO edge, details refer to the note on Polarity above Table 11-5) of GPIO pulse. Then Timer Tick (i.e. counting value) is increased by 1 on each positive edge of system clock from preset initial Tick value. Generally the initial Tick value is set to 0.

While a negative/positive edge of GPIO pulse is detected, an interrupt is generated and timer stops counting. The GPIO pulse width could be calculated in terms of tick count and period of pclk.

Steps of setting Timer1 for Mode 2 is taken as an example.

Step 1 Set initial Timer1 Tick value

Set Initial value of Tick via registers TMR_TICK1_0~TMR_TICK1_3, from lowest byte to highest byte respectively. It's recommended to clear initial Timer Tick value to 0.

Step 2 Select GPIO source and edge for Timer1

Select certain GPIO to be the clock source via setting "M1" register (details refer to the note on Polarity above Table 11-5).

Select positive edge or negative edge of GPIO input to trigger Timer1 counting start via setting "Polarity" register (details refer to the note on Polarity above Table 11-5).

Step 3 Set Timer2 to Mode 2 and enable Timer1

Set TMR_CTRL0 [5:4] to 2'b10 to select Mode 2; set register TMR_CTRL3[3] to 1 to enable timer1 mode interrupt mask. Meanwhile set TMR_CTRL0 [6] to 1'b1 to enable Timer1. Timer1 Tick is triggered by a positive/ negative (specified during the 2nd step) edge of GPIO pulse. Timer1 starts counting upward and Timer1 Tick value is increased by 1 on each positive edge of pclk.

While a negative/positive edge of GPIO pulse is detected, an interrupt is generated and Timer1 tick stops.

Step 4 Interrupt processing

After entering the interrupt, if Timer1 is not needed to continue working, TMR_CTRL0 [6] can be set to 1'b0 to disable Timer1, and the interrupt status bit can be cleared by writing a 1 to register TMR_STATS1[3].

Step 5 Read current Timer1 Tick value to calculate GPIO pulse width

Read current Timer1 Tick value.

Then GPIO pulse width is calculated as follows:

GPIO Pulse Width = System Clock Period *(Current Timer1 Tick - Initial Timer1 Tick)

For initial Timer1 Tick value is set to the recommended value of 0, then:

GPIO Pulse Width = System Clock Period * Current Timer1 Tick

(4) Mode 3 (Tick Mode)

In Mode 3, pclk is employed. After Timer is enabled, Timer Tick starts counting upward, and Timer Tick value is increased by 1 on each positive edge of pclk.

This mode could be used as time indicator. There is no interrupt generated. Timer Tick keeps rolling from 0 to 0xffffffff. When Timer tick overflows, it returns to 0 and starts counting upward again.

Steps of setting Timer0 for Mode 3 is taken as an example.

Step 1 Set initial Tick value of Timer0

Set Initial value of Tick via TMR_TICK0_1 ~TMR_TICK0_3, from lowest byte to highest byte respectively.

Step 2 Set Timer0 to Mode 3 and enable Timer0

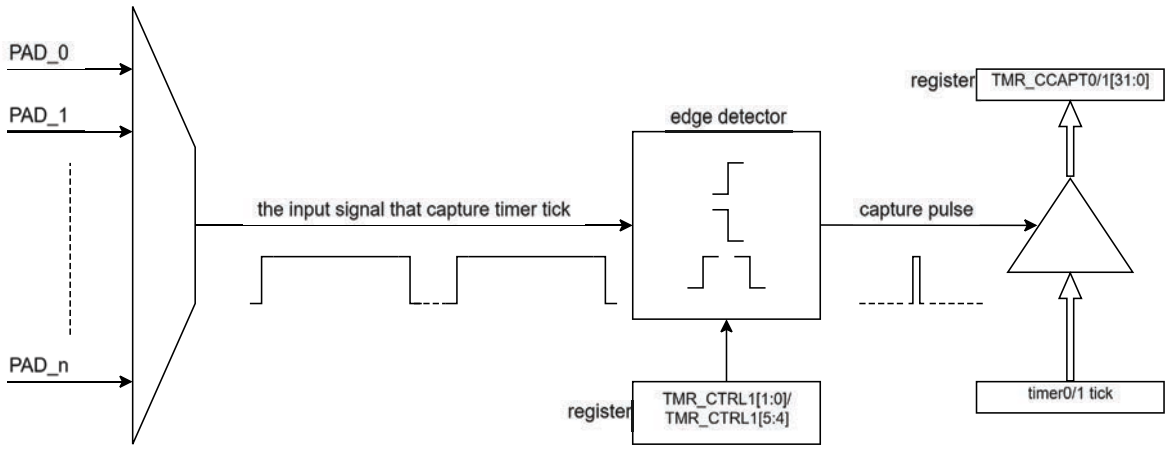
Set TMR_CTRL0[1:0] to 2'b11 to select Mode 3, meanwhile set address TMR_CTRL0[2] to 1'b1 to enable Timer0. Timer0 Tick starts to roll.

Step 3 Read current Timer0 Tick value

Current Timer0 Tick value can be read from TMR_TICK0_1 ~TMR_TICK0_3.

8.1.1.2 Input Capture Function

Figure 8-1 Input Capture of Timer0/1



The input signal of GPIO is used as the capture_in signal of timer0/1, and the edge of capture_in signal is selected as the trigger pulse of capture through registers TMR_CTRL1[1:0]/TMR_CTRL1[5:4]. When the capture trigger pulse is generated, the current tick value of timer0/1 is latched into the registers TMR_CCAPT0/1[31:0]. If the register TMR_CTRL3[1]/TMR_CTRL3[4] is set to 1 before that, timer0/1 capture interrupt is generated, and the interrupt status can be viewed by reading register TMR_STATS1[1]/TMR_STATS1[4], and the interrupt status can be cleared by writing 1 to TMR_STATS1[1]/TMR_STATS1[4].

Here are two examples to illustrate how input capture works:

1. case1

Configure the registers TMR_CTRL0[1:0]/TMR_CTRL0[5:4] to be 2'b00, timer0/1 works in mode0;

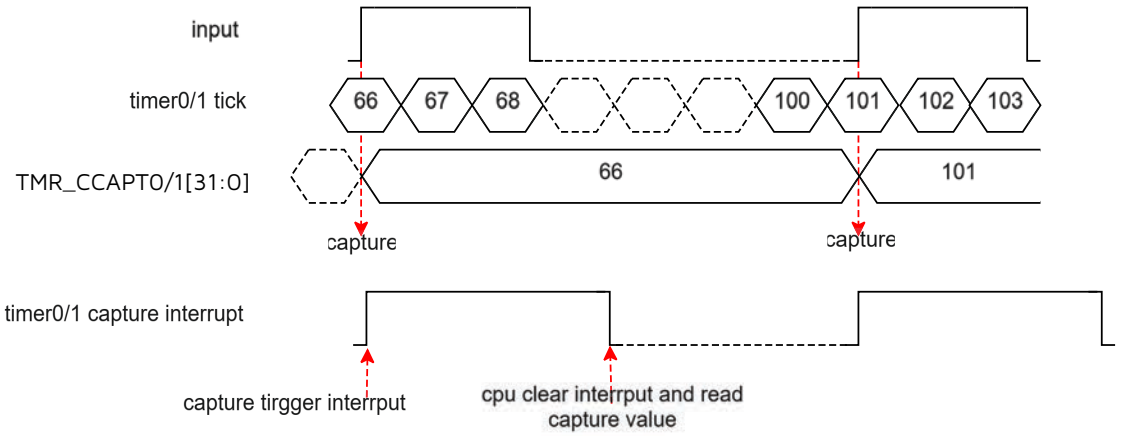
Configure the registers TMR_CTRL1[1:0]/TMR_CTRL1[5:4] to 2'b00, the rising edge of capture_in signal triggers capture;

After timer0/1 is enabled, it starts counting from the preset tick value and adds 1 on the rising edge of pclk;

The rising edge of capture_in signal triggers capture, and the tick value is latched into registers TMR_CCAPT0/1[31:0]; if registers TMR_CTRL3[1]/TMR_CTRL3[4] are set to 1 before this, a timer0/1 capture interrupt is generated;

The CPU handles timer0/1 capture interrupt: read the value of register TMR_CCAPT0/1[31:0], write 1 to register TMR_STATS1[1]/TMR_STATS1[4] to clear this interrupt status.

Figure 8-2 Input Capture Principle case1



2. case2

Configure the registers TMR_CTRL0[1:0]/TMR_CTRL0[5:4] to 2'b00, timer0/1 work in mode0;

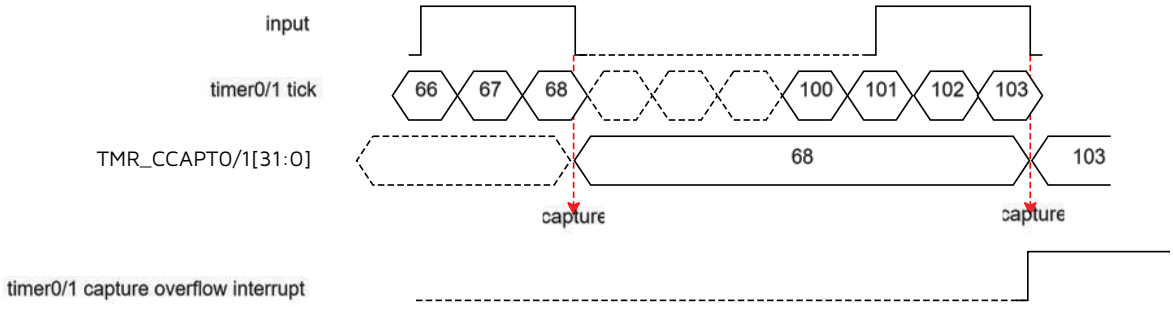
Configure the registers TMR_CTRL1[1:0]/TMR_CTRL1[5:4] to 2'b01, the falling edge of capture_in signal triggers capture;

After timer0/1 is enabled, it starts counting from the preset tick value and adds 1 on the rising edge of pclk;

The falling edge of capture_in signal triggers capture and the tick value is latched into register TMR_CCAPT0/1[31:0];

If the value of register TMR_CCAPT0/1[31:0] is not read away, a new capture is triggered, the old value of register TMR_CCAPT0/1[31:0] is overwritten by the new one. If the register TMR_CTRL3[6]/TMR_CTRL3[7] is set to 1 before that, it triggers timer0/1 capture overflow interrupt, read register TMR_STATS1[6]/TMR_STATS1[7] to check the interrupt status, and write 1 to register TMR_STATS1[6]/TMR_STATS1[7] to clear the interrupt status;

Figure 8-3 Input Capture Principle case2



8.1.2 Watchdog

In watchdog, pclk is employed as clock source.

Watchdog Capture has 24bits, which consists of WT_TARGET_1~WT_TARGET_3 as byte 1 ~byte 3. Watchdog can reset the chip when TMR_STATS0[0] is set to 1, but watchdog does not work in low-power mode.

Step 1 Set WT_TARGET_1~WT_TARGET_3

Set registers WT_TARGET_1~WT_TARGET_3, from lowest byte to highest byte respectively.

Step 2 Enable Watchdog

Set TMR_WD_EN[0] to 1'b1 to enable Watchdog.

During normal working condition, TMR_STATS0[1] need write 1 to clean the watchdog before the watchdog hits WT_TARGET3-1, or it reboots the whole chip, and the TMR_STATS0[0] is assert to 1, this bit is clean when write 1.

8.1.3 Timer Register Description

The Timer related register are listed in table below. The base address for the following registers is 0x80140140.

Table 8-1 Registers for Timer 0 ~ Timer 1

Address Offset	Name	Type	Description	Reset Value
0x00	TMR_CTRL0	R/W	[1:0] tmr0m_sel, 0:tmr0m0,using pclk 1:tmr0m1, count gpio2risc0 posedge 2:tmr0m2 count gpio2risc0 high width 3:tmr0m3,tick [2] tmren0, Timer0 enable [3] tm0_nowrap 1: Timer0 tick continues to count when Timer0 tick value is equal to the Timer0 tick target value set by TMR_COMP0_0~TMR_COMP0_3 registers; 0: Tick starts counting again from 0. [5:4] tmr1m_sel, 0:tmr1m0,using pclk 1:tmr1m1, count gpio2risc1 posedge 2:tmr1m2 count gpio2risc1 high width 3:tmr1m3,tick [6] tmren1, Timer2 enable [7] tm1_nowrap 1: Timer1 tick continues to count when Timer1 tick value is equal to the Timer1 tick target value set by TMR_COMP0_0~TMR_COMP0_3 registers; 0: Tick starts counting again from 0.	0x0

Address Offset	Name	Type	Description	Reset Value
0x01	TMR_CTRL 1	R/W	[1:0] tmr0_capt_mode, 2'b00: capture_in signal rising edge triggers timer0 capture 2'b01: capture_in signal falling edge triggers timer0 capture 2'b10: capture_in signal rising/falling edge trigger timer0 capture 2'b11: reserve [3:2] reserve [5:4] tmr1_capt_mode, 2'b00: capture_in signal rising edge triggers timer1 capture 2'b01: capture_in signal falling edge triggers timer1 capture 2'b10: capture_in signal rising/falling edge trigger timer1 capture 2'b11: reserve [7:6] reserve	0x0
0x02	TMR_CTRL 2	R/W	[3:0] reserve, [4] tmr0_capt_en, 0: disable timer0 capture function 1: enable timer0 capture function [5] reserve, [6] tmr1_capt_en, 0: disable timer1 capture function 1: enable timer1 capture function [7] reserve,	0x0
0x03	TMR_CTRL 3	R/W	[2:0] tmr0_irq_mask [0]: mode_irq_mask, timer0 mode interrupt mask [1]: capt_irq_mask, timer0 capture interrupt mask [2]:reserve [5:3] tmr1_irq_mask [3]: mode_irq_mask, timer1 mode interrupt mask [4]: capt_irq_mask, timer1 capture interrupt mask [5]: reserve [6]: tmr0_ov_irq_mask, timer0 capture overflow interrupt mask [7]: tnr1_ov_irq_mask, timer1 capture overflow interrupt mask	0x0
0x04	TMR_COMP0_0	R/W	comp0[7:0] Byte 0 of timer0 compare	0x0

Address Offset	Name	Type	Description	Reset Value
0x05	TMR_COMP0_1	R/W	comp0[15:8] Byte 1 of timer0 compare	0x0
0x06	TMR_COMP0_2	R/W	comp0[23:16] Byte 2 of timer0 compare	0x0
0x07	TMR_COMP0_3	R/W	comp0[31:24] Byte 3 of timer0 compare	0x0
0x08	TMR_COMP1_0	R/W	comp1[7:0] Byte 0 of timer1 compare	0x0
0x09	TMR_COMP1_1	R/W	comp1[15:8] Byte 1 of timer1 compare	0x0
0x0a	TMR_COMP1_2	R/W	comp1[23:16] Byte 2 of timer1 compare	0x0
0x0b	TMR_COMP1_3	R/W	comp1[31:24] Byte 3 of timer1 compare	0x0
0x0d	WT_TARGET_1	R/W	watchdog_target2[15:8] Byte 1 of watchdog target value watchdog_target[15:8] Byte 0 of watchdog target value is fixed to 0x00, watchdog target period: 0x100/(sys_clk.pclk*1000)ms ~ 0xffffff00/(sys_clk.pclk*1000)ms	0x0
0x0e	WT_TARGET_2	R/W	watchdog_target2[23:16] Byte 2 of watchdog target value	0x0
0x0f	WT_TARGET_3	R/W	watchdog_target2[31:24] Byte 3 of watchdog target value	0x0
0x10	TMR_TICK0_0	R/W	ticko[7:0] Byte 0 of timer0 ticker	0x0
0x11	TMR_TICK0_1	R/W	ticko[15:8] Byte 1 of timer0 ticker	0x0
0x12	TMR_TICK0_2	R/W	ticko[23:16] Byte 2 of timer0 ticker	0x0
0x13	TMR_TICK0_3	R/W	ticko[31:24] Byte 3 of timer0 ticker	0x0
0x14	TMR_TICK1_0	R/W	tick1[7:0] Byte 0 of timer1 ticker	0x0

Address Offset	Name	Type	Description	Reset Value
0x15	TMR_TICK1_1	R/W	tick1[15:8] Byte 1 of timer1 ticker	0x0
0x16	TMR_TICK1_2	R/W	tick1[23:16] Byte 2 of timer1 ticker	0x0
0x17	TMR_TICK1_3	R/W	tick1[31:24] Byte 3 of timer1 ticker	0x0
0x18	TMR_CCAPT0_0	R	capt0[7:0], Byte 0 of timer0 capture	0x0
0x19	TMR_CCAPT0_1	R	capt0[15:8], Byte 1 of timer0 capture	0x0
0x1a	TMR_CCAPT0_2	R	capt0[23:16], Byte 2 of timer0 capture	0x0
0x1b	TMR_CCAPT0_3	R	capt0[31:24], Byte 3 of timer0 capture	0x0
0x1c	TMR_CCAPT1_0	R	capt1[7:0], Byte 0 of timer1 capture	0x0
0x1d	TMR_CCAPT1_1	R	capt1[15:8], Byte 1 of timer1 capture	0x0
0x1e	TMR_CCAPT1_2	R	capt1[23:16], Byte 2 of timer1 capture	0x0
0x1f	TMR_CCAPT1_3	R	capt1[31:24], Byte 3 of timer1 capture	0x0
0x20	TMR_STAS0	R/W	[0] wd_sts, watchdog status, W1C, when Watchdog tick value is equal to Watchdog tick target value set by WT_TARGET_1~WT_TARGET_3 registers, this bit is set to 1 and reboot the whole chip; when 32k Watchdog tick value equals to the target value, the hardware is reset, after reboot, vbus watchdog tick value equals to the target value, when in deep or deep retention mode, this bit is set to 0; this bit remains unchanged after the Watchdog and system reboot back. [1] wd_cnt_clr, clear wd_cnt [6:2] reserved [7] software_irq, write 1 to generate software interrupt, write 0 to clear software interrupt	0x0

Address Offset	Name	Type	Description	Reset Value
0x21	TMR_STATS1	W1C	[0] tmr0_mode_irq, timer0 interrupt in mode0/mode1/mode2 [1] tmr0_capt_irq, timer0 capture interrupt [2] reserve [3] tmr1_mode_irq, timer1 interrupt in mode0/mode1/mode2 [4] tmr1_capt_irq, timer1 capture interrupt [5] reserve [6] tmr0_capt_ov_irq, timer0 capture overflow interrupt [7] tmr1_capt_ov_irq, timer1 capture overflow interrupt	0x0
0x22	TMR_WD_EN	R/W	[0]: wd_en, watch_dog enable [1]: pem_event_en, [7:2] reserve	0x0
0x23	TMR_PEM_TASK_EN	R/W	[0]: pem_task0_en, [1]: pem_task1_en, [2]: pem_task2_en, [3]: pem_task3_en, [4]: pem_task4_en, [5]: pem_task5_en, [6]: pem_task6_en, [7]: pem_task7_en,	0x0

8.2 32K LTimer

The SoC also supports a low frequency (32 kHz) LTIMER in suspend mode or deep sleep mode. This timer can be used as one kind of wakeup source.

In order to avoid the situation of not being able to wake up in low power mode and power on error, a new watch dog function has been added to the 32 kHz timer. And the watch dog function is enabled by default.

The 32K LTimer features include:

- The frequency of the clock source is 32 KHz;
- The width of the LTimer is 32 bits;
- Supports watch dog function;
- Can be used as one of wakeup sources

The corresponding register configuration is as follows.

Table 8-2 32K LTimer Related Registers

Address	Name	Description	Default Value
afe_0x64	status	write 1 to clean the status: [0]:wkup pad [1]:wkup dig [2]:wkup timer [3]:wkup cmp [4]:rsvd [5]:rsvd [6]:rsvd [7]:wakup_vbus (this bit is set to 1 when there is voltage on vbus, write 1 for hardware reset and clear to 0 when vbus rst timer is reset.)	-
afe_0x69	pg_status	1: indicate power down status [0]:zb power status [1]:usb power status [2]:audio power status [4:3] rsvd [5]:sm_busy [6]: vbus_detect (write 1 to disable vbus rst timer, this bit is 1 if vbus is in inserted state and cleared to 0 when usb is unplugged.) [7]: watch_dog status	
afe_0x79	ltimer_watchdog_en	[0]: ltimer_watchdog_en [3:1] rsvd [7:4] rsvd, ltimer_watchdog_v[7:0] is 0x0	1
afe_0x7a	ltimer_watchdog_v[15:8]	[7:0] ltimer_watchdog_v[15:8]	01110001
afe_0x7b	ltimer_watchdog_v[23:16]	[7:0] ltimer_watchdog_v[23:16]	00000010
afe_0x7c	ltimer_watchdog_v[31:24]	[7:0] ltimer_watchdog_v[31:24]	0

The afe_0x79[0] is the ltimer_watchdog enable signal, write 1 to enable the ltimer_watchdog function, write 0 to disable the ltimer_watchdog.

The afe_0x7a to afe_0x7c combined into ltimer_watchdog_v[31:8] is the target value corresponding to the 32k timer reset for the entire system. The time range that can be set is: 8ms ~ 134217720 ms. The default value is 0x271, which is 0x27100 for 32k cycle, corresponding to 5 seconds. (After power up, if the firmware does not modify this value in time, it resets the whole system after 5 seconds).

The afe_0x69[7] is the status of watch dog, write 1 to clear the status.

The difference between this watch dog and the regular watch dog is that clearing the dog is achieved by modifying the ltimer_watchdog_v value. Because the 32k timer is reused, the calculator keeps adding until

the count reaches the maximum value and then continues from 0. When modifying the value, the enable signal needs to be disabled first.

8.3 System Timer

The SoC also supports a System Timer, the clock frequency for System Timer is fixed as 24 MHz irrespective of system clock. The System Timer supports Input Capture function (refers to the input capture function description in 8.1.1 Timer0/1).

In suspend mode, both System Timer and Timer0 ~ Timer1 stop counting, and 32K Timer starts counting. When the chip restores to active mode, Timer0 ~ Timer1 reset tick to 0; In contrast, System Timer continues counting from an adjusted number which is a sum of the number when it stops and an offset calculated from the counting value of 32K Timer during suspend mode.

8.3.1 Enable Mode

There are two ways to enable sys_timer: manual mode and auto mode.

Manual mode:

- enable sys_timer: SYS_TIMER_CTRL[1] is set to 1.
- disable sys_timer: SYS_TIMER_CTRL[1] is set to 0.

Auto mode:

- enable sys_timer: First, set SYS_TIMER_CTRL[2] to 1, enable timer_auto. Then, when SYS_TIMER_UP[1] is set to 0, enable sys_timer automatically during write operation to SYS_TIMER0~SYS_TIMER3 registers. When SYS_TIMER_UP[1] is set to 1, sys_timer is automatically enabled at the first rising edge of clk_32k after a write operation is performed to SYS_TIMER0~SYS_TIMER3 registers.
- disable sys_timer: first, set SYS_TIMER_CTRL[2] to 1, enable timer_auto; then, set SYS_TIMER_CTRL[2] to 1, enable timer_auto.

8.3.2 Irq_level Interrupt

The irq_level interrupt condition is: irq_level <= sys_timer < irq_level+2^26.

Interrupts are generated in the following cases.

1. **Case 1**: SYS_TIMER_IRQ_MASK[2] is set to 0 or SYS_TIMER_UP[1] is set to 0. When sys_timer meets the irq_level interrupt condition, the irq_level_pulse is generated immediately. If SYS_TIMER_IRQ_MASK[0] is set to 1 before irq_level_pulse, irq_level interrupt is triggered. If SYS_TIMER_IRQ_MASK[0] is set to 1 after irq_level_pulse, irq_level interrupt is not triggered.
2. **Case 2**: SYS_TIMER_IRQ_MASK[2] is set to 1,and SYS_TIMER_UP[1] is set to 1, but there is no write operation to SYS_TIMER0~SYS_TIMER3 registers. When sys_timer meets the irq_level interrupt condition, then irq_level_pulse is generated immediately. If SYS_TIMER_IRQ_MASK[0] is set to 1 before irq_level_pulse, irq_level interrupt is triggered. If SYS_TIMER_IRQ_MASK[0] is set to 1 after irq_level_pulse, irq_level interrupt is not triggered.
3. **Case 3**: SYS_TIMER_IRQ_MASK[2] is set to 1 and SYS_TIMER_UP[1] is set to 1. Write operation is performed to SYS_TIMER0~SYS_TIMER3 registers. Between the moment of the write operation of the SYS_TIMER0~SYS_TIMER3 registers and the moment of the rising edge of clk_32k, no irq_level_pulse is

generated even if sys_timer meets the irq_level interrupt condition. Outside this time interval, sys_timer meets the irq_level interrupt condition before generating an irq_level_pulse. An irq_level interrupt is triggered if SYS_TIMER_IRQ_MASK[0] is set to 1 before the irq_level_pulse. If SYS_TIMER_IRQ_MASK[0] is set to 1 after irq_level_pulse, irq_level interrupt is not triggered.

4. **Case 4**: The irq_level interrupt is a pulse trigger, that is, irq_level_pulse triggers irq_level interrupt. In case 1 to case 3, if SYS_TIMER_IRQ_MASK[0] is set to 1 after irq_level_pulse, irq_level_pulse does not trigger irq_level interrupt, even if the irq_level interrupt condition is met after SYS_TIMER_IRQ_MASK[0] is set to 1. Therefore, when sys_timer is in the above situation, you can set SYS_TIMER_IRQ_MASK[3] to 1, and then change the value of irq_level. If the changed value of irq_level meets the condition of irq_level interrupt, it generates an irq_level_pulse again, which triggers the irq_level interrupt. level interrupt.

8.3.3 Calibration Function

Since the RC32k clock is not accurate, it is needed to calculate the actual RC32k frequency by using the accurate system timer clock.

The Calibration process is as follows:

SYS_TIMER_CTRL[3] is set to 1, wait for the first clk_32k rising edge, and then enable the calibration;

After calibration is enabled, at the rising edge of system timer clock, the counter cal_cnt is added 1; at the rising edge of clk_32k, the counter ccnt is added 1;

When ccnt equals to 2^(16-SYS_TIMER_CTRL[7:4]), ccnt is reset to 0, cal_cnt is reset to 1, and the value of cal_cnt at this time is latched into CAL_LATCH0~CAL_LATCH3 registers, that is

2^(16 - SYS_TIMER_CTRL[7:4]) × TRC32K = cal_latch × T(system_timer_clock);

irq_cal interrupt:

After SYS_TIMER_IRQ_MASK[0] is set to 1, and calibration is enabled, irq_cal interrupt is generated when ccnt equals 2^(16-SYS_TIMER_CTRL[7:4]).

8.3.4 32K_Timer Set/Read Function

32k_Timer set flow:

After SYS_TIMER_CTRL[0] is set to 1, SYS_TIMER_ST[3] is set to 1 to start the 32k_Timer set, during which the system timer module synchronizes the contents of the 32K_TIMER_SET0~32K_TIMER_SET3 registers to the 32ktimer counter;

Reading SYS_TIMER_ST[3] as 1 indicates that the 32k_Timer set is in progress;

Reading SYS_TIMER_ST[3] as 0 indicates that the 32k_Timer set is finished.

32k_Timer read flow:

First SYS_TIMER_ST[5] is set to 1 to clear the state of this bit;

After SYS_TIMER_CTRL[0] is set to 0, wait for the first clk_32k rising edge to arrive, then start 32k_Timer read, during this period, the system timer module synchronizes the 32ktimer counter value to the 32K_TIMER_READ0~32K_TIMER_READ3 registers.

Reading SYS_TIMER_ST[6] as 1 indicates that the 32k_Timer read is in progress;

Reading SYS_TIMER_ST[6] as 0 indicates that the 32k_Timer read is finished;

8.3.5 Update on 32K clock

When SYS_TIMER_UP[0] is set to 0: whenever sys_timer[2:0] is read as 0, the tick value of sys_timer is latched into SYS_TIMER0~SYS_TIMER3 registers, sys_timer[2:0] are fixed to 0.

When SYS_TIMER_UP[0] is set to 1: whenever the rising edge of clk_32k, the tick value of sys_timer is latched to SYS_TIMER0~SYS_TIMER3 registers.

8.3.6 System Timer Register Description

The system timer related registers are listed in table below. The base address for the registers is 0x80140200.

Table 8-3 System Timer Related Registers

Address Offset	Name	Type	Description	Reset Value
0x00	SYS_TIMER0	R/W	sys_timer[7:0] when reading this byte, the lower 3bits are fixed to 0	0x00
0x01	SYS_TIMER1	R/W	sys_timer[15:8]	0x00
0x02	SYS_TIMER2	R/W	sys_timer[23:16]	0x00
0x03	SYS_TIMER3	R/W	sys_timer[31:24]	0x00
0x04	IRQ_LEVEL0	R/W	irq_level[7:0]	0xf0
0x05	IRQ_LEVEL1	R/W	irq_level[15:8]	0x0f
0x06	IRQ_LEVEL2	R/W	irq_level[23:16]	0x0f
0x07	IRQ_LEVEL3	R/W	irq_level[31:24]	0x0e
0x08	SYS_TIMER_IRQ_MASK	R/W	[0] irq_sys_timer_mask [1] irq_cal_mask [2] irq_wait [3] trig_past_en [4] capt_irq_mask, sys_timer capture interrupt mask [5] capt_ov_irq_mask, sys_timer capture overflow interrupt mask	0x00
0x09	SYS_TIMER_IRQ	R	[0] irq_sys_timer, W1C [1] irq_cal, W1C [2] capt_irq, W1C [3] capt_ov_irq, W1C	0x00

Address Offset	Name	Type	Description	Reset Value
0x0a	SYS_TIMER_CTRL	R/W	[0] wr_32k, 1:32k write mode; 0:32k read mode [1] timer_en, system timer enable [2] timer_auto, 1: auto mode, 0: manual mode; To ensure accurate sleep time, it is recommended to select auto mode. Writing SYS_TIMER_ST[1] to 1 can stop system timer when using auto mode [3] cal_32k_en, 32k calibration enable [7:4] cal_32k_mode, 32k calibration mode (2^(16-cal_32k_mode)) cycles of 32k clock	0xc1
0x0b	SYS_TIMER_ST	R/W	[1] cmd_stop, write 1, stop system timer when using auto mode [3] cmd_sync, write 1, start 32k count write; R:st_list3(wr_busy) [4] clk_32k, 32k clock read [5] clr_rd_done, clear read 32k update flag; R:st_list5(rd_done) [6] rd_busy, 32k read busy status [7] cmd_set_dly_done, system timer set done status upon next 32k posedge	0x00
0x0c	32K_TIMER_SET0	R/W	32k_timer_set[7:0]	0x00
0x0d	32K_TIMER_SET1	R/W	32k_timer_set[15:8]	0x00
0x0e	32K_TIMER_SET2	R/W	32k_timer_set[23:16]	0x00
0x0f	32K_TIMER_SET3	R/W	32k_timer_set[31:24]	0x00
0x10	32K_TIMER_READ0	R	32k_timer_read[7:0]	0x00
0x11	32K_TIMER_READ1	R	32k_timer_read[15:8]	0x00
0x12	32K_TIMER_READ2	R	32k_timer_read[23:16]	0x00
0x13	32K_TIMER_READ3	R	32k_timer_read[31:24]	0x00
0x14	CAL_LATCH0	R	cal_latch[7:0]	0x00
0x15	CAL_LATCH1	R	cal_latch[15:8]	0x00
0x16	CAL_LATCH2	R	cal_latch[23:16]	0x00
0x17	CAL_LATCH3	R	cal_latch[31:24]	0x00

Address Offset	Name	Type	Description	Reset Value
0x18	SYS_TIMER_UP	R/W	[0] update_upon_32k [1] run_upon_nxt_32k	0x00
0x19	SYS_TIMER_CTRL1	R/W	[1:0]capt_mode 2'b00: capture_in signal rising edge triggers sys_timer capture 2'b01: capture_in signal falling edge triggers sys_timer capture 2'b10: capture_in signal rising/falling edge trigger sys_timer capture 2'b11: reserve [2]capt_en [3]pem_event_en [4]pem_task_0_en [5]pem_task_1_en [6]pem_task_2_en	0x00
0x1c	SYS_TIMER_CAPT_0	R	capt[7:0], Byte 0 of sys_timer capture	0x00
0x1d	SYS_TIMER_CAPT_1	R	capt[15:8], Byte 1 of sys_timer capture	0x00
0x1e	SYS_TIMER_CAPT_2	R	capt[23:16], Byte 2 of sys_timer capture	0x00
0x1f	SYS_TIMER_CAPT_3	R	capt[31:24], Byte 3 of sys_timer capture	0x00

8.4 Platform-Level Machine Timer (PLMT)

8.4.1 Introduction

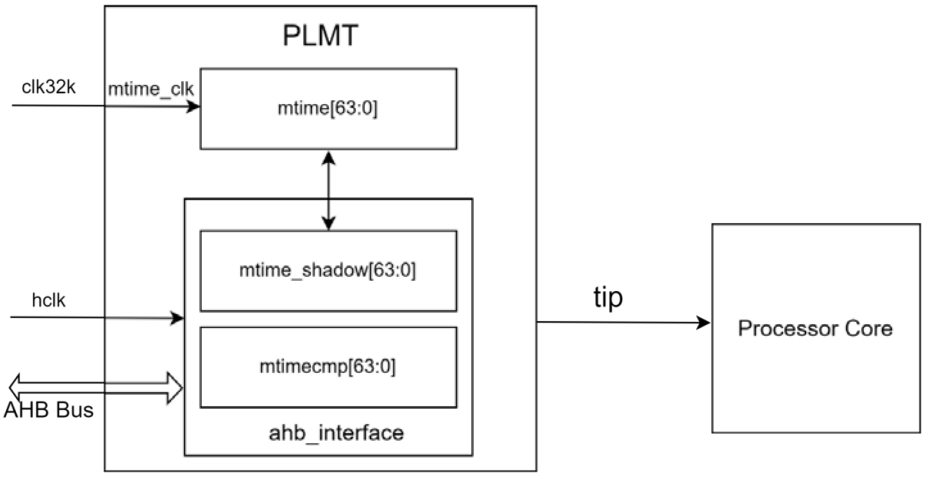
The RISC-V architecture defines a machine timer that provides a real-time counter and generates timer interrupts. Platform-Level Machine Timer (PLMT) is an implementation of the machine timer.

The PLMT supports the following features:

- Supports 64-Bits mtime and mtimecmp
- Supports timer interrupt generation when mtime >= mtimecmp

The PLMT block diagram is shown in the figure below.

Figure 8-4 Block Diagram of PLMT



The PLMT primarily consists of these memory-mapped registers: mtime and mtimecmp. The mtime register is a 64-bit real-time counter clocked by mtime_clk. The source of mtime_clk is clk32k.

The mtimecmp register stores a 64-bit value for comparing with mtime. When the value in mtime is greater than or equal to the value in mtimecmp, the mtip signal is asserted for generating a timer interrupt. When mtimecmp is written, the interrupt is cleared and the mtip signal is deasserted.

The mtime register is driven by mtime_clk, which is assumed to be slower than hclk. The mtime_shadow shadow register is maintained in the hclk domain to reduce the latency of accessing the mtime register in the slow clock domain. The values of mtime and mtime_shadow registers are constantly synchronized such that mtime_shadow maintains the most up-to-date values of mtime. The value in mtime_shadow is instantly returned when reading the mtime register. When writing the mtime register, bus write transactions finish when the values are written to the mtime_shadow register, and PLMT handles the synchronization to mtime in the background.

8.4.2 Access To Mtime

The mtime counter is a 64-bit value and it increments non-stop on every machine timer clock except the first few cycles after its control register updates. But it can only be accessed as two separate 32-bit registers by 32-bit width bus. So, please follow the following programming sequence to make sure the access to mtime is correct.

For Write Mtime sequence:

1. Write zero to mtime[31:0].
2. Write high part of the intended value to mtime[63:32].
3. Write low part of the intended value to mtime[31:0].

For Read Mtime sequence:

1. Read mtime[63:32] and save it to integer variable hi0.
2. Read mtime[31:0] and save it to integer variable lo0.
3. Read mtime[63:32] and save it to integer variable hi1.
4. If hi1 is not equal to hi0, jump to step 1. Otherwise, return ((unsigned long long)hi0 << 32) | lo0;

8.4.3 Access To Mtimecmp

The mtimecmp register is a 64-bit value. But it can only be accessed as two separate 32-bit registers by 32-bit width bus.So, please follow the following programming sequence to avoid spuriously generating an interrupt due to the intermediate value of the mtimecmp register.

For Write Mtimecmp sequence:

1. Write 0xFFFFFFFF to mtimecmp[31:0].
2. Write high part of the intended value to mtimecmp[63:32].
3. Write low part of the intended value to mtimecmp[31:0].

8.4.4 PLMT Register Description

The PLMT related registers are listed in table below. The base address for the following registers is 0xC6000000.

Please note that PLMT supports only 32-bit. Behaviors of 8-bit and 16-bit transfers are UNDEFINED, and these transfers might be ignored as well as result in error responses or unexpected register updates.

Table 8-4 PLMT Related Registers

Offset	Name	Type	Description	Reset Value
0x00	mtime_low	R/W	low part of mtime [31:0]: mtime[31:0]	0x00
0x04	mtime_high	R/W	high part of mtime [31:0]: mtime[63:32]	0x00
0x08	mtimecmp_low	R/W	low part of mtimecmp [31:0]: mtimecmp[31:0]	0xFFFFFFFF
0x0c	mtimecmp_high	R/W	high part of mtimecmp [31:0]: mtimecmp[63:32]	0xFFFFFFFF

9 Trap and PLIC

9.1 Trap

9.1.1 Introduction

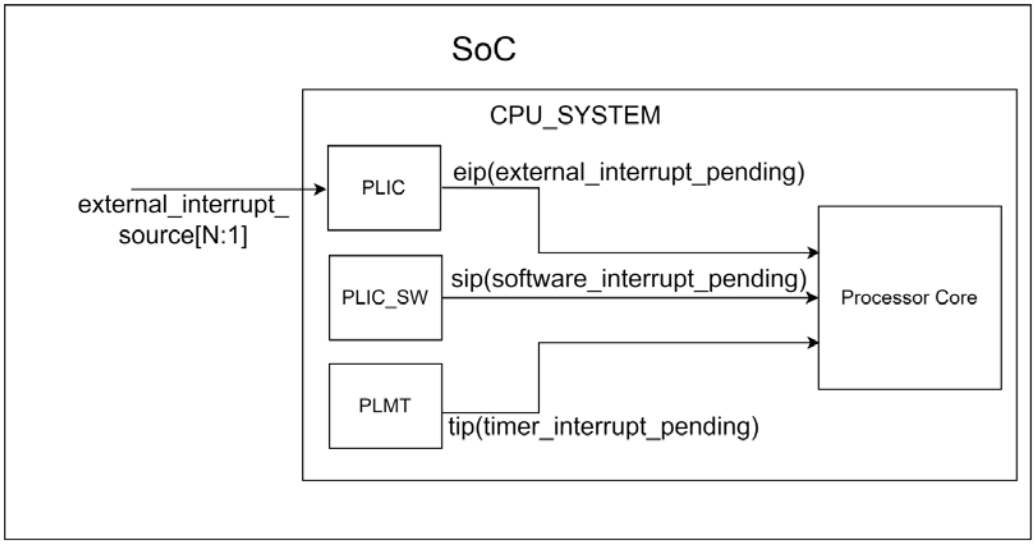
According to the RISC-V Privileged Architecture, a trap is a control flow change of normal instruction execution caused by an interrupt or an exception. An interrupt is a control flow change event initiated by an external source. An exception is a control flow change event generated as a by-product of instruction execution. When a trap happens, the processor stops processing the current flow of instructions, disables interrupts, saves enough states for later resumption, and starts executing a trap handler.

Interrupts can be local or external. The external interrupts are global interrupts that are arbitrated externally by a platform level interrupt controller (PLIC) and the selected external interrupt joins the rest of local interrupts for arbitration to take a trap.

For exceptions, mepc is the PC (Program Counter) of the faulting instruction. For Interrupts, mepc is pointing to the interrupted instruction.

9.1.2 Interrupt

Figure 9-1 Block Diagram of Interrupt



As shown in the above figure, the processor provides three interrupt inputs: platform-level machine timer (PLMT) interrupt, software platform-level interrupt controller (PLIC_SW) interrupt, and platform-level interrupt controller (PLIC) interrupt.

The PLMT interrupt and PLIC_SW interrupt are local interrupts. External interrupts are arbitrated and distributed by PLIC to the processor. Each external interrupt source can be assigned its own priority, and the processor core could select which external interrupt sources it would handle. PLIC routes the highest priority interrupt source to the target processor.

9.1.2.1 Local Interrupts

In addition to external interrupts, the processor may generate internal interrupts for the following events:

- Bus read/write transaction error
- Performance monitor overflow

9.1.2.2 Interrupt Status and Masking

The mip CSR (Control and Status Register of the processor core) contains pending bits of these interrupts, with the mie CSR contains enable bits of the respective interrupts. The processor can selectively enable interrupts by manipulating the mie CSR, or globally disable interrupts by clearing the mstatus.MIE bit.

9.1.2.3 Interrupt Priority

When multiple interrupts are taken at the same time, they are handled under the following order:

Table 9-1 Interrupt Priority

Priority	Interrupt
High	M-mode performance monitor overflow interrupt
↓	M-mode bus read/write transaction error interrupt
	M-mode external interrupt (MEI)
	M-mode software interrupt (MSI)
Low	M-mode timer interrupt (MTI)

9.1.3 Exception

The processor implements the following exceptions.

- Instruction address misaligned exceptions
  - Jump to misaligned addresses
- Instruction access faults
  - Bus errors caused by instruction fetches
- Illegal instructions
  - Unsupported instructions
  - Privileged instructions
  - Accessing non-existent CSRs (Control and Status Registers of the processor core)
  - Accessing privileged CSRs
  - Writing to read-only CSRs
- Breakpoint exceptions
- Load address misaligned exceptions
- Load access faults
  - Bus errors caused by load instructions
- Store/AMO (atomic memory operation) address misaligned exceptions
- Store/AMO access faults
- Environment calls
- Stack overflow/underflow exceptions

9.1.4 Trap Handling

9.1.4.1 Entering the Trap Handler

When a trap occurs, the following operations are applied:

- mepc is set to the current program counter.
- mstatus is updated.
  - The MPP field is set to the current privilege mode.
  - The MPIE field is set to the MIE field.
  - The MIE field is set to 0.
- mcause is updated.
- mtval is updated on any of address-misaligned or access-fault exceptions.
- The privilege mode is changed to M-mode.
- When mmisc_ctl.VEC_PLIC is 0, the program counter is set to the address specified by mtvec.
- When mmisc_ctl.VEC_PLIC is 1, the mtvec register is the base address register of a vector table with 4-byte entries storing addresses pointing to interrupt service routines.
  - mtvec[0] is for exceptions and non-external local interrupts. For these traps, the mcause register records the trap type based on RISC-V definitions.
  - mtvec[i] is for external PLIC interrupt source i triggered through the mip.MEIP pending condition.

9.1.4.2 Returning from the Trap Handler

After handling a trap, the MRET instruction can be executed for returning to the instruction and the privilege context before the trap happened. Alternatively, the trap handler could assign new PC, privilege level and/or interrupt enable status to mepc, mstatus.MPP and mstatus.MPIE before MRET. Specifically, the following operations take place when an MRET instruction is executed:

- The program counter is set to mepc.
- The privilege mode is set to mstatus.MPP.
- mstatus is updated.
  - The MPP field is set to U-mode.
  - The MIE field is set to the MPIE field.
  - The MPIE field is set to 1.

9.1.5 Machine Trap Related CSRs

9.1.5.1 Machine Status

Mnemonic Name: mstatus

Access Mode: Machine

CSR Address: 0x300

Table 9-2 Register Description of mstatus

Name	Bits	Type	Description	Reset
MIE	[3]	R/W	M-mode interrupt enable bit.	0
MPIE	[7]	R/W	MPIE holds the value of the MIE bit prior to a trap.	0
MPP	[12:11]	R/W	MPP holds the privilege mode prior to a trap. 0: User mode 1: Reserved 2: Reserved 3: Machine mode	3

9.1.5.2 Machine Interrupt Enable

Mnemonic Name: mie

Access Mode: Machine

CSR Address: 0x304

Table 9-3 Register Description of mie

Name	Bits	Type	Description	Reset
MSIE	[3]	R/W	M-mode software interrupt enable bit.	0
MTIE	[7]	R/W	M-mode timer interrupt enable bit.	0
MEIE	[11]	R/W	M-mode external interrupt enable bit.	0
BWEI	[17]	R/W	Bus write transaction error local interrupt enable bit. The processor may receive bus errors on store instructions or cache writebacks.	0
PMOVI	[18]	R/W	Performance monitor overflow local interrupt enable bit.	0

9.1.5.3 Machine Trap Vector Base Address

Mnemonic Name: mtvec

Access Mode: Machine

CSR Address: 0x305

This register determines the base address of the trap vector. The least significant 2 bits are hardwired to zeros. When mmisc_ctl.VEC_PLIC is 0 (PLIC is not in the vector mode), this register indicates the entry points for the trap handler and it may point to any 4-byte aligned location in the memory space.

On the other hand, when mmisc_ctl.VEC_PLIC is 1 (PLIC is in the vector mode), this register is the base address of a vector table with 4-byte entries storing addresses pointing to interrupt service routines.

- This register should be aligned to 256-byte boundary.
- mtvec[0] is for exceptions, local interrupts.

- mtvec[i] is for external PLIC interrupt source i triggered through the mip.MEIP pending condition

Table 9-4 Register Description of mtvec

Name	Bits	Type	Description	Reset
Base[31:2]	[31:2]	R/W	Base address for interrupt and exception handlers.	0

9.1.5.4 Machine Exception Program Counter

Mnemonic Name: mepc

Access Mode: Machine

CSR Address: 0x341

This register is written with the virtual address of the instruction that encountered traps when these events occurred.

Table 9-5 Register Description of mepc

Name	Bits	Type	Description	Reset
EPC	[31:0]	R/W	Exception program counter. Bit[0] is hardwired to zero.	0

9.1.5.5 Machine Cause Register

Mnemonic Name: mcause

Access Mode: Machine

CSR Address: 0x342

This register indicates the cause of trap, reset or the interrupt source ID of a vector interrupt. This register is updated when a trap, reset or vector interrupt occurs. When multiple events may cause a trap to be taken with the same mcause value, the value of mdcause records the exact event that causes the trap.

Table 9-6 Register Description of mcause

Name	Bits	Type	Description	Reset
Exception_code	[11:0]	R/W	Exception code	0
Interrupt	[31]	R/W	Interrupt	0

The following table shows the possible values of mcause:

Table 9-7 Possible Values of mcause

Interrupt	Exception Code	Description
1	3	Machine software interrupt
1	7	Machine timer interrupt
1	11	Machine non-vector external interrupt
1	17	Bus read/write transaction error interrupt (M-mode)

Interrupt	Exception Code	Description
1	18	Performance monitor overflow interrupt (M-mode)
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	11	Environment call from M-mode
0	32	Stack overflow exception
0	33	Stack underflow exception

The following tables show the possible values of mcause after reset:

Table 9-8 Possible values of mcause after reset

Interrupt	Exception Code	Description
0	0	Initial value when the processor comes out of reset

The following tables show the possible values of mcause after vector interrupt:

Table 9-9 Possible values of mcause after vector interrupt

Interrupt	Exception Code	Description
0	Interrupt source ID	Interrupt source ID when a vector interrupt occurs

9.1.5.6 Machine Trap Value

Mnemonic Name: mtval

Access Mode: Machine

CSR Address: 0x343

This register is updated when a trap is taken to M-mode. The updated value is dependent on the cause of traps:

- For Hardware Breakpoint exceptions, Address Misaligned exceptions, or Access Fault exceptions, it is the effective faulting addresses.

- For illegal instruction exceptions, the updated value is the faulting instruction.
- For other exceptions, mtval is set to zero.

For instruction-fetch access faults, this register is updated with the address pointing to the portion of the instruction that caused the fault, while the mepc register is updated with the address pointing to the beginning of the instruction.

Table 9-10 Register Description of mtval

Name	Bits	Type	Description	Reset
mtval	[31:0]	R/W	Exception-specific information for software trap handling	0

9.1.5.7 Machine Interrupt Pending

Mnemonic Name: mip

Access Mode: Machine

CSR Address: 0x344

Table 9-11 Register Description of mip

Name	Bits	Type	Description	Reset
MSIP	[3]	RO	M-mode software interrupt pending bit.	0
MTIP	[7]	RO	M-mode timer interrupt pending bit.	0
MEIP	[11]	RO	M-mode external interrupt pending bit.	0
BWEI	[17]	R/W	Bus write transaction error local interrupt pending bit. The processor may receive bus errors on store instructions or cache writebacks.	0
PMOVI	[18]	R/W	Performance monitor overflow local interrupt pending bit.	0

9.1.5.8 Machine Detailed Trap Cause

Mnemonic Name: mdcause

Access Mode: Machine

CSR Address: 0x7c9

Table 9-12 Register Description of mdcause

Name	Bits	Type	Description	Reset
mdcause	[2:0]	R/W	This register further disambiguates causes of traps recorded in the mcause register. See the below for details.	0

Table 9-13 Detailed mdcause meaning in different mcause condition

mcause condition	mdcause value	Meaning
mcause == 1 (Instruction access fault)	0	Reserved
	1	Reserved
	2	PMP instruction access violation
	3	Reserved
	4	Reserved
mcause == 2 (Illegal instruction)	0	The actual faulting instruction is stored in the mtval CSR.
	1	FP (Floating-Point) disabled exception
mcause == 5 (Load access fault)	0	Reserved
	1	Reserved
	2	PMP load access violation
	3	Bus error
	4	Misaligned address
	5	Reserved
	6	Reserved
	7	Reserved
mcause == 7 (Store access fault)	0	Reserved
	1	Reserved
	2	PMP store access violation
	3	Bus error
	4	Misaligned address
	5	Reserved
	6	Reserved
	7	Reserved

9.1.5.9 Machine Miscellaneous Control Register

Mnemonic Name: mmisc_ctl

Access Mode: Machine

CSR Address: 0x7d0

Table 9-14 Register Description of mdcause

Name	Bits	Type	Description	Reset
VEC_PLIC	[1]	R/W	Select the operation mode of PLIC: 0: Non-Vector mode; 1: Vector mode; Please note that both this bit and PLIC_FEN.VECTORED in PLIC should be turned on for the vectored interrupt support to work correctly.	0

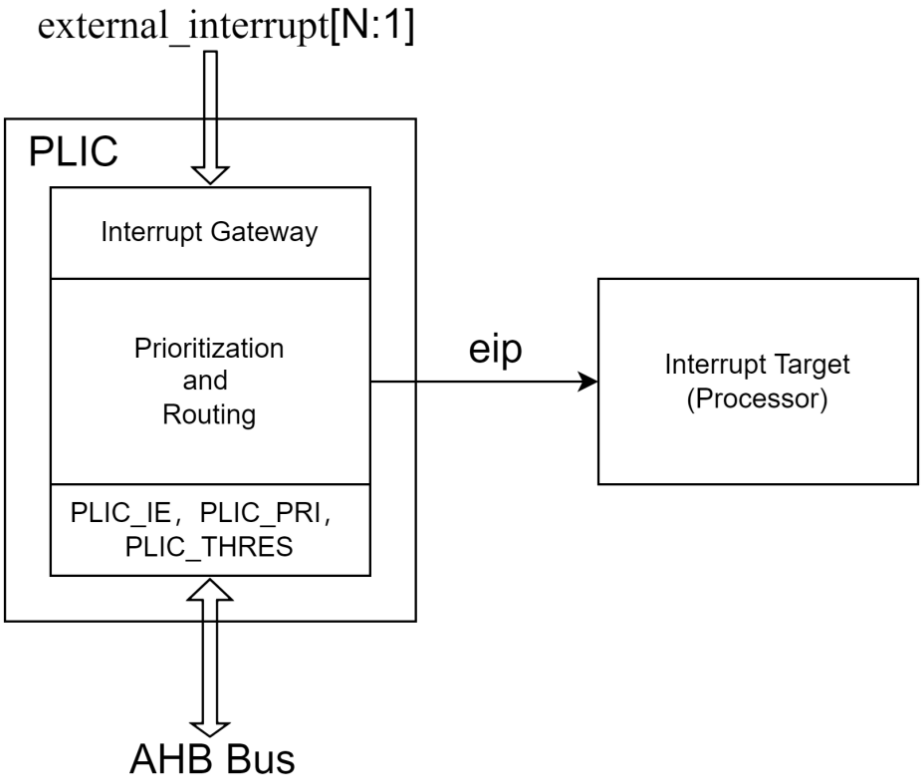
9.2 Platform-Level Interrupt Controller (PLIC)

9.2.1 Introduction

The SoC embeds a Platform-Level Interrupt Controller (PLIC) prioritizes and distributes global interrupts. It is compatible with RISC-V PLIC with the following features:

- Number of interrupts: 52
- Programmable interrupt priority: 1/2/3
- Preemptive priority interrupt extension
- Vectored interrupt extension
- Software-programmable interrupt generation

Figure 9-2 Block Diagram of PLIC



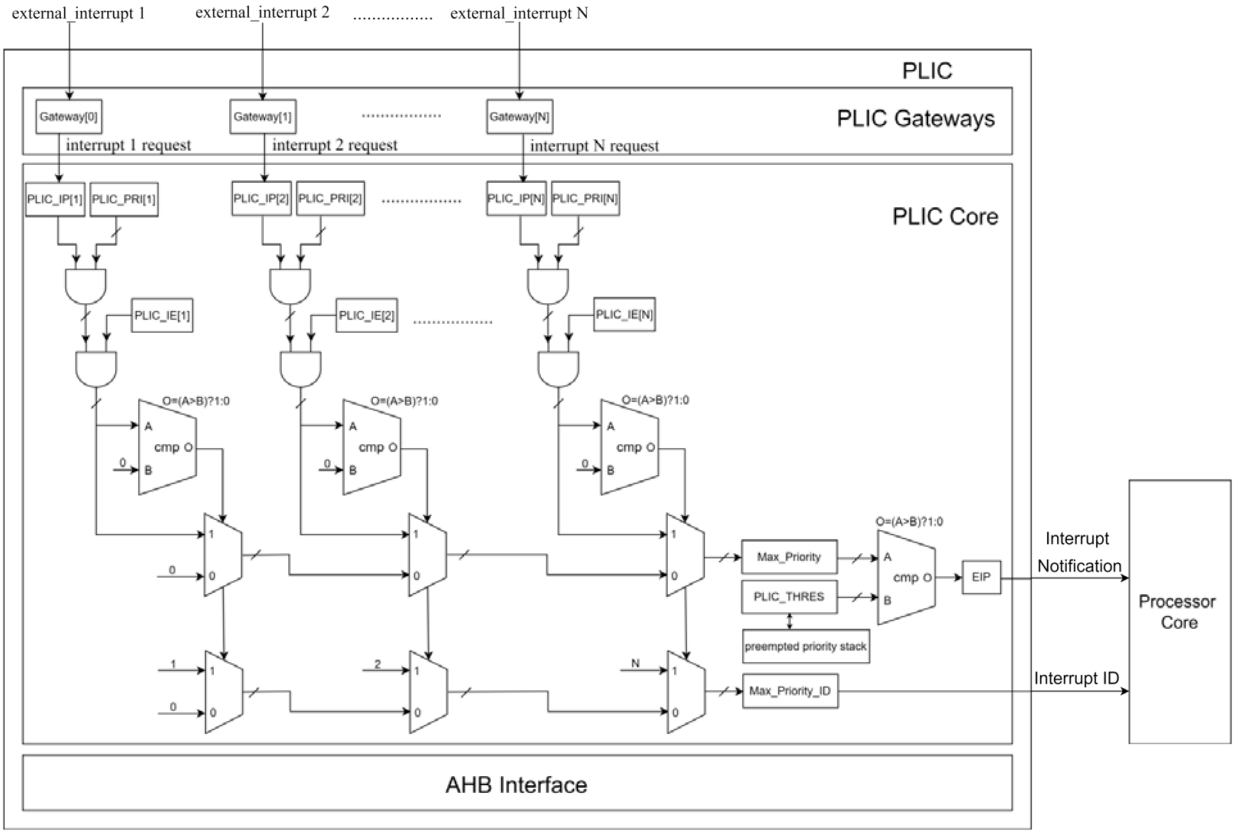
The above figure shows the block diagram of PLIC. External interrupt sources (e.g., peripherals) send interrupt requests to PLIC through external_interrupt[N:1] signals. The signals are level-triggered, and they are converted to interrupt requests by the interrupt gateway. Interrupt requests are prioritized and routed to interrupt targets (e.g., processor core) according to interrupt settings. Interrupt settings include enable bits (PLIC_IE), priorities (PLIC_PRI), and priority thresholds (PLIC_THRES), and these settings are programmable through the bus interface. Note that interrupt targets should not modify PLIC_IE, PLIC_PRI and PLIC_THRES if there are any un-serviced interrupts.

The eip is an external interrupt pending notification signal to the target. It is a level signal summarizing the interrupt pending status of all interrupt sources (PLIC_IP) to the target. When a target takes the external interrupt, it should send an interrupt claim request (bus read request) to retrieve the interrupt ID, upon which the corresponding PLIC_IP bit is cleared and eip is de-asserted. The eip is guaranteed to be de-asserted for at least one cycle even if there are pending interrupt sources still remaining. This is done to ensure that the interrupt detection logic of the target processor can see the remaining interrupt pending status.

The interrupt gateway stops processing newer interrupt requests from its interrupt sources once it reports an interrupt request. When the target has serviced the interrupt, it should send the interrupt completion message (bus write request) to PLIC such that the interrupt gateway resumes processing newer interrupt requests.

The PLIC_IP register provides a summary of all interrupt sources status. In addition, it is also writable for setting software-programmed interrupts for the corresponding interrupt sources.

Figure 9-3 Detailed Block Diagram of PLIC



The above figure shows a more detailed block diagram. The PLIC contains multiple interrupt gateways, one per interrupt source, together with a PLIC core that performs interrupt prioritization and routing External interrupts are sent from their source to an interrupt gateway that processes the interrupt signal from each

source and sends a single interrupt request to the PLIC core, which latches these in the core interrupt pending bits (PLIC_IP). Each interrupt source is assigned a separate priority (PLIC_PRI) and interrupt enable (PLIC_IE). The PLIC core generates an interrupt notification to the processor core if there are any pending interrupts enabled, and the priority of the pending interrupts exceeds the target threshold (PLIC_THRES). When the target takes the external interrupt, it sends an interrupt claim request to retrieve the identifier of the highest-priority global interrupt source pending for that target from the PLIC core, which then clears the corresponding interrupt source pending bit. After the target has serviced the interrupt, it sends the associated interrupt gateway an interrupt completion message and the interrupt gateway can now forward another interrupt request for the same source to the PLIC.

9.3 External Interrupt Sources

There are 52 external interrupt sources, listed in table below.

Table 9-15 Interrupt Sources

No.	Interrupt Source	No.	Interrupt Source
1	stimer_irq: system timer interrupt	27	gpio2risc[1]_irq
2	algm_irq: analog register master interface interrupt	28	soft_irq: software interrupt
3	timer1_irq	29	mspi_irq
4	timer0_irq	30	usb_reset_irq: USB reset interrupt
5	dma_irq	31	usb_250us_sof_irq: USB 250us or SOF interrupt
6	bmc_irq: ahb bus matrix controller interrupt	32	ir_learn
7	usb_setup_irq: USB setup interrupt	33	qdec_irq
8	usb_data_irq: USB data interrupt	34	gpio_src_irq[0]: gpio_group_irq[0]
9	usb_status_irq: USB status interrupt	35	gpio_src_irq[1]: gpio_group_irq[1]
10	usb_setinf_irq: USB set interface interrupt	36	gpio_src_irq[2]: gpio_group_irq[2]
11	usb_edp_irq: USB edp(1-8) interrupt	37	gpio_src_irq[3]: gpio_group_irq[3]
12	reserved	38	gpio_src_irq[4]: gpio_group_irq[4]
13	reserved	39	gpio_src_irq[5]: gpio_group_irq[5]
14	reserved	40	gpio_src_irq[6]: gpio_group_irq[6]
15	zb_ble_tl_irq:BLE(TL) sub-system interrupt	41	gpio_src_irq[7]: gpio_group_irq[7]
16	pwm_irq	42	trng
17	pke_irq	43	hash
18	uart1_irq	44	pm_wkup_irq: PM wakeup interrupt

No.	Interrupt Source	No.	Interrupt Source
19	uart0_irq	45	pm_mix_irq: PM mixed interrupt
20	dfifo_irq: audio dma fifo interrupt	46	dpr_irq
21	i2c_irq	47	ske
22	lspi_irq	48	uart2
23	gspi_irq	49	rsvd
24	usb_pwdn_irq: USB suspend interrupt	50	chacah20
25	gpio_irq	51	saradc_dig
26	gpio2risc[0]_irq	52	rz

9.3.1 Support for Preemptive Priority Interrupt

The PLIC implements the preemptive priority interrupt extension which enables faster responses for high-priority interrupts. This feature is enabled by setting PLIC_FEN.PREEMPT to 1.

With this extension, if a high-priority interrupt arrives and the global interrupt is enabled (i.e., mstatus.MIE is 1), the processor stops servicing the current low-priority interrupt and begin servicing this new high-priority interrupt. The handling of the suspended lower-priority interrupts resume only after the handling of the higher-priority interrupt ends. Interrupts of same or lower priorities do not cause preemption to take effect and interfere the handling of the current interrupt. They have to wait until the handling of the current interrupt finishes.

To support this feature, the PLIC core is enhanced with a preempted priority stack. The stack saves and restores priorities of the nested/preempted interrupts.

The operation of the preempted stack is implicitly performed through two regular PLIC operations (Interrupt Claim and Interrupt Completion). See the next two subsections for more information.

9.3.1.1 Interrupt Claims with Preemptive Priority

When the target sends an interrupt claim message to the PLIC core, the PLIC core atomically determines the ID of the highest-priority pending interrupt for the target and then de-assert the corresponding source's PLIC_IP bit. The PLIC core then returns the ID to the target.

At the same time, the priority number in the target's Priority Threshold Register (PLIC_THRES) is saved to a preempted priority stack for that target and the new priority number of the claimed interrupt is written to PLIC_THRES.

9.3.1.2 Interrupt Completion with Preemptive Priority

When the target sends an interrupt completion message to the PLIC core, in addition to forwarding the completion message to the associated gateway, the PLIC core restores the highest priority number in the preempted priority stack back to PLIC_THRES.

Note that out-of-order completion of interrupts is not allowed when this feature is turned on — the latest claimed interrupt should be completed first.

9.3.1.3 Programming Sequence to Allow Preemption of Interrupts

Turning on the global interrupt enable flag (mstatus.MIE) is all it takes to allow the current interrupt handler to be preempted by higher priority interrupts. However, as the preemptive priority stack operations do not allow out-of-order completion, some care should be taken to make sure that the claim and completion operations are nested properly.

For the non-vectored mode single-entry interrupt handler, the global interrupt enable flag could be turned on after the processor context are saved and Interrupt Claim is performed to allow preemption of the current interrupt handler. At the end of interrupt handler, an Interrupt Completion message is performed to signal that the handler has processed the interrupt and PLIC may deliver the next interrupt from the same interrupt source again. As both claim and completion messages are done through load/store instructions to device regions, they should automatically be ordered correctly. Compared with the vectored mode interrupt handler two paragraphs below, the global interrupt flag does not need to be disabled and no FENCE needs to be inserted after sending the completion message.

In summary, below is the suggested sequence for a non-vector mode interrupt handler for supporting preemptive priority interrupts:

1. Save registers/CSRs to stack
2. Send Interrupt Claim message to PLIC (device-load)
3. Enable global interrupt (mstatus.MIE)
4. Handle the expected interrupt
5. Send Interrupt Completion message to PLIC (device-store)
6. Restore registers/CSRs
7. Return from interrupt

For vector mode interrupt handlers, Interrupt Claim is implicit when the external interrupt is taken. The global interrupt enable flag could be turned on as long as the processor context are saved to allow preemption of the current interrupt handler. However, the global interrupt flag should be turned off before Interrupt Completion operations are performed, since the processor triggers the next implicit Interrupt Claim operation as soon as the global interrupt enable flag is turned on and cause races between Interrupt Claim and Interrupt Completion. Additionally, a FENCE io,io operation should be inserted after the Interrupt Completion operation to make sure that the completion message reaches PLIC before the interrupt handler returns, which turns on the interrupt enable flag again and cause the next Interrupt Claim to be performed.

In summary, below is the suggested sequence for a vector mode interrupt handler for supporting preemptive priority interrupts:

1. Save registers/CSRs to stack
2. Enable global interrupt (mstatus.MIE)
3. Handle the expected interrupt
4. Disable global interrupt (mstatus.MIE)
5. Send Interrupt Completion message to PLIC (device-store)
6. Restore registers/CSRs
7. Use a FENCE io, io instruction to ensure that the completion message has reached PLIC.
8. Return from interrupt

9.3.2 Vectored Interrupts

The PLIC enhances the RISC-V PLIC functionality with the vector mode extension to allow the interrupt target to receive the interrupt source ID without going through the target claim request protocol. This feature can shorten the latency of interrupt handling by enabling the interrupt target to run the corresponding interrupt handler directly upon accepting the external interrupt. It is enabled by setting PLIC_FEN.VECTORED to 1.

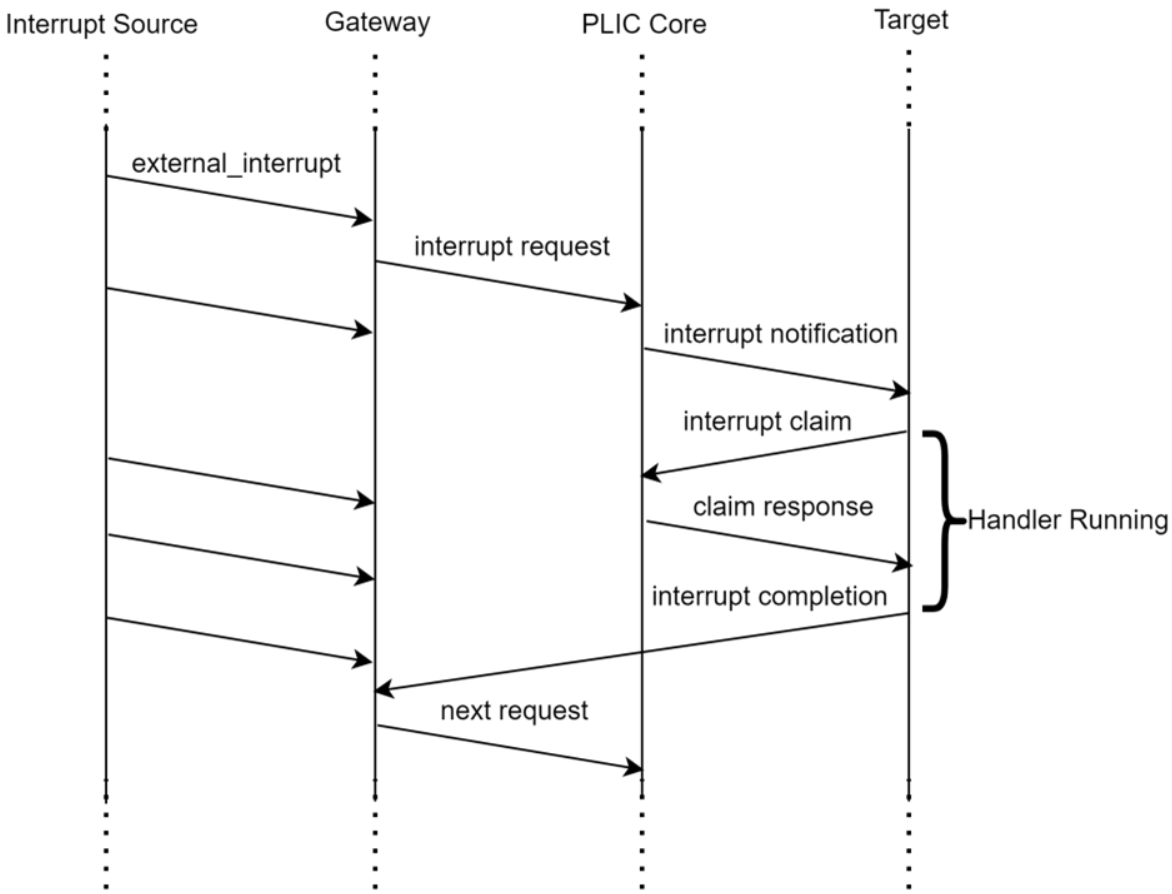
9.3.3 Support for Software-Generated Interrupt

The PLIC also adds support for a software-generated interrupt feature. The interrupt pending registers (PLIC_IP) are writable and has the operation definition of "write-1-to-set", so software can set the pending bit of an interrupt source by writing a 1 to the corresponding bit of the interrupt pending register of the interrupt source.

9.3.4 Interrupt Flow

The below figure shows the messages flowing between agents when handling interrupts via the PLIC.

Figure 9-4 Interrupt Flow



The gateway only forwards a single interrupt request at a time to the PLIC, and not forward subsequent interrupts requests until an interrupt completion is received. The PLIC sets the PLIC_IP bit once it accepts an interrupt request from the gateway, and sometime later forward an interrupt notification to the target. The target might take a while to respond to a new interrupt arriving, but then sends an interrupt claim request to the PLIC core to obtain the interrupt ID. The PLIC core atomically returns the ID and clear the corresponding

PLIC_IP bit. Once the handler has processed the interrupt, it sends an interrupt completion message to the gateway to allow a new interrupt request.

9.4 PLIC Register Description

The PLIC related register are listed in table below. The base address for the following registers is 0xC4000000.

Please note that PLIC supports only 32-bit. Behaviors of 8-bit and 16-bit transfers are undefined, and these transfers might be ignored as well as result in error responses or unexpected register updates.

Table 9-16 Register Configuration for PLIC

Offset	Name	Type	Description	Reset Value
0x00	PLIC_FEN	R/W	Feature Enable Register [0]: PREEMPT, Preemptive priority interrupt enable [1]: VECTORED, Vector mode enable Please note that both this bit and the mmisc_ctl.VEC_PLIC bit of the processor should be turned on for the vectored interrupt support to work correctly.	0x00
0x04*n	PLIC_PRI	R/W	Interrupt Source Priority. This register determines the priority for interrupt source n. [1:0]: Interrupt source priority. 0: Never interrupt, 1-3: Interrupt source priority. The larger the value, the higher the priority.	0x01
0x1000	PLIC_IP	R/W	Interrupt sources 1~31 Pending. The registers provide the interrupt pending status of interrupt sources 1~31, and a way for software to trigger an interrupt without relying on external devices. Every interrupt source occupies 1 bit.When these registers are read, the interrupt pending status of interrupt sources are returned. The pending bits could be set by writing a bit mask that specifies the bit positions to be set, and this action would result in software-programmed interrupts of the corresponding interrupt sources. The pending bits could only be cleared through the Interrupt Claim requests. [31:1]: interrupt pending status of interrupt sources 1~31.	0x00

Offset	Name	Type	Description	Reset Value
0x1004	PLIC_IP_H	R/W	Interrupt sources 32~52 Pending. The registers provide the interrupt pending status of interrupt sources 32~46, and a way for software to trigger an interrupt without relying on external devices. Every interrupt source occupies 1 bit.When these registers are read, the interrupt pending status of interrupt sources are returned. The pending bits could be set by writing a bit mask that specifies the bit positions to be set, and this action would result in software-programmed interrupts of the corresponding interrupt sources. The pending bits could only be cleared through the Interrupt Claim requests. [14:0]: interrupt pending status of interrupt sources 32~52.	0x00
0x2000	PLIC_IE	R/W	Interrupt Enable Bits for interrupt sources 1~31 Every interrupt source occupies 1 bit. [31:1]: Interrupt Enable Bits for interrupt sources 1~31	0x00
0x2004	PLIC_IE_H	R/W	Interrupt Enable Bits for interrupt sources 32~52 Every interrupt source occupies 1 bit. [20:0]: Interrupt Enable Bits for interrupt sources 32~46.	0x00
0x200000	PLIC_THRES	R/W	Priority Threshold [31:0]: THRESHOLD, Interrupt priority threshold	0x0
0x200004	PLIC_CLAIM_COMP	R/W	Claim and Complete Register [9:0]: INTERRUPT_ID, On reads, indicating the interrupt source that has being claimed. On writes, indicating the interrupt source that has been handled (completed).	0x0
0x200400	PLIC_PPSTACK	R/W	Preempted Priority Stack Register The register is read/writable registers for accessing the preempted priority stack. The purpose of the register is for saving and restoring priorities of the nested/preempted interrupts. [3:0]: Each bit indicates if the corresponding priority level has been preempted by a higher-priority interrupt.	0x00

9.5 Software Platform-Level Interrupt Controller (PLIC_SW)

9.5.1 Introduction

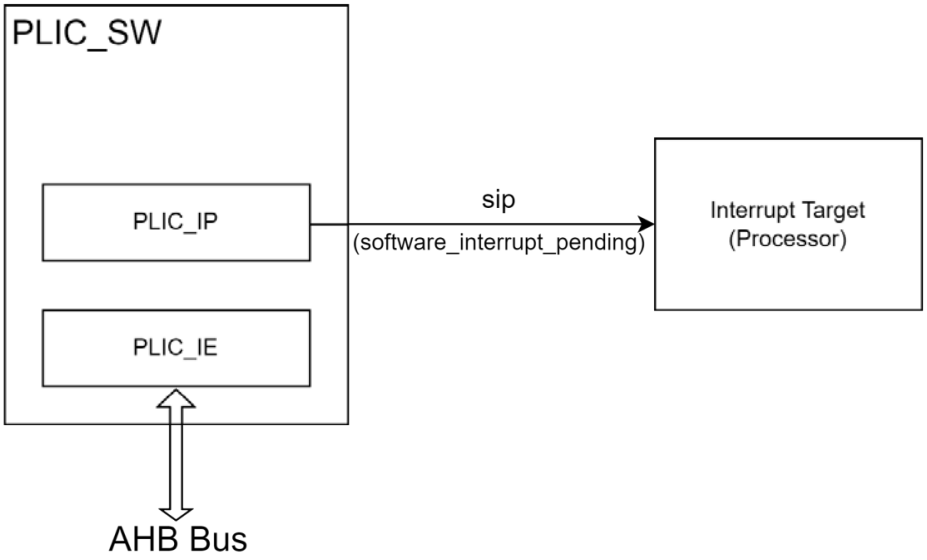
The SoC embeds another one PLIC named PLIC_SW for Software-programmable interrupt generation.It is compatible with RISC-V PLIC with the following features:

- Number of interrupt: 1
- Software-programmable interrupt generation

The below Figure shows the block diagram of PLIC_SW. PLIC_SW doesn't support handling external interrupts, only support Software-programmable interrupt.

For detailed functional descriptions, please refer to PLIC.

Figure 9-5 Block Diagram of PLIC_SW



9.5.2 PLIC_SW Register Description

The PLIC_SW related register are listed in table below. The base address for the following registers is 0xC6400000.

Please note that PLIC_SW supports only 32-bit. Behaviors of 8-bit and 16-bit transfers are UNDEFINED, and these transfers might be ignored as well as result in error responses or unexpected register updates.

Table 9-17 Register Configuration for PLIC_SW

Offset	Name	Type	Description	Reset Value
0x1000	PLIC_IP	R/W	Interrupt sources 1 Pending. The registers provide the interrupt pending status of interrupt sources 1, and a way for software to trigger an interrupt without relying on external devices.Every interrupt source occupies 1 bit.When these registers are read, the interrupt pending status of interrupt sources are returned. The pending bits could be set by writing a bit mask that specifies the bit positions to be set, and this action would result in software-programmed interrupts of the corresponding interrupt sources. The pending bits could only be cleared through the Interrupt Claim requests. [1]: interrupt pending status of interrupt sources 1.	0x00
0x2000	PLIC_IE	R/W	Interrupt Enable Bits for interrupt sources 1. Every interrupt source occupies 1 bit. [1]: Interrupt Enable Bits for interrupt sources 1	0x00
0x200004	PLIC_CLAIM_COMP	R/W	Claim and Complete Register [9:0]: INTERRUPT_ID, On reads, indicating the interrupt source that has being claimed. On writes, indicating the interrupt source that has been handled (completed).	0x00

10 DMA

10.1 Introduction

The SoC embeds DMA (Direct Memory Access) module with DMAC. DMAC is a direct memory access controller which transfers regions of data efficiently on bus.

DMAC features include:

- Supports up to 8 DMA channels
- Supports up to 22 request/acknowledge pairs for hardware handshaking
- Supports chain transfer

10.1.1 Function Description

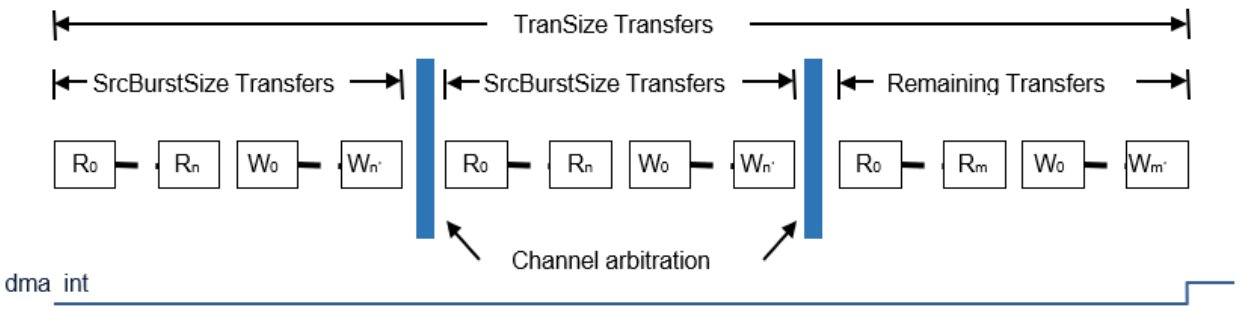
DMAC supports up to 8 DMA channels. Each DMA channel provides a set of registers to describe the intended data transfers. Multiple DMA channels can be enabled concurrently, but the DMA controller services one channel at a time.

Figure 10-1 shows an illustration of data transfer timing for a channel. In this figure, R means Read, W means Write, n is related with BurstSize, for example, when BurstSize is set to 2, it means 4 DMA transfers are required, n is 3 and details refer to the SrcBurstSize register description in Table 10-5. The details of channel arbitration refers to the next section. To prevent channels from being starved, the DMA controller services all ready-channels alternatively, performing at most SrcBurstSize data transfers each time. Consequently, the data transfers of a channel may be split into several chunks when the total transfer size (TranSize) is larger than the source burst size (SrcBurstSize). When the overall data transfers of a channel complete, the DMA controller updates the interrupt status register, IntStatus, and assert the interrupt signal if the terminal count interrupt is enabled.

The peripherals that support DMA burst transfer include: Audio, MSPI, LSPI and GSPI. For specific supported BurstSize and direction, please refer to the relevant sections of the corresponding peripheral interfaces.

The data transfers of a channel is stopped when an error occurs. The data transfers of a channel can also be aborted by software. In either case, the DMA controller disables the channel, and assert the interrupt signal if the corresponding interrupt is enabled.

Figure 10-1 Example of DMA Data Transfers



10.1.1.1 Channel Arbitration

DMA provides two priority levels for channel arbitration. Every channel is associated with a priority level by the Priority field of the channel control register, ChnCtrl. During the channel arbitration, the DMA controller selects a high priority channel first. A low priority channel is only selected if there is no high priority channel. Channels of the same priority level is selected by the round-robin scheme.

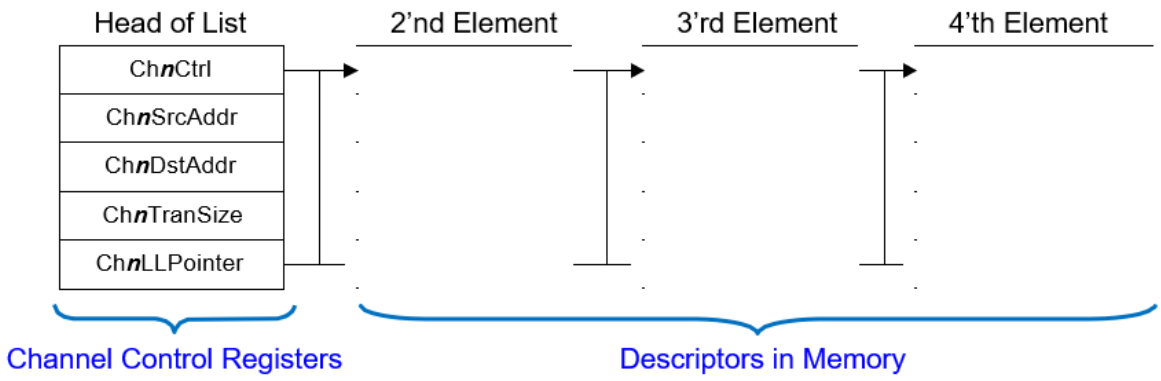
10.1.1.2 Chain Transfer

DMA provides the chain transfer function, with which multiple blocks of data can be transferred consecutively without the intervention of the main processor.

Before a chain transfer is started, a linked list structure must be built to describe the data blocks to move and the associated control setups. The first element of the list (the head of the list) is described by the channel control registers. The rest of elements of the list are specified by the linked list descriptors stored in the memory, where the linked list descriptor holds the control values to load to the channel control registers to continue the data transfer. Figure 2 shows an example of the linked list structure.

When the channel is enabled, the DMA controller first transfers data according to the channel control registers. After the data transfer completes, the DMA controller continues the data transfer by following the Ch*n*LLPointer. The content of the linked list descriptor pointed by Ch*n*LLPointer is loaded to the channel control registers if Ch*n*LLPointer is not zero. The loaded descriptor becomes the new head of the list and this process repeats until the Ch*n*LLPointer is zero.

Figure 10-2 Linked List Structure for Chain Transfers



There are three modes of interrupt generation for the linked list, see the description of register Chn_llp_int_mode below for details.

(1) When Chn_llp_int_mode==0, the linked list interrupt is generated under the following conditions

When the terminal count interrupt (IntTCMask) of a channel is enabled, the DMA controller generates an interrupt and disable the channel when the data transfer for the head of the list is done. If the Ch*n*LLPointer is not zero, the channel control registers is preloaded with the next descriptor before the interrupt is generated. The interrupt handling software could resume the chain transfer by just re-enabling the channel.

(2) When Chn_llp_int_mode==1, the linked list interrupt is generated under the following conditions

When the terminal count interrupt (IntTCMask) of a channel is enabled, the DMA controller generates an interrupt and disable the channel when the data transfer for the head of the list is done. If the Ch*n*LLPointer is not zero, the interrupt is generated after each channel control register is completed.

(3) When Chn_llp_int_mode==2, the linked list interrupt is generated under the following conditions

When the terminal count interrupt (IntTCMask) of a channel is enabled, the DMA controller generates an interrupt and disable the channel when the data transfer for the head of the list is done. If the Ch*n*LLPointer is not zero, the interrupt is generated after each channel control register is completed, and the linked list is aborted, the software should resume the chain transfer by re-enabling the channel.

The following table shows the format of the linked list descriptor. The bit field definition of each descriptor word is the same as the corresponding channel control register except the channel enable bit, which is reserved in the linked list descriptor.

Table 10-1 Format of Linked List Descriptor

Name	Offset	Description	Format
Ctrl	0x00	Channel control	See Table 10-5
SrcAddr	0x04	Source address	See Table 10-7
DstAddr	0x08	Destination address	See Table 10-8
TranSize	0x0C	Total transfer size	See Table 10-9
LLPointer	0x10	Linked list pointer	See Table 10-10

The peripherals supporting chain transfer include: RX of UART, and audio.

10.1.1.3 Data Order

DMA provides three address control modes: increment mode, decrement mode, and fixed mode. At the increment mode, the address is increased after the DMA controller accesses a data of the source/destination. At the decrement mode, the address is decreased after the DMA controller accesses a data of the source/ destination. At the fixed mode, the address remains unchanged after the DMA controller accesses a data of the source/destination.

10.2 Registers

10.2.1 Register Summary

The table below shows a summary of the DMA registers. The base address of DMA is 0x80100400

Table 10-2 DMA Related Registers

Offset	Name	Description	Category
+0x30	IntStatus	Interrupt status register	Channel status register
+0x38~0x3c	-	Reserved	

Offset	Name	Description	Category
+0x40	ChAbort	Channel abort register	Channel control registers
+0x44 + n*0x14	Ch*n*Ctrl	Channel *n* control register	
+0x48 + n*0x14	Ch*n*SrcAddr	Channel *n* source address register	
+0x4c + n*0x14	Ch*n*DstAddr	Channel *n* destination address register	
+0x50 + n*0x14	Ch*n*TranSize	Channel *n* transfer size register	
+0x54 + n*0x14	Ch*n*LLPointer	Channel *n* linked list pointer register	

10.2.2 Interrupt Status Register (Offset 0x30)

This register contains the terminal count, error, and abort status. The terminal count status of a channel is asserted when the channel encounters the terminal counter event. The error/abort status of a channel is asserted when the channel encounters the error/abort event. There is one bit of status for each channel and the status bit is zero when the corresponding channel is not configured.

Table 10-3 Interrupt Status Register

Name	Bit	Type	Description	Reset
Reserved	31:24	-	Reserved	-
TC	23:16	R/W1C	The terminal count status of DMA channels, one bit per channel. The terminal count status is asserted when a channel transfer finishes without abort or error event. 0x0: channel N has no terminal count status 0x1: channel N has terminal count status	0x0
Abort	15:8	R/W1C	The abort status of channel, one bit per channel. The abort status is asserted when a channel transfer is aborted. Configure the channel abort register (offset 0x40) corresponding bit to 1 to indicate abort the corresponding DMA channel. 0x0: channel N has no abort status 0x1: channel N has abort status	0x0

Name	Bit	Type	Description	Reset
Error	7:0	R/W1C	The error status, one bit per channel. The error status is asserted when a channel transfer encounters the following error events: • Bus error • Unaligned address • Unaligned transfer width • Reserved configuration 0x0: channel N has no error status 0x1: channel N has error status	0x0

10.2.3 Channel Abort Register (Offset 0x40)

The register controls the abortion of the DMA channel transfers, one-bit per channel. Write 1 to stop the current transfer of the corresponding channel. The abort bit is automatically cleared by hardware when the corresponding status bit in the interrupt status register is cleared.

Table 10-4 Channel Abort Register

Name	Bit	Type	Description	Reset
ChAbort	7:0	WO	Write 1 to this field to stop the channel transfer. The bits can only be set when the corresponding channels are enabled. Otherwise, the writes is ignored for channels that are not enabled.	0x0

10.2.4 Channel n Control Register (Offset 0x44+n*0x14)

Table 10-5 Channel n Control Register

Name	Bit	Type	Description	Reset
Auto enable en	31	R/W	BB TX RX AUTO EN	0X0
Write_num_en	30	R/W	Enable write num If this register is enabled, the peripheral to SRAM writes the number of bytes received to the first 4 bytes of the destination address at the end of the process. It should be noted that when write_num_en is enabled, CHnTranSize should be set to 0xffffff.	0x0
Priority	29	R/W	Channel priority level 0x0: lower priority 0x1: reserved	0x0

Name	Bit	Type	Description	Reset
Read_num_en	28	R/W	1:tx_size from ram 0:tx_size from reg If the register is enabled, when TX enables the first data transfer, it writes the first word of the source address to the CHnTranSize register and clear rnum_en. If rnum_en is not enabled, it is needed to configure the CHnTranSize register.	0x0
SrcBurstSize	26:24	R/W	Source burst size. This field indicates the number of transfers before DMA channel re-arbitration. This is configured according to the DMA BusrtSize that can be supported by the peripheral. Total byte of a burst is SrcBurstSize * SrcWidth. 0x0: 1 transfer 0x1: 2 transfers 0x2: 4 transfers 0x3: 8 transfers 0x4: 16 transfers 0x5: 32 transfers 0x6: 64 transfers 0x7: 128 transfers	0x0
SrcWidth	23:22	R/W	Source transfer width 0x0: byte transfer 0x1: half-word transfer 0x2: word transfer 0x3: reserved, setting the field with this value triggers error exception	0x2

Name	Bit	Type	Description	Reset
DstWidth	21:20	R/W	Destination transfer width. Both the total transfer byte and the total burst bytes should be aligned to the destination transfer width; otherwise the error event is triggered. For example, destination transfer width should be set as byte transfer if total transfer byte is not aligned to word or half-word. See SrcBurstSize field above for the definition of total burst byte for the definition of the total transfer bytes. 0x0: byte transfer 0x1: half-word transfer 0x2: word transfer 0x3: reserved, set the field as this value triggers error exception	0x2
SrcMode	19	R/W	Source DMA handshake mode 0x0: normal mode 0x1: handshake mode	0x0
DstMode	18	R/W	Destination DMA handshake mode 0x0: normal mode 0x1: handshake mode	0x0
SrcAddrCtrl	17:16	R/W	Source address control 0x0: increment address 0x1: decrement address 0x2: fixed address 0x3: reserved, setting the field with this value triggers the error exception	0x0
DstAddrCtrl	15:14	R/W	Destination address control 0x0: increment address 0x1: decrement address 0x2: fixed address 0x3: reserved, setting the field with this value triggers the error exception	0x0
SrcReqSel	13:9	R/W	Source DMA request select. Select the request/ack handshake pair that the source. See Table 10-6.	0x0

Name	Bit	Type	Description	Reset
DstReqSel	8:4	R/W	Destination DMA request select. Select the request/ack handshake pair that the destination. See Table 10-6.	0x0
IntAbtMask	3	R/W	Channel abort interrupt mask 0x0: allow the abort interrupt to be triggered 0x1: disable the abort interrupt	0x0
IntErrMask	2	R/W	Channel error interrupt mask 0x0: allow the error interrupt to be triggered 0x1: disable the error interrupt	0x0
IntTCMask	1	R/W	Channel terminal count interrupt mask. 0x0: allow the terminal count interrupt to be triggered 0x1: disable the terminal count interrupt	0x0
Enable	0	R/W	Channel enable bit 0x0: disable 0x1: enable	0x0

The following table shows the labels of the request/ack handshake pair for hardware connections. The SrcReqSel and DstReqSel registers select appropriate number from this table according to the actual functional needs.

Table 10-6 Request/Ack Handshake Pair for Hardware Connection

Signal Name	Request/Ack Selection
lspi_tx	0
lspi_rx	1
uart0_tx	2
uart0_rx	3
gspi_tx	4
gspi_rx	5
i2c_tx	6
i2c_rx	7
zb_tx	8
zb_rx	9
pwm_tx	10

Signal Name	Request/Ack Selection
RSVD	11
algm_tx	12
algm_rx	13
uart1_tx	14
uart1_rx	15
audio0_tx	16
audio0_rx	17
audio1_tx	18
audio1_rx	19
mspi_tx	20
mspi_rx	21

10.2.5 Channel n Source Address Register (Offset 0x48+n*0x14)

Table 10-7 Channel n Source Address Register

Name	Bit	Type	Description	Reset
SrcAddr	31:0	R/W	Source starting address. When a transfer completes, its value is updated to the ending address + sizeof(SrcWidth). This address must be aligned to the source transfer size; otherwise, an error event is triggered.	0x0

10.2.6 Channel n Destination Address Register (Offset 0x4C+n*0x14)

Table 10-8 Channel n Destination Address Register

Name	Bit	Type	Description	Reset
DstAddr	31:0	R/W	Destination starting address. When a transfer completes, its value is updated to the ending address + sizeof(DstWidth). This address must be aligned to the destination transfer size; otherwise the error event is triggered.	0x0

Since the data width of the peripherals for DMA transfer are word, it is unified that the DMA SrcAddr and DstAddr are word-aligned.

10.2.7 Channel n Transfer Size Register (Offset 0x50+n*0x14)

Table 10-9 Channel n Transfer Size Register

Name	Bit	Type	Description	Reset
Reserved	31:24	-	-	-
TranSize_idx	23:22	R/W	Byte size	0x0
TranSize	21:0	R/W	Total transfer size from source. The total number of transferred bytes is TranSize * SrcWidth. The value is updated to zero when the DMA transfer is done. If a channel is enabled with zero total transfer size, the error event is triggered and the transfer is terminated.	0x0

The actual amount of data transferred is as follows.

For RX, transize_idx is invalid, the actual amount of data transferred = TranSize * SrcWidth.

For TX, transize_idx is valid, when transize_idx is not 0, the actual amount of data transferred = (TranSize - 1) * SrcWidth + TranSize_idx; when transize_idx is 0, the actual amount of data transferred = TranSize * SrcWidth.

When the DMA transfer direction is from peripheral to SRAM, the actual size written to SRAM is TranSize and TranSize_idx is ignored.

10.2.8 Channel n Linked List Pointer Register (Offset 0x54+n*0x14)

Table 10-10 Channel Linked List Pointer Register

Name	Bit	Type	Description	Reset
LLPointer	31:2	R/W	Pointer to the next block descriptor. The pointer must be word aligned.	0x0
Reserved	1:0	-	-	-

10.2.9 Baseband Related Register

Baseband TX can only use channel 0 of DMA, and baseband RX can only use channel 1 of DMA.

For DMA, there are specific functions for the baseband module, the detailed registers are as follows.

Table 10-11 Baseband Related Registers

Name	Address	Bit	Type	Description	Reset
BB_TX_SIZE	0xf0~0xf1	15:0	R/W	Size of each TX buffer, unit is byte.	0x0
BB_TX_CHN_DEP	0xf3	2:0	R/W	Depth of TX FIFO, the actual depth is 2^BB_TX_CHN_DEP	0x0
BB_RX_WPTR	0xf4	4:0	R/W	RX_WPTR pointer	0x0

Name	Address	Bit	Type	Description	Reset
RX_RPTR_CLR	0xf5	7	W1C	Clear the RX_RPTR pointer	0x0
RX_RPTR_NXT		6	W1C	Add 1 to RX_RPTR pointer	0x0
RX_RPTR_SET		5	W1C	Set RX_RPTR pointer	0x0
BB_RX_RPTR		4:0	R/W	Set the specific value of the RX_RPTR pointer	0x0
BB_RX_SIZE	0xf6~0xf7	15:0	R/W	Size of each RX buffer, unit is byte.	0x0
TX_WPTRn	0x100+n*2, n=[0:5]	15:0	R/W	TX_WPTR pointer corresponding to the baseband channel	0x0
TX_RPTRn_CLR	0x101+n*2(n=[0:5])	7	W1C	Clear the TX_WPTR pointer corresponding to the baseband channel	0x0
TX_RPTRn_NXT		6	W1C	Add 1 to the TX_WPTR pointer corresponding to the baseband channel	0x0
TX_RPTRn_SET		5	W1C	Set the TX_WPTR pointer corresponding to the baseband channel	0x0
TX_RPTRn		4:0	R/W	TX_RPTR pointer corresponding to the baseband channel	0x0

Name	Address	Bit	Type	Description	Reset
Dma_req_d1_en	0x10c	5	R/W	Synchronize the dma_request signal to hclk domain	0x0
Ch1_rx_err_en		4	R/W	DMA TC interrupt does not work if baseband rx_err occurs	0x0
Ch_1_rnum_en_bk		3	R/W	ch_1_rnum_en_bk needs to be used with read_num_en because the read_num_en register is cleared to 0 after each first load of TranSize and the value of ch_1_rnum_en_bk is automatically loaded after the DMA transfer is completed	0x0
Ch_0_rnum_en_bk		2	R/W	ch_0_rnum_en_bk needs to be used with read_num_en because the read_num_en register is cleared to 0 after each first load of TranSize and the value of ch_0_rnum_en_bk is automatically loaded after the DMA transfer is completed	0x1
rx_multi_en		1	R/W	The role of rx_multi_en: it automatically loads the destination address register after each DMA transfer and automatically load 0xffffff into the TranSize register	0x0
Tx_multi_en		0	R/W	The role of tx_multi_en: DMA checks the read/write pointer of the current baseband channel after receiving the send request from the baseband, if the FIFO of the current baseband channel is empty, DMA reads data from the default buffer.	0x0
Rx_wptr_mask	0x10d	4:0	R/W	Depth of RX FIFO, the actual depth is 2^rx_wptr_mask	0x0

10.2.10 Miscellaneous Register

Table 10-12 Linked List Interrupt Mode

Name	Address	Bit	Type	Description	Reset
Chn_llp_int_mode	0x113~0x114	[1+n*2:0+n*2]	R/W	0: llp continue mode, the linked list transfer is continuous, interrupt is generated only when the last chain completes 1: llp interrupt mode, the linked list does not stop, the interrupt is generated at the completion of each chain 2: llp terminal mode, the linked list stops automatically at the completion of each chain, interrupt is generated at the completion of each chain 3: rsvd	0xff

10.3 Usage Guide

10.3.1 From SRAM to SRAM

It is recommended to disable wnum_en, rnum_en and auto_enable_en. For SRAM, the transize_idx (only applicable to peripherals) is invalid. If byte-unit data needs to be transferred from SRAM to SRAM, the srcwidth and dstwidth control registers need to be set.

Assuming that 1023 bytes of data need to be transferred from address A to address B, B should be written to the destination address register, A should be written to the source address register, 1023 should be written to the transize register, and finally the Channel n Control Register (Offset 0x44+n*0x14) should be configured as the table below.

Table 10-13 Register Configuration for SRAM to SRAM

Bit	Name	Configuration
31	auto_enable_en	Set to 0.
30	wnum_en	Set to 0.
29	priority	Set to 0.
28	rnum_en	Set to 0.
27	reserved	Reserved bit
24-26	src_burst_size	Set to any value stated in Table 10-5.
22-23	srcwidth	Set to any value stated in Table 10-5.

Bit	Name	Configuration
20-21	dstwidth	The dstwidth must be aligned with the DMA TranSize. For example, if the DMA TranSize is not aligned with a half word, it should be configured as a byte. If the DMA TranSize is neither aligned with a byte nor aligned with a word, it should be configured as a half word.
19	src_mode	Set to 0.
18	dst_mode	Set to 0.
16-17	src_addr_ctl	Set to 0.
14-15	dst_addr_ctl	Set to 0.
9-13	src_req_sel	Ignore.
4-8	dst_req_sel	Ignore.
3	abort interrupt enable	Enable Abort Interrupt when write 1 to the corresponding channel ChAbort Register.
2	error interrupt enable	Enable Error Interrupt when an error occurs. The detailed error description refers to Error register (Offset 0x30).
1	tc interrupt enable	Enable TC interrupt when DMA transmission completes.
0	enable	Write 1 to start DMA transmission, write 0 to abort transmission.

10.3.2 From SRAM to Peripherals

From SRAM to peripherals, only transferring with srcwidth and dstwidth of word is supported.

Assuming that 1023 bytes of data need to be transferred from SRAM address A to peripheral address B, B should be written to the destination address register, A should be written to the source address register, (1023+3)/4 should be written to the transize register, and 1023%4 should be written to the transize_idx register. Finally, the Channel n Control Register (Offset 0x44+n*0x14) should be configured. as the table below.

Table 10-14 Register Configuration for SRAM to Peripherals

Bit	Name	Configuration
31	auto_enable_en	Set to 0.
30	wnum_en	Set to 0.
29	priority	Set to 0.
28	rnum_en	Set to 0.
27	reserved	Reserved bit

Bit	Name	Configuration
24-26	src_burst_size	Set to the burst size that peripheral supports.
22-23	srcwidth	Set to word.
20-21	dstwidth	Set to word.
19	src_mode	Set to 0.
18	dst_mode	Set to 1.
16-17	src_addr_ctl	Set to 0.
14-15	dst_addr_ctl	Set to 2.
9-13	src_req_sel	Ignore.
4-8	dst_req_sel	Set according to the corresponding Request/Ack Selection number of the peripheral in Table 10-6.
3	abort interrupt enable	Set to the burst size that peripheral supports.
2	error interrupt enable	Enable Error Interrupt when an error occurs. The detailed error description refers to Error register (Offset 0x30).
1	tc interrupt enable	Enable TC Interrupt when DMA transmission completes.
0	enable	Write 1 to start DMA transmission, write 0 to abort transmission.

10.3.3 From Peripherals to SRAM

From peripherals to SRAM, only transferring with srcwidth and dstwidth of word is supported.

If transferring data from peripheral address A to SRAM address B, B should be written to the destination address register, A should be written to the source address register, 0xffffffff should be written to the transize register since the amount of bytes being transferred is unknown. Finally, the Channel n Control Register (Offset 0x44+n*0x14) should be configured as the table below.

Table 10-15 Register Configuration for Peripherals to SRAM

Bit	Name	Configuration
31	auto_enable_en	Set to 0.
30	wnum_en	Set to 1 to write the number of data received to a word before the destination address.
29	priority	Set to 0.
28	rnum_en	Set to 0.

Bit	Name	Configuration
27	reserved	reserved bit
24-26	src_burst_size	Set to the burst size that peripheral supports.
22-23	srcwidth	Set to word.
20-21	dstwidth	Set to word.
19	src_mode	Set to 1.
18	dst_mode	Set to 0.
16-17	src_addr_ctl	Set to 2.
14-15	dst_addr_ctl	Set to 0.
9-13	src_req_sel	Set according to the corresponding Request/Ack Selection number of the peripheral in Table 10-6.
4-8	dst_req_sel	Ignore.
3	abort interrupt enable	Enable Abort Interrupt when write 1 to the corresponding channel ChAbort Register.
2	error interrupt enable	Enable Error Interrupt when an error occurs. The detailed error description refers to Error register (Offset 0x30).
1	tc interrupt enable	Enable TC Interrupt when DMA transmission completes.
0	enable	Write 1 to start DMA transmission, write 0 to abort transmission.

10.3.4 From SRAM to Baseband

The configuration method for transferring data from SRAM to baseband is the same as that from SRAM to peripheral mentioned above. However, for the previous method, after each DMA transfer, it's necessary to reconfigure the transize register, source address register, and enable control register (en). In comparison, there are some enhanced functionalities for baseband. After enabling tx_multi_en, DMA automatically loads the source address register. After enabling rnum_en, DMA automatically loads the transize register (also needing to set ch_0_rnum_en_bk at dma_base+0x10c to 1 since rnum_en is cleared to 0 after every first loading of transize, and it automatically loads the value of ch_0_rnum_en_bk after completing a DMA transfer). By enabling auto_enable_en, the en in the enable control register is automatically enabled depending on the requests made by baseband. The mechanism for automatic loading of the source address register works as follows:

TX has a total of 6 channels (0-5), it is needed to set the FIFO depth of each chn tx_chn_dep (where 0 represents one buffer, 1 represents two buffers, and 2 represents four buffers) and set the size of each buffer buf_size(bb_tx_size) bytes.

The channel is fixed on the baseband side for each transfer. After enabling the tx_multi_en of DMA, DMA receives the send request from baseband and views the read/write pointer of the current channel. If the FIFO

of the current channel is empty, DMA reads data from the default buff. If the current channel's FIFO is not empty, DMA goes to the corresponding rptr of the current channel to read the data. The write pointer of TX is maintained by software while the read pointer is maintained by hardware (incremented every time a tx_commit is received by baseband in multi-mode. In case the baseband does not use multi-mode, the software can neglect maintaining the write pointer, making all txfifos empty; then, the DMA automatically reads data from the default buffer during transmission). Hence, the multi-mode of DMA can be used even when the baseband is in non-multi mode.

10.3.5 From Baseband to SRAM

The same method used for peripheral to SRAM can also be applied to transfer data from baseband to SRAM. Similar to the previous case, after every DMA transfer, configuring destination address register, transize register, and enable control register's en becomes necessary. However, for the Rx channel of baseband, DMA has enhanced functionality. By enabling rx_multi_en, upon completing every DMA transfer, the DMA automatically loads the destination address register and sets the transize register with 0xffffff. The auto_enable_en in the control register is activated such that when the baseband initiates an RX request, the en register in the control register is automatically enabled.

The mechanism for DMA to automatically load the destination registers:

Unlike TX which has 6 channels, RX has only one FIFO buffer, therefore only the RX FIFO depth rx_wptr_mask(dma_base+0x10d[4:0]) and the buffer size (bb_rx_size) in bytes need to be set.

Whenever a DMA transfer is initiated, DMA reads data from baseband and writes it to the specified address in the destination address register. After completing the transfer, if the received packet is valid, the baseband generates an rx_commit signal to the DMA, which increments the rx_wptr. The next packet of data is then written into the buffer pointed to by rx_wptr. If the rxfifo's depth is zero, incoming data continues to be written to the same location.

11 Interface

11.1 GPIO

The SoC supports up to 48 GPIOs (differs for specific part number, refers to 1.4 Ordering Information). All digital IOs can be used as general purpose IOs.

11.1.1 GPIO Main Features

The GPIO main features include:

- Up to 8 GPIO pins per GPIO port
- Configurable output drive strength
- Output data from output data register or peripheral
- Input data to input data register or peripheral
- Internal pull-up and pull-down resistors
- Trigger interrupt on state changes on any pin except for flash IO
- Wake-up from high or low level triggers on all pins except for flash IO

11.1.2 Basic Configuration of GPIO

All GPIOs can be configured with related registers, as described as following.

Table 11-1 GPIO Pad Function Mux

Pad	Default	Register = [1:98]	Register = 0	Register[a]
PA[0]	GPIO	All functions[b]	-	0x80140c70[6:0]
PA[1]	GPIO	All functions	-	0x80140c71[6:0]
PA[2]	GPIO	All functions	-	0x80140c72[6:0]
PA[3]	GPIO	All functions	-	0x80140c73[6:0]
PA[4]	GPIO	All functions	-	0x80140c74[6:0]
PA[5]	GPIO	GPIO	DM	-
PA[6]	GPIO	GPIO	DP	-
PA[7]	SWS	GPIO	SWS	-
PB[0]	GPIO	All functions	-	0x80140c78[6:0]
PB[1]	GPIO	All functions	-	0x80140c79[6:0]
PB[2]	GPIO	All functions	-	0x80140c7a[6:0]
PB[3]	GPIO	All functions	-	0x80140c7b[6:0]
PB[4]	GPIO	All functions	-	0x80140c7c[6:0]

Pad	Default	Register = [1:98]	Register = 0	Register[a]
PB[5]	GPIO	All functions	-	0x80140c7d[6:0]
PB[6]	GPIO	All functions	-	0x80140c7e[6:0]
PB[7]	GPIO	All functions	-	0x80140c7f[6:0]
PC[0]	SSPI_CN	GPIO, All functions	SSPI_CN	0x80140c80[6:0]
PC[1]	SSPI_CK	GPIO, All functions	SSPI_CK	0x80140c81[6:0]
PC[2]	SSPI_SI	GPIO, All functions	SSPI_SI	0x80140c82[6:0]
PC[3]	SSPI_SO	GPIO, All functions	SSPI_SO	0x80140c83[6:0]
PC[4]	TDI	GPIO, All functions	TDI	0x80140c84[6:0]
PC[5]	TDO	GPIO, All functions	TDO	0x80140c85[6:0]
PC[6]	TMS	GPIO, All functions	TMS	0x80140c86[6:0]
PC[7]	TCK	GPIO, All functions	TCK	0x80140c87[6:0]
PD[0]	GPIO	All functions	-	0x80140c88[6:0]
PD[1]	GPIO	All functions	-	0x80140c89[6:0]
PD[2]	GPIO	All functions	-	0x80140c8a[6:0]
PD[3]	GPIO	All functions	-	0x80140c8b[6:0]
PD[4][c]	GPIO	All functions	-	0x80140c8c[6:0]
PD[5][d]	GPIO	All functions	-	0x80140c8d[6:0]
PD[6][e]	GPIO	All functions	-	0x80140c8e[6:0]
PD[7]	GPIO	All functions	-	0x80140c8f[6:0]
PE[0]	GPIO	All functions	-	0x80140c90[6:0]
PE[1][f]	GPIO	-	LSPI_CK	-
PE[2]	GPIO	-	LSPI_MOSI	-
PE[3]	GPIO	-	LSPI_MISO	-
PE[4]	GPIO	-	LSPI_IO2	-
PE[5]	GPIO	-	LSPI_IO3	-
PE[6]	GPIO	All functions	-	0x80140c96[6:0]
PE[7]	GPIO	All functions	-	0x80140c97[6:0]
PF[0]	GPIO	All functions	-	0x80140c98[6:0]

Pad	Default	Register = [1:98]	Register = 0	Register[a]
PF[1]	GPIO	All functions	-	0x80140c99[6:0]
PF[2]	GPIO	All functions	-	0x80140c9a[6:0]
PF[3]	GPIO	All functions	-	0x80140c9b[6:0]
PF[4]	GPIO	All functions	-	0x80140c9c[6:0]
PF[5]	GPIO	All functions	-	0x80140c9d[6:0]
PF[6]	GPIO	All functions	-	0x80140c9e[6:0]
PF[7]	GPIO	All functions	-	0x80140c9f[6:0]

a. The bit width of the register is [6:0], and the default value is 0x00.

b. "All functions" include 89 functions, see Table 11-2 below.

c. PD[4] is for internal use and not recommended for customer to use. Contact Telink FAE for details.

d. (1) PD[5], PD[6], PD[7] of TL7218A and TL7215A are not recommended used for Channel Sounding application.
(2) PD[5] is recommended to be used as output only, the details refer to the hardware design guideline.

e. PD[6] and PD[7] are not recommended to be used as PWM output.

f. The RF sensitivity is affected when PE[1] to PE[5] are configured as LSPI working above 15MHz, the details refer to the hardware design guideline.

The functions included in the "All functions" are listed in the table below:

Table 11-2 GPIO functions

Register value	Function	Register value	Function	Register value	Function
1	PWM0	31	I2S0_LR0	65	ATSEL_5
2	PWM1	32	I2S0_DAT0	66	IR_LEARN
3	PWM2	33	I2S0_LR1	67	UART2_CTS
4	PWM3	34	I2S0_DAT1	68	UART2_RTS
5	PWM4	35	I2S0_CLK	69	UART2_TX
6	PWM5	36	I2S1_BCK	70	UART2_RTX
7	PWM0_N	37	I2S1_LR0	71	SDM0_P
8	PWM1_N	38	I2S1_DAT0	72	SDM0_N
9	PWM2_N	39	I2S1_LR1	73	SDM1_P
10	PWM3_N	40	I2S1_DAT1	74	SDM1_N
11	PWM4_N	41	I2S1_CLK	75	I2S2_BCK
12	PWM5_N	42	DMIC0_CLK	76	I2S2_LR0

Register value	Function	Register value	Function	Register value	Function
13	GSPI_CN0	43	DMIC0_DAT	77	I2S2_DAT0
14	GSPI_CK	44	MSPI_CN2	78	I2S2_LR1
15	GSPI_IO3	45	MSPI_CN3	79	I2S2_DAT1
16	GSPI_IO2	47	TX_CYC2PA	80	I2S2_CLK
17	GSPI_MISO	48	WIFI_DENY	81	SSPI_CN
18	GSPI_MOSI	49	BT_ACTIVITY	82	SSPI_CK
19	I2C_SCL	50	BT_STATUS	83	SSPI_SI
20	I2C_SDA	52	ATSEL_0	84	SSPI_SO
21	UART0_CTS	53	ATSEL_1	85	Reserved
22	UART0_RTS	54	ATSEL_2	86	PWM_SYNC
23	UART0_TX	55	ATSEL_3	87	PWM6
24	UART0_RTX	56	RX_CYC2LNA	88	PWM6_N
25	UART1_CTS	59	I2C1_SDA	89	TMR0_CMP
26	UART1_RTS	60	I2C1_SCL	90	TMR1_CMP
27	UART1_TX	61	GSPI_CN1	92	LSPI_CN
28	UART1_RTX	62	GSPI_CN2	96	MSPI_CN1
29	CLK_7816	63	GSPI_CN3	97	RZ_TX
30	I2S0_BCK	64	ATSEL_4	98	SWM

NOTE:

For GPIO Multiple Function Switching,

1. *If the default function is GPIO, users need to configure the desired function MUX first and then disable the GPIO function.*
2. *If the default is the function IO, which needs to be changed to GPIO output. Users need to set the output set, output clear, output toggle and OEN of the corresponding IO first, and then enable GPIO function.*
3. *If the default is the function IO, which needs to be changed to GPIO input,*
   - The IO needs to be pulled up:
     - Case 1 (digital pull-up): set the pullup to 1, OEN to 1;
     - Case 2 (analog pull-up): set the analog registers for Pull-up.
   - The IO that does not need to be pulled up:
     - Case 1 (digital pull-up): set the pullup to 0, OEN to 1;
     - Case 2 (analog pull-up): set the analog registers for Pull-up.
   - Finally enable GPIO function.

Table 11-3 GPIO Setting 1

Pad	Input	IE	OEN	Polarity	DS	Act as GPIO
PA[0]	0x80140c00[0]	0x80140c01[0]	0x80140c02[0]	0x80140c04[0]	0x80140c05[0]	0x80140c06[0]
PA[1]	0x80140c00[1]	0x80140c01[1]	0x80140c02[1]	0x80140c04[1]	0x80140c05[1]	0x80140c06[1]
PA[2]	0x80140c00[2]	0x80140c01[2]	0x80140c02[2]	0x80140c04[2]	0x80140c05[2]	0x80140c06[2]
PA[3]	0x80140c00[3]	0x80140c01[3]	0x80140c02[3]	0x80140c04[3]	0x80140c05[3]	0x80140c06[3]
PA[4]	0x80140c00[4]	0x80140c01[4]	0x80140c02[4]	0x80140c04[4]	0x80140c05[4]	0x80140c06[4]
PA[5]	0x80140c00[5]	0x80140c01[5]	0x80140c02[5]	0x80140c04[5]	0x80140c05[5]	0x80140c06[5]
PA[6]	0x80140c00[6]	0x80140c01[6]	0x80140c02[6]	0x80140c04[6]	0x80140c05[6]	0x80140c06[6]
PA[7]	0x80140c00[7]	0x80140c01[7]	0x80140c02[7]	0x80140c04[7]	0x80140c05[7]	0x80140c06[7]
PB[0]	0x80140c10[0]	0x80140c11[0]	0x80140c12[0]	0x80140c14[0]	0x80140c15[0]	0x80140c16[0]
PB[1]	0x80140c10[1]	0x80140c11[1]	0x80140c12[1]	0x80140c14[1]	0x80140c15[1]	0x80140c16[1]
PB[2]	0x80140c10[2]	0x80140c11[2]	0x80140c12[2]	0x80140c14[2]	0x80140c15[2]	0x80140c16[2]
PB[3]	0x80140c10[3]	0x80140c11[3]	0x80140c12[3]	0x80140c14[3]	0x80140c15[3]	0x80140c16[3]
PB[4]	0x80140c10[4]	0x80140c11[4]	0x80140c12[4]	0x80140c14[4]	0x80140c15[4]	0x80140c16[4]
PB[5]	0x80140c10[5]	0x80140c11[5]	0x80140c12[5]	0x80140c14[5]	0x80140c15[5]	0x80140c16[5]
PB[6]	0x80140c10[6]	0x80140c11[6]	0x80140c12[6]	0x80140c14[6]	0x80140c15[6]	0x80140c16[6]
PB[7]	0x80140c10[7]	0x80140c11[7]	0x80140c12[7]	0x80140c14[7]	0x80140c15[7]	0x80140c16[7]

Pad	Input	IE	OEN	Polarity	DS	Act as GPIO
PC[0]	0x80140c20[0]	ana_0xbd[0]	0x80140c22[0]	0x80140c24[0]	ana_0xbf[0]	0x80140c26[0]
PC[1]	0x80140c20[1]	ana_0xbd[1]	0x80140c22[1]	0x80140c24[1]	ana_0xbf[1]	0x80140c26[1]
PC[2]	0x80140c20[2]	ana_0xbd[2]	0x80140c22[2]	0x80140c24[2]	ana_0xbf[2]	0x80140c26[2]
PC[3]	0x80140c20[3]	ana_0xbd[3]	0x80140c22[3]	0x80140c24[3]	ana_0xbf[3]	0x80140c26[3]
PC[4]	0x80140c20[4]	ana_0xbd[4]	0x80140c22[4]	0x80140c24[4]	ana_0xbf[4]	0x80140c26[4]
PC[5]	0x80140c20[5]	ana_0xbd[5]	0x80140c22[5]	0x80140c24[5]	ana_0xbf[5]	0x80140c26[5]
PC[6]	0x80140c20[6]	ana_0xbd[6]	0x80140c22[6]	0x80140c24[6]	ana_0xbf[6]	0x80140c26[6]
PC[7]	0x80140c20[7]	ana_0xbd[7]	0x80140c22[7]	0x80140c24[7]	ana_0xbf[7]	0x80140c26[7]
PD[0]	0x80140c30[0]	ana_0xc2[0]	0x80140c32[0]	0x80140c34[0]	ana_0xc3[0]	0x80140c36[0]
PD[1]	0x80140c30[1]	ana_0xc2[1]	0x80140c32[1]	0x80140c34[1]	ana_0xc3[1]	0x80140c36[1]
PD[2]	0x80140c30[2]	ana_0xc2[2]	0x80140c32[2]	0x80140c34[2]	ana_0xc3[2]	0x80140c36[2]
PD[3]	0x80140c30[3]	ana_0xc2[3]	0x80140c32[3]	0x80140c34[3]	ana_0xc3[3]	0x80140c36[3]
PD[4]	0x80140c30[4]	ana_0xc2[4]	0x80140c32[4]	0x80140c34[4]	ana_0xc3[4]	0x80140c36[4]
PD[5]	0x80140c30[5]	ana_0xc2[5]	0x80140c32[5]	0x80140c34[5]	ana_0xc3[5]	0x80140c36[5]
PD[6]	0x80140c30[6]	ana_0xc2[6]	0x80140c32[6]	0x80140c34[6]	ana_0xc3[6]	0x80140c36[6]
PD[7]	0x80140c30[7]	ana_0xc2[7]	0x80140c32[7]	0x80140c34[7]	ana_0xc3[7]	0x80140c36[7]
PE[0]	0x80140c40[0]	0x80140c41[0]	0x80140c42[0]	0x80140c44[0]	0x80140c45[0]	0x80140c46[0]
PE[1]	0x80140c40[1]	0x80140c41[1]	0x80140c42[1]	0x80140c44[1]	0x80140c45[1]	0x80140c46[1]
PE[2]	0x80140c40[2]	0x80140c41[2]	0x80140c42[2]	0x80140c44[2]	0x80140c45[2]	0x80140c46[2]
PE[3]	0x80140c40[3]	0x80140c41[3]	0x80140c42[3]	0x80140c44[3]	0x80140c45[3]	0x80140c46[3]
PE[4]	0x80140c40[4]	0x80140c41[4]	0x80140c42[4]	0x80140c44[4]	0x80140c45[4]	0x80140c46[4]
PE[5]	0x80140c40[5]	0x80140c41[5]	0x80140c42[5]	0x80140c44[5]	0x80140c45[5]	0x80140c46[5]
PE[6]	0x80140c40[6]	0x80140c41[6]	0x80140c42[6]	0x80140c44[6]	0x80140c45[6]	0x80140c46[6]
PE[7]	0x80140c40[7]	0x80140c41[7]	0x80140c42[7]	0x80140c44[7]	0x80140c45[7]	0x80140c46[7]
PF[0]	0x80140c50[0]	0x80140c51[0]	0x80140c52[0]	0x80140c54[0]	0x80140c55[0]	0x80140c56[0]
PF[1]	0x80140c50[1]	0x80140c51[1]	0x80140c52[1]	0x80140c54[1]	0x80140c55[1]	0x80140c56[1]
PF[2]	0x80140c50[2]	0x80140c51[2]	0x80140c52[2]	0x80140c54[2]	0x80140c55[2]	0x80140c56[2]
PF[3]	0x80140c50[3]	0x80140c51[3]	0x80140c52[3]	0x80140c54[3]	0x80140c55[3]	0x80140c56[3]

Pad	Input	IE	OEN	Polarity	DS	Act as GPIO
PF[4]	0x80140c50[4]	0x80140c51[4]	0x80140c52[4]	0x80140c54[4]	0x80140c55[4]	0x80140c56[4]
PF[5]	0x80140c50[5]	0x80140c51[5]	0x80140c52[5]	0x80140c54[5]	0x80140c55[5]	0x80140c56[5]
PF[6]	0x80140c50[6]	0x80140c51[6]	0x80140c52[6]	0x80140c54[6]	0x80140c55[6]	0x80140c56[6]
PF[7]	0x80140c50[7]	0x80140c51[7]	0x80140c52[7]	0x80140c54[7]	0x80140c55[7]	0x80140c56[7]

Table 11-4 GPIO Setting 2

Pad	Pull down	Pull up	Output set	Output clear	Output toggle
PA[0]	0x80140c0a[0]	0x80140c0b[0]	0x80140c0c[0]	0x80140c0d[0]	0x80140c0e[0]
PA[1]	0x80140c0a[1]	0x80140c0b[1]	0x80140c0c[1]	0x80140c0d[1]	0x80140c0e[1]
PA[2]	0x80140c0a[2]	0x80140c0b[2]	0x80140c0c[2]	0x80140c0d[2]	0x80140c0e[2]
PA[3]	0x80140c0a[3]	0x80140c0b[3]	0x80140c0c[3]	0x80140c0d[3]	0x80140c0e[3]
PA[4]	0x80140c0a[4]	0x80140c0b[4]	0x80140c0c[4]	0x80140c0d[4]	0x80140c0e[4]
PA[5]	0x80140c0a[5]	0x80140c0b[5]	0x80140c0c[5]	0x80140c0d[5]	0x80140c0e[5]
PA[6]	0x80140c0a[6]	0x80140c0b[6]	0x80140c0c[6]	0x80140c0d[6]	0x80140c0e[6]
PA[7]	0x80140c0a[7]	0x80140c0b[7]	0x80140c0c[7]	0x80140c0d[7]	0x80140c0e[7]
PB[0]	0x80140c1a[0]	0x80140c1b[0]	0x80140c1c[0]	0x80140c1d[0]	0x80140c1e[0]
PB[1]	0x80140c1a[1]	0x80140c1b[1]	0x80140c1c[1]	0x80140c1d[1]	0x80140c1e[1]
PB[2]	0x80140c1a[2]	0x80140c1b[2]	0x80140c1c[2]	0x80140c1d[2]	0x80140c1e[2]
PB[3]	0x80140c1a[3]	0x80140c1b[3]	0x80140c1c[3]	0x80140c1d[3]	0x80140c1e[3]
PB[4]	0x80140c1a[4]	0x80140c1b[4]	0x80140c1c[4]	0x80140c1d[4]	0x80140c1e[4]
PB[5]	0x80140c1a[5]	0x80140c1b[5]	0x80140c1c[5]	0x80140c1d[5]	0x80140c1e[5]
PB[6]	0x80140c1a[6]	0x80140c1b[6]	0x80140c1c[6]	0x80140c1d[6]	0x80140c1e[6]
PB[7]	0x80140c1a[7]	0x80140c1b[7]	0x80140c1c[7]	0x80140c1d[7]	0x80140c1e[7]
PC[0]	ana_0xc0[0]	ana_0xc1[0]	0x80140c2c[0]	0x80140c2d[0]	0x80140c2e[0]
PC[1]	ana_0xc0[1]	ana_0xc1[1]	0x80140c2c[1]	0x80140c2d[1]	0x80140c2e[1]
PC[2]	ana_0xc0[2]	ana_0xc1[2]	0x80140c2c[2]	0x80140c2d[2]	0x80140c2e[2]
PC[3]	ana_0xc0[3]	ana_0xc1[3]	0x80140c2c[3]	0x80140c2d[3]	0x80140c2e[3]
PC[4]	ana_0xc0[4]	ana_0xc1[4]	0x80140c2c[4]	0x80140c2d[4]	0x80140c2e[4]
PC[5]	ana_0xc0[5]	ana_0xc1[5]	0x80140c2c[5]	0x80140c2d[5]	0x80140c2e[5]

Pad	Pull down	Pull up	Output set	Output clear	Output toggle
PC[6]	ana_0xc0[6]	ana_0xc1[6]	0x80140c2c[6]	0x80140c2d[6]	0x80140c2e[6]
PC[7]	ana_0xc0[7]	ana_0xc1[7]	0x80140c2c[7]	0x80140c2d[7]	0x80140c2e[7]
PD[0]	ana_0xc4[0]	ana_0xc5[0]	0x80140c3c[0]	0x80140c3d[0]	0x80140c3e[0]
PD[1]	ana_0xc4[1]	ana_0xc5[1]	0x80140c3c[1]	0x80140c3d[1]	0x80140c3e[1]
PD[2]	ana_0xc4[2]	ana_0xc5[2]	0x80140c3c[2]	0x80140c3d[2]	0x80140c3e[2]
PD[3]	ana_0xc4[3]	ana_0xc5[3]	0x80140c3c[3]	0x80140c3d[3]	0x80140c3e[3]
PD[4]	ana_0xc4[4]	ana_0xc5[4]	0x80140c3c[4]	0x80140c3d[4]	0x80140c3e[4]
PD[5]	ana_0xc4[5]	ana_0xc5[5]	0x80140c3c[5]	0x80140c3d[5]	0x80140c3e[5]
PD[6]	ana_0xc4[6]	ana_0xc5[6]	0x80140c3c[6]	0x80140c3d[6]	0x80140c3e[6]
PD[7]	ana_0xc4[7]	ana_0xc5[7]	0x80140c3c[7]	0x80140c3d[7]	0x80140c3e[7]
PE[0]	0x80140c4a[0]	0x80140c4b[0]	0x80140c4c[0]	0x80140c4d[0]	0x80140c4e[0]
PE[1]	0x80140c4a[1]	0x80140c4b[1]	0x80140c4c[1]	0x80140c4d[1]	0x80140c4e[1]
PE[2]	0x80140c4a[2]	0x80140c4b[2]	0x80140c4c[2]	0x80140c4d[2]	0x80140c4e[2]
PE[3]	0x80140c4a[3]	0x80140c4b[3]	0x80140c4c[3]	0x80140c4d[3]	0x80140c4e[3]
PE[4]	0x80140c4a[4]	0x80140c4b[4]	0x80140c4c[4]	0x80140c4d[4]	0x80140c4e[4]
PE[5]	0x80140c4a[5]	0x80140c4b[5]	0x80140c4c[5]	0x80140c4d[5]	0x80140c4e[5]
PE[6]	0x80140c4a[6]	0x80140c4b[6]	0x80140c4c[6]	0x80140c4d[6]	0x80140c4e[6]
PE[7]	0x80140c4a[7]	0x80140c4b[7]	0x80140c4c[7]	0x80140c4d[7]	0x80140c4e[7]
PF[0]	0x80140c5a[0]	0x80140c5b[0]	0x80140c5c[0]	0x80140c5d[0]	0x80140c5e[0]
PF[1]	0x80140c5a[1]	0x80140c5b[1]	0x80140c5c[1]	0x80140c5d[1]	0x80140c5e[1]
PF[2]	0x80140c5a[2]	0x80140c5b[2]	0x80140c5c[2]	0x80140c5d[2]	0x80140c5e[2]
PF[3]	0x80140c5a[3]	0x80140c5b[3]	0x80140c5c[3]	0x80140c5d[3]	0x80140c5e[3]
PF[4]	0x80140c5a[4]	0x80140c5b[4]	0x80140c5c[4]	0x80140c5d[4]	0x80140c5e[4]
PF[5]	0x80140c5a[5]	0x80140c5b[5]	0x80140c5c[5]	0x80140c5d[5]	0x80140c5e[5]
PF[6]	0x80140c5a[6]	0x80140c5b[6]	0x80140c5c[6]	0x80140c5d[6]	0x80140c5e[6]
PF[7]	0x80140c5a[7]	0x80140c5b[7]	0x80140c5c[7]	0x80140c5d[7]	0x80140c5e[7]

NOTE:

- IE: Input enable, high active. 1: enable input, 0: disable input.
- OEN: Output enable, low active. 0: enable output, 1: disable output.
- Output set, Output clear and Output toggle: configure GPO output.
- Input: read GPI input.
- DS: Drive strength. 1: maximum DS level (default), 0: minimal DS level.
- Act as GPIO: enable (1) or disable (0) GPIO function.
- Polarity: By configuring "Polarity" registers, user can determine GPIO edges in Timer modes. In Timer Mode 1, it determines GPIO edge when Timer Tick counting increases. In Timer Mode 2, it determines GPIO edge when Timer Tick starts counting. Users can read addresses to see which GPIO asserts counting signals (Mode 1) / control signal (Mode 2) for Timers.

Table 11-5 GPIO Function Mux Configuration Registers

Address	Type	Description	Default Value
0x80140c70	RW	[6:0]: function control bits of PA0_FS	0x00
0x80140c71	RW	[6:0]: function control bits of PA1_FS	0x00
0x80140c72	RW	[6:0]: function control bits of PA2_FS	0x00
0x80140c73	RW	[6:0]: function control bits of PA3_FS	0x00
0x80140c74	RW	[6:0]: function control bits of PA4_FS	0x00
0x80140c75	R	[6:0]: function control bits of PA5_FS	0x00
0x80140c76	R	[6:0]: function control bits of PA6_FS	0x00
0x80140c77	R	[6:0]: function control bits of PA7_FS	0x00
0x80140c78	RW	[6:0]: function control bits of PB0_FS	0x00
0x80140c79	RW	[6:0]: function control bits of PB1_FS	0x00
0x80140c7a	RW	[6:0]: function control bits of PB2_FS	0x00
0x80140c7b	RW	[6:0]: function control bits of PB3_FS	0x00
0x80140c7c	RW	[6:0]: function control bits of PB4_FS	0x00
0x80140c7d	RW	[6:0]: function control bits of PB5_FS	0x00
0x80140c7e	RW	[6:0]: function control bits of PB6_FS	0x00
0x80140c7f	RW	[6:0]: function control bits of PB7_FS	0x00
0x80140c80	RW	[6:0]: function control bits of PC0_FS	0x00
0x80140c81	RW	[6:0]: function control bits of PC1_FS	0x00
0x80140c82	RW	[6:0]: function control bits of PC2_FS	0x00

Address	Type	Description	Default Value
0x80140c83	RW	[6:0]: function control bits of PC3_FS	0x00
0x80140c84	RW	[6:0]: function control bits of PC4_FS	0x00
0x80140c85	RW	[6:0]: function control bits of PC5_FS	0x00
0x80140c86	RW	[6:0]: function control bits of PC6_FS	0x00
0x80140c87	RW	[6:0]: function control bits of PC7_FS	0x00
0x80140c88	RW	[6:0]: function control bits of PD0_FS	0x00
0x80140c89	RW	[6:0]: function control bits of PD1_FS	0x00
0x80140c8a	RW	[6:0]: function control bits of PD2_FS	0x00
0x80140c8b	RW	[6:0]: function control bits of PD3_FS	0x00
0x80140c8c	RW	[6:0]: function control bits of PD4_FS	0x00
0x80140c8d	RW	[6:0]: function control bits of PD5_FS	0x00
0x80140c8e	RW	[6:0]: function control bits of PD6_FS	0x00
0x80140c8f	RW	[6:0]: function control bits of PD7_FS	0x00
0x80140c90	R	[6:0]: function control bits of PE0_FS	0x00
0x80140c91	R	[6:0]: function control bits of PE1_FS	0x00
0x80140c92	R	[6:0]: function control bits of PE2_FS	0x00
0x80140c93	R	[6:0]: function control bits of PE3_FS	0x00
0x80140c94	R	[6:0]: function control bits of PE4_FS	0x00
0x80140c95	R	[6:0]: function control bits of PE5_FS	0x00
0x80140c96	RW	[6:0]: function control bits of PE6_FS	0x00
0x80140c97	RW	[6:0]: function control bits of PE7_FS	0x00
0x80140c98	RW	[6:0]: function control bits of PF0_FS	0x00
0x80140c99	RW	[6:0]: function control bits of PF1_FS	0x00
0x80140c9a	RW	[6:0]: function control bits of PF2_FS	0x00
0x80140c9b	RW	[6:0]: function control bits of PF3_FS	0x00
0x80140c9c	RW	[6:0]: function control bits of PF4_FS	0x00
0x80140c9d	RW	[6:0]: function control bits of PF5_FS	0x00
0x80140c9e	RW	[6:0]: function control bits of PF6_FS	0x00

Address	Type	Description	Default Value
0x80140c9f	RW	[6:0]: function control bits of PF7_FS	0x00

11.1.3 Drive Strength

The registers in the "DS" column are used to configure the corresponding pin's driving strength: "1" indicates maximum drive level, while "0" indicates minimal drive level.

The "DS" configuration takes effect when the pin is used as output. It's set as the strongest driving level by default. In actual applications, driving strength can be decreased to lower level if necessary.

- Standard drive strength (suitable for GPIO pins PA[0:7], PB[0:7], PC[4:7], PD[0:7], PE[0:7], PF[0:7])
  - "DS" = 1, maximum drive strength = 8 mA under 3.3V, 3 mA under 1.8V
  - "DS" = 0, minimum drive strength = 4 mA under 3.3V, 1.5 mA under 1.8V
- High drive strength (suitable for GPIO pins PC[0:3])
  - "DS" = 1, maximum drive strength = 16 mA under 3.3V, 5.5 mA under 1.8V
  - "DS" = 0, minimum drive strength = 12 mA under 3.3V, 4 mA under 1.8V

11.1.4 Connection Relationship between GPIO and Related Modules

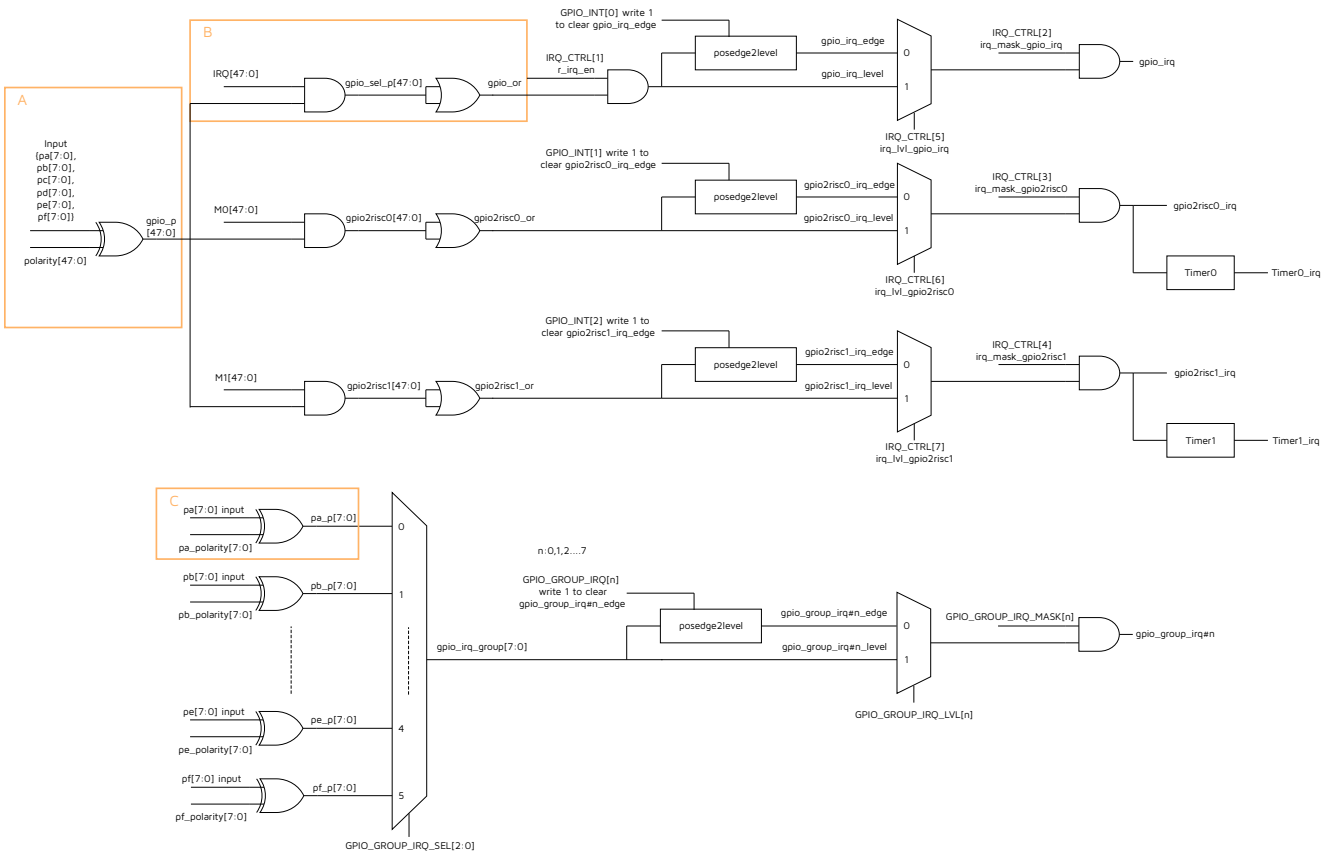
GPIO can be used to generate GPIO interrupt signal for interrupt system, counting or control signal for Timer/ Counter module, gpio2risc interrupt signal for interrupt system and GPIO group interrupt signal.

For the "Exclusive Or (XOR)" operation result for input signal from any GPIO pin other than flash IO and respective "Polarity" value, on one hand, it takes "And" operation with "irq" and generates GPIO interrupt request signal; on the other hand, it takes "And" operation with "M0/M1", and generates counting signal in Mode 1 or control signal in Mode 2 for Timer0/Timer1, or generates GPIO2RISC[0]/GPIO2RISC[1] interrupt request signal.

- **gpio_irq**: GPIO interrupt request signal = | ((Input ^ Polarity) & IRQ), it is the interrupt request signal generated from GPIO;
- **gpio2risc[0]_irq**: GPIO2RISC[0] interrupt request signal = | ((Input ^ Polarity) & M0), it is the interrupt request signal generated from GPIO and M0 registers;
- **timer0_irq**: Counting (Mode 1) or control (Mode 2) signal for Timer0 = | ((Input ^ Polarity) & M0), it is the interrupt request signal generated from GPIO, M0 and Timer0;
- **gpio2risc[1]_irq**: GPIO2RISC[1] interrupt request signal = | ((Input ^ Polarity) & M1), it is the interrupt request signal generated from GPIO and M1 registers;
- **timer1_irq**: Counting (Mode 1) or control (Mode 2) signal for Timer1 = | ((input ^ polarity) & M1), it is the interrupt request signal generated from GPIO, M0 and Timer1;
- **gpio_group_irq**: gpio_group_irq[7:0] interrupt request signals are from a set of GPIO, which can be configured via registers GPIO_GROUP_IRQ_SEL[2:0].

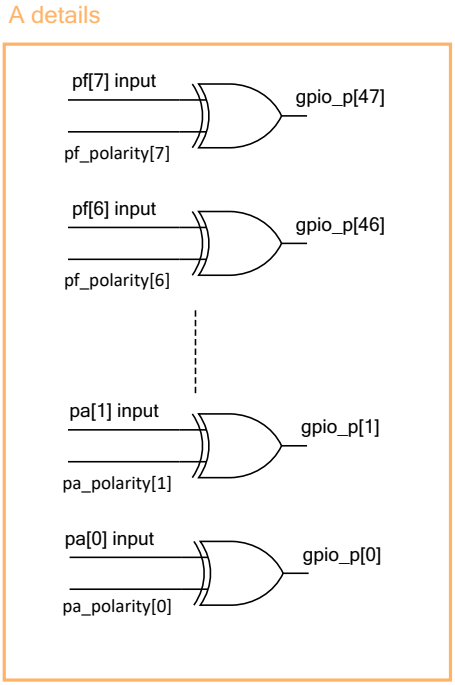
The logic relationship is shown in figure below.

Figure 11-1 Logic Relationship between GPIO and Related Modules



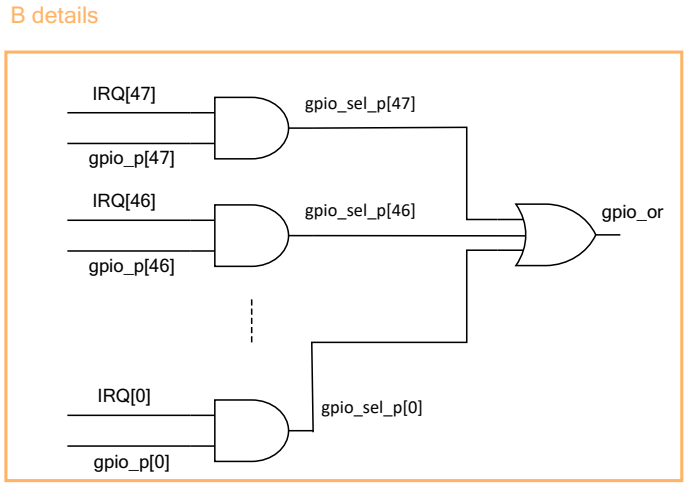
The detailed digital circuit for part A in Figure 11-1 is shown as below. The input register and polarity register of each corresponding GPIO pin take XOR operation to get one gpio_p signal. The output of this part is gpio_p[47:0] which is a set of 48 bits signals.

Figure 11-2 Detailed Circuit for Part A



The detailed digital circuit for part B in Figure 11-1 is shown as below. The IRQ register and gpio_p signal of each corresponding GPIO pin take AND operation to get one gpio_sel_p signal, then 48 bits gpio_sel_p signals take OR operation to get one gpio_or signal. The output of this part is gpio_or which is the 1 bit signal.

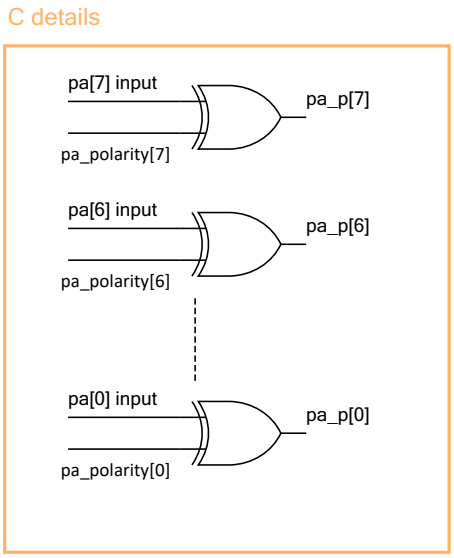
Figure 11-3 Detailed Circuit for Part B



For the detailed digital circuit of M0[47:0] to gpio2risc[0]_irq and M1[47:0] to gpio2risc[1]_irq, the operations are similar with the detailed circuit of part B.

The detailed digital circuit for part C in Figure 11-1 is shown as below. The output of this part is pa_p[7:0] which is a set of 8 bits signals.

Figure 11-4 Detailed Circuit for Part C



Similarly, the output of the circuits below part C are pb_p[7:0], pc_p[7:0], pd_p[7:0], pe_p[7:0], and pf_p[7:0]. The 6 groups of px_pol signals mux with GPIO_GROUP_IRQ_SEL[2:0] to get gpio_irq_group[7:0] signals.

Please note that gpio_irq_group[7] is the result from pa_p[7], pb_p[7], pc_p[7], pd_p[7], pe_p[7], pf_p[7] and GPIO_GROUP_IRQ_SEL[2:0], and similarly, gpio_irq_group[6] is the result from pa_p[6], pb_p[6], pc_p[6], pd_p[6], pe_p[6], pf_p[6] and GPIO_GROUP_IRQ_SEL[2:0], and so on. This is different from the above interrupt request signals shown in Figure 11-1, they are the operation results from all 48 GPIOs.

11.1.5 GPIO IRQ Signal

Select GPIO interrupt trigger edge (positive edge or negative edge) via configuring "Polarity", and set corresponding GPIO interrupt enabling bit "Irq".

11.1.6 GPIO2RISC IRQ Signal

Select GPIO2RISC interrupt trigger edge (positive edge or negative edge) via configuring "Polarity", and set corresponding GPIO enabling bit "m0"/"m1", then enable GPIO2RISC[0]/GPIO2RISC[1] interrupt.

Table 11-6 GPIO IRQ Table

Pad	Input	IRQ	m0	m1	Polarity
PA[0]	0x80140c00[0]	0x80140c07[0]	0x80140c08[0]	0x80140c09[0]	0x80140c04[0]
PA[1]	0x80140c00[1]	0x80140c07[1]	0x80140c08[1]	0x80140c09[1]	0x80140c04[1]
PA[2]	0x80140c00[2]	0x80140c07[2]	0x80140c08[2]	0x80140c09[2]	0x80140c04[2]
PA[3]	0x80140c00[3]	0x80140c07[3]	0x80140c08[3]	0x80140c09[3]	0x80140c04[3]
PA[4]	0x80140c00[4]	0x80140c07[4]	0x80140c08[4]	0x80140c09[4]	0x80140c04[4]
PA[5]	0x80140c00[5]	0x80140c07[5]	0x80140c08[5]	0x80140c09[5]	0x80140c04[5]
PA[6]	0x80140c00[6]	0x80140c07[6]	0x80140c08[6]	0x80140c09[6]	0x80140c04[6]
PA[7]	0x80140c00[7]	0x80140c07[7]	0x80140c08[7]	0x80140c09[7]	0x80140c04[7]
PB[0]	0x80140c10[0]	0x80140c17[0]	0x80140c18[0]	0x80140c19[0]	0x80140c14[0]
PB[1]	0x80140c10[1]	0x80140c17[1]	0x80140c18[1]	0x80140c19[1]	0x80140c14[1]
PB[2]	0x80140c10[2]	0x80140c17[2]	0x80140c18[2]	0x80140c19[2]	0x80140c14[2]
PB[3]	0x80140c10[3]	0x80140c17[3]	0x80140c18[3]	0x80140c19[3]	0x80140c14[3]
PB[4]	0x80140c10[4]	0x80140c17[4]	0x80140c18[4]	0x80140c19[4]	0x80140c14[4]
PB[5]	0x80140c10[5]	0x80140c17[5]	0x80140c18[5]	0x80140c19[5]	0x80140c14[5]
PB[6]	0x80140c10[6]	0x80140c17[6]	0x80140c18[6]	0x80140c19[6]	0x80140c14[6]
PB[7]	0x80140c10[7]	0x80140c17[7]	0x80140c18[7]	0x80140c19[7]	0x80140c14[7]
PC[0]	0x80140c20[0]	0x80140c27[0]	0x80140c28[0]	0x80140c29[0]	0x80140c24[0]
PC[1]	0x80140c20[1]	0x80140c27[1]	0x80140c28[1]	0x80140c29[1]	0x80140c24[1]
PC[2]	0x80140c20[2]	0x80140c27[2]	0x80140c28[2]	0x80140c29[2]	0x80140c24[2]
PC[3]	0x80140c20[3]	0x80140c27[3]	0x80140c28[3]	0x80140c29[3]	0x80140c24[3]
PC[4]	0x80140c20[4]	0x80140c27[4]	0x80140c28[4]	0x80140c29[4]	0x80140c24[4]
PC[5]	0x80140c20[5]	0x80140c27[5]	0x80140c28[5]	0x80140c29[5]	0x80140c24[5]

Pad	Input	IRQ	m0	m1	Polarity
PC[6]	0x80140c20[6]	0x80140c27[6]	0x80140c28[6]	0x80140c29[6]	0x80140c24[6]
PC[7]	0x80140c20[7]	0x80140c27[7]	0x80140c28[7]	0x80140c29[7]	0x80140c24[7]
PD[0]	0x80140c30[0]	0x80140c37[0]	0x80140c38[0]	0x80140c39[0]	0x80140c34[0]
PD[1]	0x80140c30[1]	0x80140c37[1]	0x80140c38[1]	0x80140c39[1]	0x80140c34[1]
PD[2]	0x80140c30[2]	0x80140c37[2]	0x80140c38[2]	0x80140c39[2]	0x80140c34[2]
PD[3]	0x80140c30[3]	0x80140c37[3]	0x80140c38[3]	0x80140c39[3]	0x80140c34[3]
PD[4]	0x80140c30[4]	0x80140c37[4]	0x80140c38[4]	0x80140c39[4]	0x80140c34[4]
PD[5]	0x80140c30[5]	0x80140c37[5]	0x80140c38[5]	0x80140c39[5]	0x80140c34[5]
PD[6]	0x80140c30[6]	0x80140c37[6]	0x80140c38[6]	0x80140c39[6]	0x80140c34[6]
PD[7]	0x80140c30[7]	0x80140c37[7]	0x80140c38[7]	0x80140c39[7]	0x80140c34[7]
PE[0]	0x80140c40[0]	0x80140c47[0]	0x80140c48[0]	0x80140c49[0]	0x80140c44[0]
PE[1]	0x80140c40[1]	0x80140c47[1]	0x80140c48[1]	0x80140c49[1]	0x80140c44[1]
PE[2]	0x80140c40[2]	0x80140c47[2]	0x80140c48[2]	0x80140c49[2]	0x80140c44[2]
PE[3]	0x80140c40[3]	0x80140c47[3]	0x80140c48[3]	0x80140c49[3]	0x80140c44[3]
PE[4]	0x80140c40[4]	0x80140c47[4]	0x80140c48[4]	0x80140c49[4]	0x80140c44[4]
PE[5]	0x80140c40[5]	0x80140c47[5]	0x80140c48[5]	0x80140c49[5]	0x80140c44[5]
PE[6]	0x80140c40[6]	0x80140c47[6]	0x80140c48[6]	0x80140c49[6]	0x80140c44[6]
PE[7]	0x80140c40[7]	0x80140c47[7]	0x80140c48[7]	0x80140c49[7]	0x80140c44[7]
PF[0]	0x80140c50[0]	0x80140c57[0]	0x80140c58[0]	0x80140c59[0]	0x80140c54[0]
PF[1]	0x80140c50[1]	0x80140c57[1]	0x80140c58[1]	0x80140c59[1]	0x80140c54[1]
PF[2]	0x80140c50[2]	0x80140c57[2]	0x80140c58[2]	0x80140c59[2]	0x80140c54[2]
PF[3]	0x80140c50[3]	0x80140c57[3]	0x80140c58[3]	0x80140c59[3]	0x80140c54[3]
PF[4]	0x80140c50[4]	0x80140c57[4]	0x80140c58[4]	0x80140c59[4]	0x80140c54[4]
PF[5]	0x80140c50[5]	0x80140c57[5]	0x80140c58[5]	0x80140c59[5]	0x80140c54[5]
PF[6]	0x80140c50[6]	0x80140c57[6]	0x80140c58[6]	0x80140c59[6]	0x80140c54[6]
PF[7]	0x80140c50[7]	0x80140c57[7]	0x80140c58[7]	0x80140c59[7]	0x80140c54[7]

11.1.7 GPIO GROUP IRQ Signal

Select GPIO GROUP interrupt trigger edge (positive edge or negative edge) via configuring "Polarity", and select a set of GPIO as interrupt source via configuring "gpio_group_sel".

11.1.8 GPIO Interrupt Configuration Process

The GPIO_IRQ/GPIO2RISC0_IRQ/GPIO2RISC1_IRQ interrupt configuration process is as follows:

Step 1 Set the "Act as GPIO" register of the corresponding pin to 1 to configure the pin for GPIO function. For example, set PA0 to GPIO function: 0x80140c06[0] = 1'b1.

Step 2 Configure the pull-up and pull-down function of the pin according to the trigger types: if it is triggered by high level/rising edge, configure the pin to pull down; if it is triggered by low level/falling edge, configure the pin to pull up, and enable the input function at the same time. For example, set PA0 to 1M ohm pull up: AFE_0X17[1:0] = 2'b01; enable the input function of PA0: 0x80140c01[0] = 1'b1.

Step 3 Set IRQ/M0/M1 of the pin to 1. For example, set PA0 as IRQ interrupt: 0x80140c07[0] = 1'b1.

Step 4 Configure irq_lvl_gpio_irq (IRQ_CTRL[5]) and the corresponding Polarity of the pins according to the trigger types. For example, set PA0 as a rising edge interrupt: 0x80140c04[0]=1'b0, 0x80140ca2[5]=1'b0.

Step 5 Set IRQ_CTRL[1](r_irq_en) to 1 (0x80140ca2[1]=1'b1) If it is an IRQ interrupt. If it is a GPIO2RISC0_IRQ/GPIO2RISC1_IRQ interrupt, there is no need to configure.

Step 6 Clear the corresponding interrupt trigger flag bit in GPIO_INT (write 1 to clear), this is a necessary operation, otherwise an interrupt is triggered by mistake, and this flag in the interrupt handler function also needs to be manually set to 1. For example, set the pin as IRQ interrupt: 0x80140ca8[1]=1'b1.

Step 7 Set the register IRQ_CTRL to enable the corresponding mask. For example, set the pin as IRQ interrupt: 0x80140ca2[2]=1'b1.

Step 8 Enable plic correspondence bit.

Step 9 Enable the general interrupt.

The GPIO_GROUP_IRQ interrupt configuration process is as follows:

Step 1 Set the "Act as GPIO" register of the corresponding pin to 1 to configure the pin for GPIO function. For example, set PA0 to GPIO function: 0x80140c06[0] = 1'b1.

Step 2 Configure the pull-up and pull-down function of the pin according to the trigger types: if it is triggered by high level/rising edge, configure the pin to pull down; if it is triggered by low level/falling edge, configure the pin to pull up, and enable the input function at the same time. For example, set PA0 to 1M ohm pull up: AFE_0X17[1:0] = 2'b01; enable the input function of PA0: 0x80140c01[0] = 1'b1.

Step 3 Configure the register GPIO_GROUP_IRQ_SEL to select the group source of the interrupt, ranging from 0 to 5 (A-F). For example, set PA0 as GPIO_GROUP_IRQ interrupt: GPIO_GROUP_IRQ_SEL[2:0] = 3'b000.

Step 4 Configure the corresponding GPIO_GROUP_IRQ_LVL and Polarity of the pins according to the trigger types. For example, set PA0 as a rising edge interrupt: 0x80140c04[0]=1'b0, 0x80140ca4[0]=1'b0.

Step 5 Clear the corresponding interrupt trigger flag bit in GPIO_GROUP_IRQ (write 1 to clear), this is a necessary operation, otherwise an interrupt is triggered by mistake, and this flag in the interrupt

handler function also needs to be manually set to 1. For example, set PA0 as GPIO_GROUP_IRQ interrupt: 0x80140ca9[0]=1'b1.

Step 6 Set the register GPIO_GROUP_IRQ_MASK to turn on the corresponding mask. For example, set PA0 as GPIO_GROUP_IRQ interrupt: 0x80140ca6[0]=1'b1.

Step 7 Enable plic correspondence bit.

Step 8 Enable the general interrupt.

11.1.9 GPIO Interrupt Related Registers

The GPIO interrupt related registers are listed as following, the base address of the following registers is 0x80140c00. The default values are all 0x00.

Table 11-7 GPIO Interrupt Related Registers

Address Offset	Name	Type	Description	Default Value
0xa2	IRQ_CTRL	R/W	[0]: r_wakeup_en 1: enable, 0: disable, use in suspend mode [1]: r_irq_en 1: gpio irq enable, 0: gpio irq disable [2]: irq_mask_gpio_irq 1: enable, 0: disable [3]:irq_mask_gpio2risc0 1: enable, 0: disable [4]: irq_mask_gpio2risc1 1: enable, 0: disable [5]: irq_lvl_gpio_irq 0: edge trigger 1: level trigger There are four triggering methods for GPIO_IRQ can be configured by setting this bit and the polarity bit of the corresponding pin. RISING_EDGE: polarity: 0, irq_lvl_gpio_irq to: 0 FALLING_EDGE: polarity: 1, irq_lvl_gpio_irq to: 0 HIGH_EDGE: polarity: 0, irq_lvl_gpio_irq to: 1 LOW_EDGE: polarity: 1, irq_lvl_gpio_irq to: 1 [6]: irq_lvl_gpio2risc0 similar to irq_lvl_gpio_irq [7]: irq_lvl_gpio2risc1 similar to irq_lvl_gpio_irq	0x00

Address Offset	Name	Type	Description	Default Value
0xa4	GPIO_GROUP_IRQ_LVL	R/W	[7:0]: gpio_irq_lvl similar to irq_lvl_gpio_irq The following X is generated by GPIO_ GROUP_ IRQ_ SEL determination, value range from A to F. [0]: GPIO_PX0 [1]: GPIO_PX1 [2]: GPIO_PX2 [3]: GPIO_PX3 [4]: GPIO_PX4 [5]: GPIO_PX5 [6]: GPIO_PX6 [7]: GPIO_PX7	0x00
0xa5	GPIO_GROUP_IRQ_SEL	R/W	[2:0]: gpio_irq_sel select the irq group source, the range is 0 to 5 0: GPIO_GROUP_A 1: GPIO_GROUP_B 2: GPIO_GROUP_C 3: GPIO_GROUP_D 4: GPIO_GROUP_E 5: GPIO_GROUP_F	0x00
0xa6	GPIO_GROUP_IRQ_MASK	R/W	[7:0]: gpio_irq_src_mask 1:enable,0:disable [0]: gpio_group_irq_0 [1]: gpio_group_irq_1 [2]: gpio_group_irq_2 [3]: gpio_group_irq_3 [4]: gpio_group_irq_4 [5]: gpio_group_irq_5 [6]: gpio_group_irq_6 [7]: gpio_group_irq_7	0x00

Address Offset	Name	Type	Description	Default Value
0xa8	GPIO_INT	W1C	Interrupt flag bit, which is automatically set by hardware to 1 when an interrupt occurs, user needs to manually write 1 to clear flag status. [0]: gpio_irq [1]: gpio2risc0 [2]: gpio2risc1	0x00
0xa9	GPIO_GROUP_IRQ	W1C	Interrupt flag bit, which is automatically set by hardware to 1 when an interrupt occurs, user needs to manually write 1 to clear flag status. [0]: gpio_group_irq_0 [1]: gpio_group_irq_1 [2]: gpio_group_irq_2 [3]: gpio_group_irq_3 [4]: gpio_group_irq_4 [5]: gpio_group_irq_5 [6]: gpio_group_irq_6 [7]: gpio_group_irq_7	0x00

11.1.10 Pull-up/Pull-down Resistors

All GPIOs support configurable pull-up resistor of rank x1 and x100 or pull-down resistor of rank x10 which are all disabled by default. Analog registers afe_0x17<7:0> ~ afe_0x22<7:0> serve to control the pull-up/pull-down resistor for each GPIO, as shown in table below.

NOTE: *The GPIO pull-up/pull-down resistance is a simulation result by the internal MOSFET and affected by the IO voltage VDDO3. The lower the IO voltage of GPIO, the higher the pull-up/pull-down resistance of GPIO.*

Table 11-8 Analog Registers for Pull-up/Pull-down Resistor Control

Address	Type	Description	Default Value
0x17	R/W	GPIO_A<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000

Address	Type	Description	Default Value
0x18	R/W	GPIO_A<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x19	R/W	GPIO_B<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x1a	R/W	GPIO_B<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x1b	R/W	GPIO_C<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x1c	R/W	GPIO_C<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x1d	R/W	GPIO_D<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up Note: GPIO PD0 is used for audio, its internal impedance is large due to the internal affect of PGA which meets the expectation.	00000000

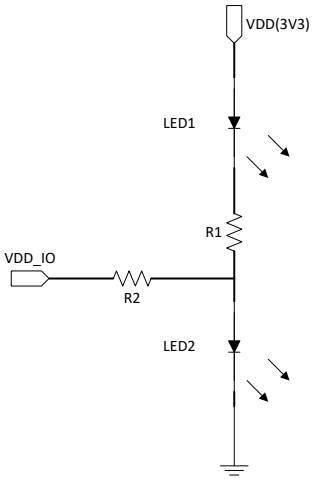
Address	Type	Description	Default Value
0x1e	R/W	GPIO_D<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x1f	R/W	GPIO_E<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x20	R/W	GPIO_E<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x21	R/W	GPIO_F<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x22	R/W	GPIO_F<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000

11.1.11 GPIO Driving LED Description

The LED for RGB (Red, Green, Blue) can be driven by GPIO, the application principle is as follows.

1. Through the three states of output high / output low / input high resistance of the GPIO to drive the LED on and off respectively (no simultaneous bright state);
2. Control LED brightness by 2 resistors.

Figure 11-5 One GPIO drives two LEDs



11.2 Swire

The SoC supports Single Wire Slave interface. SWM (Single Wire Master) and SWS (Single Wire Slave) represent the master and slave device of the single wire communication system developed by Telink. The maximum data rate can be up to 2 Mbps.

SWS usage is not supported in power-saving mode (deep sleep or suspend).

SWS related registers are listed as following, the base address of the following registers is 0x80100c00.

Table 11-9 Swire Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	SWIRE_DATA	R	[7:0]: swire_data	0x00
0x01	SWIRE_CTL	RW	[0]: swire_wr [1]: swire_rd [2]: swire_cmd [3]: swire_err_flag [4]: swire_eop [6]: swire_usb_det [7]: swire_usb_en	0x80
0x02	SWIRE_CTL2	RW	[6:0]: swire_clk_div	0x05
0x03	SWIRE_ID	RW	[4:0]: id_valid [7]: fifo_mode	0x00

11.3 JTAG and SDP

This SoC has debug interfaces of JTAG and SDP. JTAG (Joint Test Action Group) is an interface used for debugging and programming the chip, and it includes four wires (TDI, TDO, TMS and TCK) for this SoC. SDP is a two-wire serial debug interface, which is multiplexed with TCK and TMS.

The specific GPIOs used as JTAG and SDP can be referred to the Table 11-1 GPIO Pad Function Mux.

The bootstrap pin for switching between JTAG and SDP is PB[0] with the following definition:

- PB[0] is high (with pull-up): SDP is enabled;
- PB[0] is low (with pull-down): JTAG is enabled.

When power on after wake up from deep sleep mode, the SoC reads the voltage level of bootstrap pin and decide the debug port. This bootstrap pin cannot be modified through the init process or register.

11.4 I2C

11.4.1 Introduction of I2C

The SoC embeds I2C to implement half-duplex transmission and reception via I2C SDA (serial data line) and SCL (serial clock line) interface. It can be configured to transmit or receive data as master or slave.

The I2C1M can only be configured as the master.

Each device is recognized by a unique address (ID). Master device is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Slave device is the device addressed via a Master. The I2C and I2C1M restriction is that pclk must be at least 10x of data rate.

The I2C features include:

- Supports Standard-mode (100 kbps), Fast-mode (400 kbps) and Fast-mode Plus (1 Mbps)
- Half-duplex operation
- Supports models: Master transmitter, Master receiver, Slave transmitter and Slave receiver
- Supports 7-bit addressing mode
- Supports general call address
- Auto clock stretching
- 8 bytes of transmit/receive FIFOs
- Supports DMA function (RX supports DMA Linked List Pointer)
- 2 x I2C (I2C/I2C1M, The I2C1M supports only the master)

11.4.2 Block Diagram of I2C

The following features show the block diagram of I2C and I2C1M respectively.

Figure 11-6 I2C Block Diagram

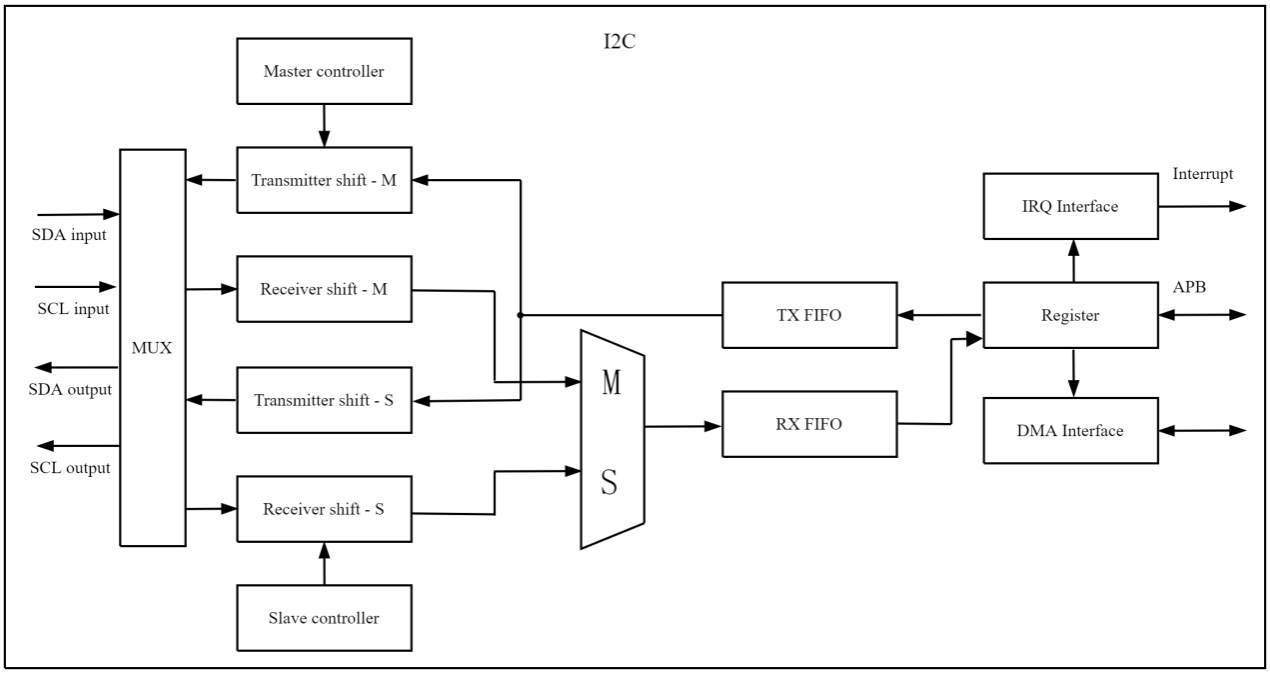
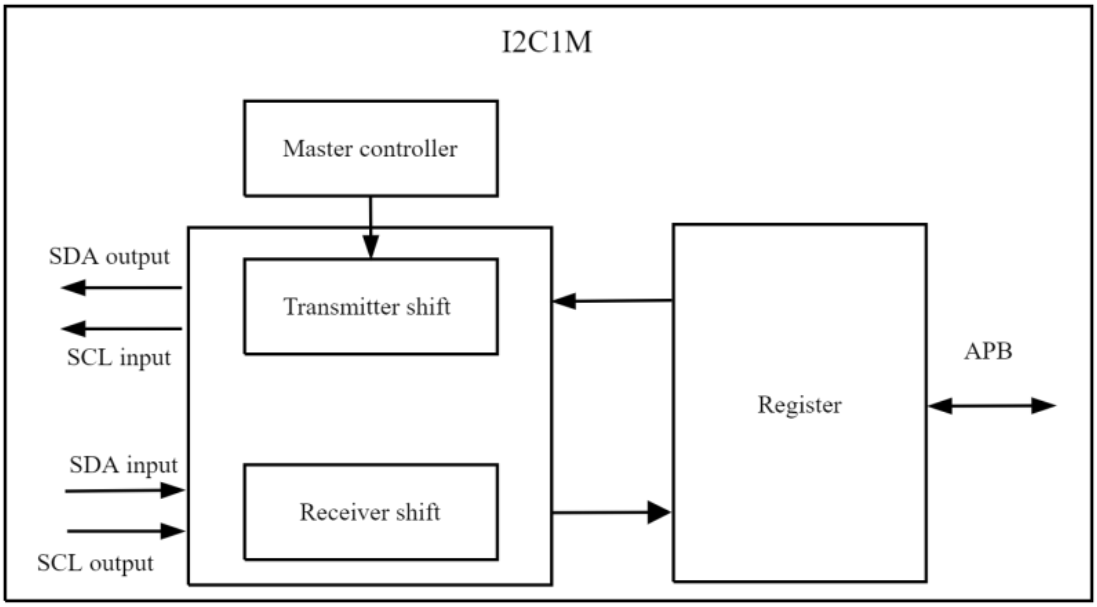


Figure 11-7 I2C1M Block Diagram



11.4.3 I2C Function Description

11.4.3.1 Pin Configuration

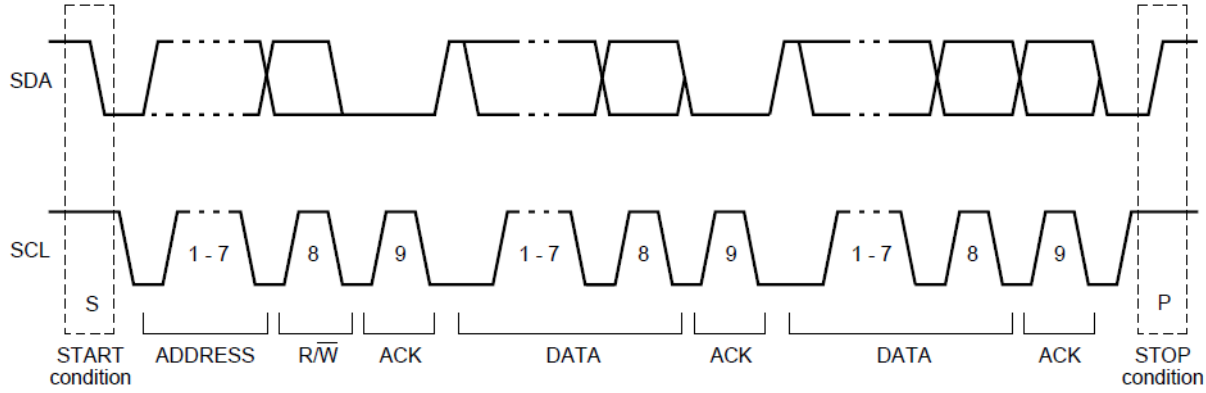
The I2C bidirectional communications require a minimum of two pins: SDA (serial data line) and SCL (serial clock line):

The two wires, SDA and SCL carry information between Master device and Slave device connected to the bus. Both SDA and SCL are bidirectional lines connected to a positive supply voltage via a pull-up resister. It's

recommended to use external 3.3 kOhm pull-up resistor. For standard mode, the internal pull-up resistor of rank x1 can be used instead of the external 3.3 kOhm pull-up.

When the bus is released, both lines are HIGH. It is noted that the data on the SDA line must be stable during the high period of the clock (SCL), and the high or low state of the data line can only change when the clock signal on the SCL line is low.

Figure 11-8 I2C Bus Protocol



11.4.3.2 I2C Master Mode

Register I2CSCT0[1] should be set to 1'b1 to enable I2C master mode.

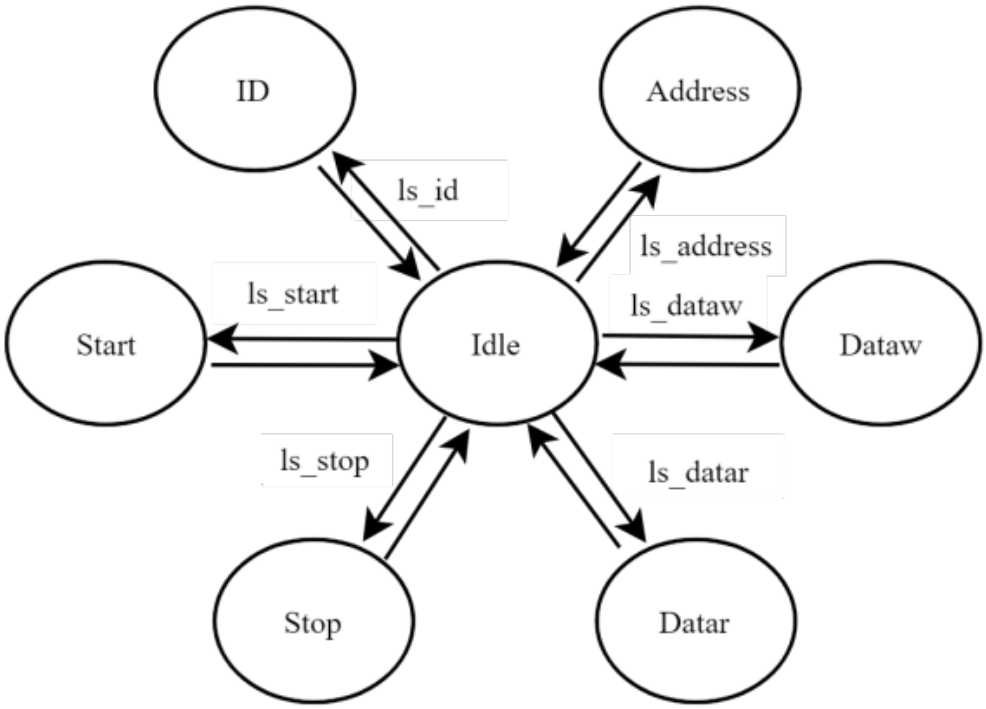
Register I2CSP sets I2C Master clock: F_{I2C} = (pclk / (4 *clock speed configured in register I2CSP).

A complete I2C protocol contains START, Slave Address, R/W bit, data, ACK and STOP. Slave address could be configured via I2C_ID (address 0x01) [7:1].

I2C Master could send START, Slave Address, R/W bit, data and STOP cycle by configuring I2CSCT1. The state machine starts running and transfer data in the correct sequence.

Register I2CMST serves to indicate whether Master/Master packet is busy, as well as Master received status. Bit[0] is set to 1 when other states are running except IDLE, and the bit can be automatically cleared after a start signal/ address byte/acknowledge signal/data /stop signal is sent. Bit[1] is set to 1 when the start condition signal is sent, and the bit is automatically cleared after the stop condition signal is sent. Bit[2] indicates whether the response was successful.

Figure 11-9 I2C Master State

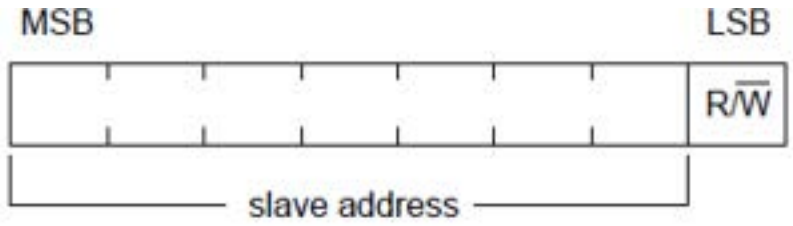


The controller (Master state) provides an efficient way to initiate I2C transactions. Every transaction can be delineated by four phases: Start, Address, Data and Stop. At the Start phase, a START condition is generated. At the Address phase, an address is sent. At the Data phase, one or more data bytes are transferred. At the Stop phase, a STOP condition is generated. The existence of each phase can be controlled independently.

11.4.3.3 I2C Slave Mode

I2C module acts as Slave mode by default. I2C slave address can be configured via register I2C_ID (address 0x01) [7:1], as shown below.

Figure 11-10 Byte Consisted of Slave Address and R/W Flag Bit



I2C slave mode supports two sub modes including Direct Memory Access (DMA) mode and No Direct Memory Access (NDMA).

In I2C Slave mode, Master could initiate transaction anytime. I2C slave module replies with ACK automatically. To monitor the start of I2C transaction, user could set interrupt from GPIO for SDA or SCL.

Read and write format of Slave modes are shown as below.

Figure 11-11 Read Format in Slave Mode

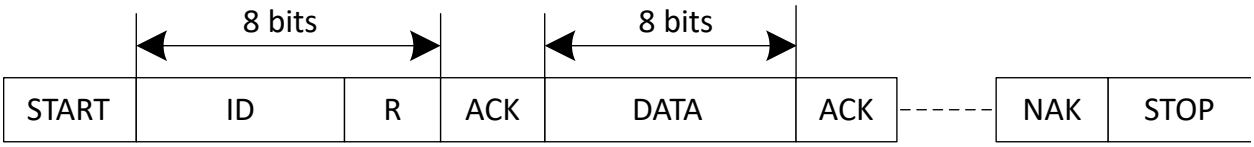
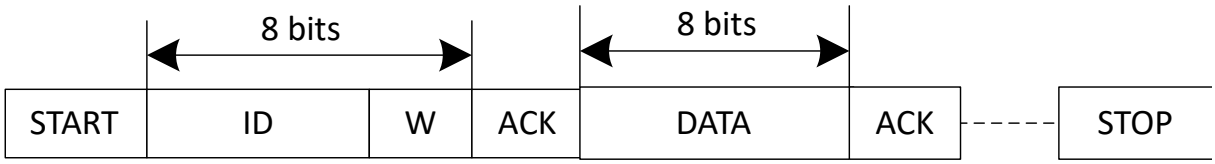
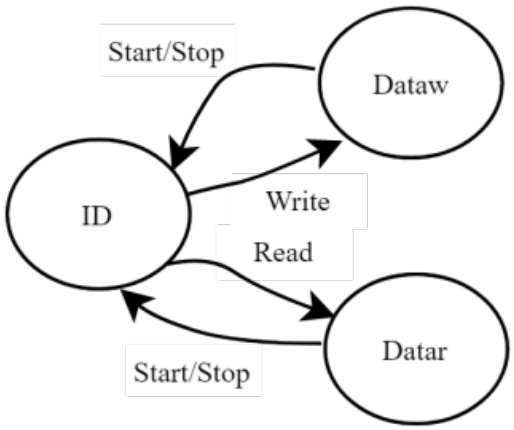


Figure 11-12 Write Format in Slave Mode



DMA and NDMA access buffer through DMA and APB, respectively.

Figure 11-13 I2C Slave State



The controller is addressed when the address byte of an I2C transaction matches the Address Register I2C_ID. An ss_rw_irq interrupt can be generated for the software to prepare for the subsequent operations.

Note: The address match of this chip needs to be optimized. The I2C slave replies ACK if the ID matches, But the data is still stored to the RF FIFO, and an interrupt is asserted. If one master corresponds to multiple slave applications, the recommended solution for I2C slave is as follows:

- If the PAD is sufficient, select a pad as our slave indicator signal, read the status through GPIO. We deal with I2C interrupt or configure DMA when the PAD is pulled up.
- If the PAD is not enough, we can only modify the communication protocol, and the Master needs to send another data as the software ID, and deal with I2C interrupt or configure DMA when the software ID is received.

11.4.3.4 I2C Master Transmitter

(1) NDMA Operation

The transmitter comprises a Transmitter FIFO (TX FIFO), a Transmitter Shift, and a Controller (Master controller). The TX FIFO holds data to be transferred through the serial interface. The TX FIFO can store up to 8 characters depending on hardware configurations and programming settings. The user can determine whether the pointer of the current TX FIFO is less than 8 via the register I2C_BUFCNT[7:4], if less than 8, continue to fill the data to TX FIFO until the end of sending. The Transmitter Shift reads a character from the

TX FIFO for the next transmission. The Transmitter Shift functions as a parallel-to-serial data converter, converting the outgoing character to serial bit streams. For each character transmission, the Controller generates a START bit, some number of Slave address bits, some number of data bits, and a STOP bit. The TX FIFO is by default a 8-byte buffer called Transmitter Buffer Register.

For example, to implement an I2C write transfer with 4-byte data, which contains START, Slave Address (ID), Write bit, ACK from Slave, 1st byte, ACK from slave, 2nd byte, ACK from slave, 3rd byte, ACK from slave,4th byte, ACK from slave and STOP. User needs to configure I2C slave address to I2C_ID[7:1]. To start I2C write transfer, After the register I2CSCT1 is configured to 0x11 (0001 0001), I2C Master launches START, Slave address (ID). The data to TX FIFO and register I2CSCT1 is configured to 0x24 (0000 0024), I2C Master sends TX FIFO data, load ACK to I2CMST[2] and then STOP sequentially.

Note: Address state is optional.

(2) DMA Operation

When the TX FIFO under the threshold 4 characters, the master controller asserts dma_tx_req to request a data transfer. The DMA controller should then transfer data to the TX FIFO followed by asserting dma_tx_ack. Next, the master controller de-asserts dma_tx_req and the DMA controller de-asserts dma_tx_ack. The master controller asserts dma_tx_req again unless the TX FIFO is full or the DMA transmission length is reached.

For example, The data to be sent is put into SRAM, set the TX DMA configuration (For details about TX DMA configuration, refer to the DMA chapter), DMA transfers 4 bytes of data to TX FIFO at a time. User needs to configure I2C slave address to I2C_ID[7:1]. After the register I2CSCT1 is configured to 0x11 (0001 0001), I2C Master launches START and Slave address. Register I2CSCT1 is configured to 0x24 (0000 0024), I2C Master sends TX FIFO data, load ACK to I2CMST[2], and then STOP sequentially.

I2C supports a single write of maximum length supported for a single DMA transfer.

11.4.3.5 I2C Master Receiver

(1) NDMA Operation

The receiver comprises a Receiver FIFO (RX FIFO), Receiver Shift, and a Controller (Master Controller). The received bits are shifted into the Receiver Shift for serial-to-parallel data conversion and the received character is stored into the RX FIFO. The RX FIFO is by default a 8byte buffer called the Receiver Buffer Register. The user can via the register I2C_BUFCNT[3:0] to determine whether the pointer to the current RX FIFO is greater than 0, and if it is greater than 0, the data can be read until the end of receiving.

For example, to implement an I2C read transfer with 4 byte data, which contains START, Slave Address (ID), Read bit, Ack from Slave, 4 byte data from Slave, Ack from master and STOP. User needs to configure I2C slave address to I2C_ID [7:1]. To start I2C read transfer, After the register I2CSCT1 is configured to 0x79 (0111 1001), I2C Master launches START, Slave address (ID), Read bit, load ACK to I2CMST [2], load data to RX FIFO, reply ACK and then STOP sequentially.

Note: Address state is optional.

(2) DMA Operation

When the RX FIFO reaches the threshold 4 characters, the master controller asserts dma_rx_req to request a data transfer. The DMA controller should then transfer data from the RX FIFO followed by asserting dma_rx_ack. Next, the controller de-asserts dma_rx_req and the DMA controller de-asserts dma_rx_ack. The controller asserts dma_rx_req again unless the RX FIFO is empty. DMA relies on rxdone to read parts of the data below 4 characters.

For example, set the RX DMA configuration (For details about RX DMA configuration, refer to the DMA chapter), user needs to configure I2C slave address to I2C_ID[7:1]. After the register I2CSCT1 is configured to 0x79 (0111 1001), I2C Master launches START, Slave address, Read bit, load ACK to I2CMST [2], load data to RX FIFO, reply ACK and then STOP sequentially.

I2C supports a single read of maximum length supported for a single DMA transfer.

11.4.3.6 I2C Slave Transmitter/Receiver

The operating principle of I2C Slave and Master is similar, the difference is that the Master mode can control the transmission time, But slave mode needs to be ready at any time.

Whether to use the stretch function(I2CCTRL3[0] and I2C_IRQ_STATUS[0]) in the I2C slave mode can be used in two ways:

- If stretch function is used (the master must support stretch function), the slave can determine the R/W bit of the master and corresponding operations.
- If stretch function is not used, the slave needs to fill data or configure DMA in advance when sending data and configure DMA in advance when receiving data.

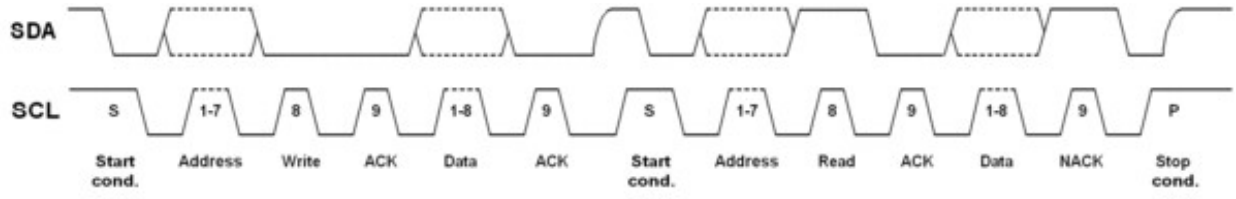
11.4.3.7 General Call Address

The General Call Address is a special address to address all slave devices on the I2C-bus. The controller at the slave mode responds with an ACK to the general call address and set the ID field of the I2C_ID Register.

11.4.3.8 I2C Master Restart

The I2C master supports the restart function. After data transmission is complete, start can be sent instead of stop for the next data transmission.

Figure 11-14 I2C Master Restart Condition



11.4.3.9 Auto Clock Stretch

(1) Slave Stretch

Clock stretching pauses a transaction by holding the SCL Line LOW. The I2C can automatically pause bus transactions by stretching clocks on the I2C-bus when the software is not ready for the next byte of data or when the RX FIFO is full or the TX FIFO is empty. When configuring register I2CCTRL3[0], Auto Clock Stretch is supported at the slave mode.

(2) Master Stretch

When configuring register I2CCTRL2[1], The master transaction cannot continue until the line is released high again.

11.4.3.10 Auto-ACK

With Auto-ACK, the I2C automatically generates proper acknowledgements for each byte received. Every received byte is responded with an ACK, except for the last byte, which should be responded with a NAK according to the I2C-bus protocol. On the other hand, if the software needs to determine last byte acknowledgement status, Auto-NAK can be turned off by disabling the resister I2CCTRL3[2].

The interrupt I2C_IRQ_STATUS[1] and register I2CSCT2[4] are used for the master. When the master detects nak, it sends a stop condition to end the current data transmission.

In master mode and DMA is used to send data. The following processing needs to be done when NAK is detected:

Disable DMA and clear TX FIFO(I2C_IRQ_STATUS[3]).

11.4.4 I2C Register Description

The I2C related registers are listed as following, the base address of the following registers is 0x80140280.

Table 11-1 I2C Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	I2CSP	RW	I2C master clock speed: pclk*1000*1000/(4*I2C_CLK_SPEED)	0x1f
0x01	I2C_ID	RW	I2C id: [7:1] I2C slave address + [0] R/W flag bit	0x5c
0x02	I2CMST	Volatile /R	[0]: mst_busy, master busy (volatile) [1]: mst_scs_n, master packet busy (volatile) [2]: mst_ack_in master received status: 1 for nak; 0 for ack (volatile) [5:3]: mst_p master state of the base: 0-ms_id, 1-ms_addr, 2-ms_dataw, 3-ms_datar, 4-ms_start, 5-ms_stop, 6-ms_idle (R) [7:6]: ss slave state of the base:0-ss_id, 1-ss_dataw, 2-ss_datar (R)	0x30
0x03	I2CSCT0	RW	[0]: I2C mask_slave_wr [1]: I2C mask_master_nak [2]: rx interrupt enable [3]: tx interrupt enable [4]: mask_txdone [5]: mask_rxdone [6]: mask_rxend [7]: mask_txend	0x00

Address Offset	Name	Type	Description	Default Value
0x04	I2CSCT1	RW	[0]: launch ID cycle [1]: launch address cycle [2]: launch data write cycle [3]: launch data read cycle [4]: launch start cycle [5]: launch stop cycle [6]: enable read ID [7]: enable ACK in read command	0x00
0x05	I2CTRIG	RW	[3:0]: rx_irq_trig level [7:4]: tx_irq_trig level	0x44
0x06	I2CCTRL2	RW	[0]: I2C master enable [1]: r_clk_stretch_en, clk stretch enable: suspend transmission by pulling SCL down to low level, and continue transmission after SCL is released to high level [2]: manual_tx_stop_hit [3]: manual_rx_stop_hit [4]: nak_stop_en [5]: tx_stretch_sel [6]: mask_stretch [7]: r_stretch_pos_sel 1'b1: stretch in any negedge position; 1'b0: stretch in ack/nak	0x00

Address Offset	Name	Type	Description	Default Value
0x07	I2CCTRL3	RW	[0]: r_clk_stretch_sen, slave auto stretch clk enable [1]: r_id_nmatch_stop_en Slave ID does not mastch the stop ss enable, The ss is in the SS_ID and may be error trigger by data [2]: r_ms_nak_en, the last byte data read is automatically returned to nack [3]: manual_sda_delay, delay sda and oen before ack (ID, ADDRESS, DATAW) [4]: ndma_rxdone_en, NDMA mode: rxdone function switch; 1:enable,0:disable;dma mode must disable [5]: auto_rxclr_en, DMA and ndma mode: auto clr function switch; 1:enable,0:disable [6]: r_hs_mode, standard mode and system clock 48M,maintain ss_scl setup time Max [7]: r_fast_mode, fast mode: ss_scl setup time Min	0x30
0x08	I2C_DATA_BUF0	RW	Write/read buffer[7:0]	0x00
0x09	I2C_DATA_BUF1	RW	Write/read buffer[15:8]	0x00
0x0a	I2C_DATA_BUF2	RW	Write/read buffer[23:16]	0x00
0x0b	I2C_DATA_BUF3	RW	Write/read buffer[31:24]	0x00
0x0c	I2C_BUFCNT	Volatile	[3:0]: rx_buf_cnt [7:4]: tx_buf_cnt	0x00
0x0d	I2C_STATUS	Volatile	[2:0]: rbcnt [0] W1C, manual_stretch_clr(manual clear slave stretch); [1] W1C, slave manual stretch clk trig; [2] W1C, manual_clr_mst_ack [3]: i2c_irq_o [6:4]: wbcnt [7]: rxdone	0x00

Address Offset	Name	Type	Description	Default Value
0x0e	I2C_STATUS1	Volatile	[0]: ss_read, judge whether slave is to read or write [1]: ss_scl, slave stretch indicate [2]: tx_empty [3]: rx_full [6]: ss_scl_irq, W1C, manual_stretch_irq_clr(manual clear slave stretch irq)	0x06
0x0f	I2C_CLR	W1C	[0]: ss_rw_clr, manual clear slave rw irq [1]: ms_nak_clr, manual clear master nak_irq [2]: rx_clr, rx manual fifo_clear [3]: tx_clr, tx manual fifo_clear [4]: rxdone_irq_clr, rxdone irq clr [5]: txdone_clr, tx_en(tx_done manual clear) [6]: rx_end_clr, manual clear rxend [7]: tx_end_clr, manual clear txend	0x00
0x10	I2CLENL	RW	Config buffer send and receive byte number (low): default 1 byte	0x01
0x11	I2CLENM	RW	Config buffer send and receive byte number (middle): default 0 byte	0x00
0x12	I2CLENH	RW	Config buffer send and receive byte number (high): default 0 byte	0x00
0x13	I2C_CTRL1	RW	[0]: pem_event_en [1]: rsvd [2]: pem_event_sel [3]: rsvd [4]: mask_trx_start [5]: mask_trx_stop [6]: r_stretch_auto_clr_dis [7]: rsvd	0x00

Address Offset	Name	Type	Description	Default Value
0x14	I2C_MST_STATUS	R	[0]: ss_busy, i2c slave busy flag [1]: ss_id, i2c slave id flag [2]: ss_dataw, i2c slave wdata flag [3]: ss_datar, i2c slave rdata flag [4]: id_busy, nak function id flag [5]: addr_busy, nak function address flag [6]: dataw_busy, nak function wdata flag [7]: datar_busy, nak function rdata flag	0x00
0x15	I2C_CTRL2	RW	[5:0]: pem_event_en	0x00

11.5 Memory SPI

11.5.1 Memory SPI Diagram

NOTE:

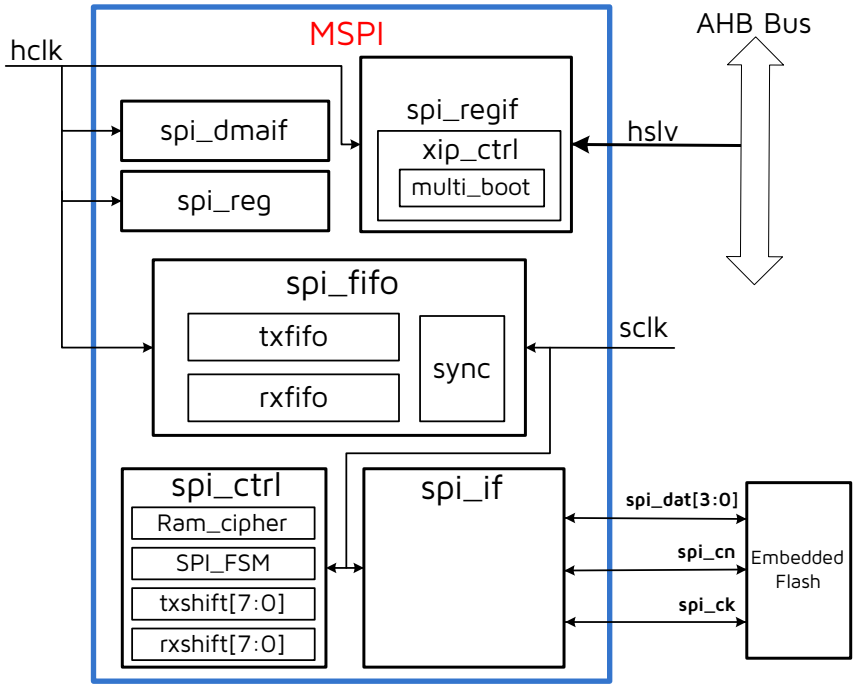
- The MSPI interface is used internally and is not available externally; the corresponding pins are not bonded out on the SoC.

Memory SPI module is a controller which serves as a SPI master to access SPI flash. Features of memory SPI are listed as following:

- Supports SPI master mode
- Supports Single line, Dual line, Quad line and 3 line I/O SPI interface
- Supports XIP function
- Supports DMA transmission
- Supports DMA burst transmission, tx_dma up to burst4, rx_dma up to burst2
- Supports HW Crypto engine

The Memory SPI diagram is shown as below:

Figure 11-15 Memory SPI Diagram



11.5.2 MSPI Register Description

Memory SPI related registers are listed as below. The base address of the following registers is 0xA3FFFF00.

Table 11-1 Memory SPI Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	MSPI_WR_RD_DATA0	RW	[7:0]: data[7:0] to transmit or received	0x00
0x01	MSPI_WR_RD_DATA1	RW	[7:0]: data[15:8] to transmit or received	0x00
0x02	MSPI_WR_RD_DATA2	RW	[7:0]: data[23:16] to transmit or received	0x00
0x03	MSPI_WR_RD_DATA3	RW	[7:0]: data[31:24] to transmit or received	0x00
0x04	MSPI_CMD	RW	[7:0]: SPI Command	0x00
0x05	MSPI_CTRL0	RW	[2]: RXFIFOIntEn, enable the SPI Receive FIFO Threshold interrupt [3]: TXFIFOIntEn, enable the SPI Transmit FIFO Threshold interrupt [4]: EndIntEn, enable the End of SPI Transfer interrupt [6]: rx_dma_en, RX DMA enable [7]: tx_dma_en, TX DMA enable	0x00

Address Offset	Name	Type	Description	Default Value
0x05	MSPI_REG_CMD1	RW	[7:0]: SPI Command1	0x00
0x07	MSPI_TIMING	RW	[2:0]: cs2sclk, the minimum time between the edge of SPI_CS and the edges of SPI_CLK. The actual duration is (SPI_CLK period*(cs2sclk+1)), MASTER ONLY [7:3]: csht_low, the minimum time that SPI CS should stay HIGH. The actual duration is (SPI_CLK period*(csht+1)), MASTER ONLY actual_csht[6:0] = {csht_high, csht_low}	0x09
0x08	MSPI_CTRL1	RW	[1:0]: reg_data_lane, reg data lane 0: single, 1: dual, 2: quad, 3: quad [3:2]: reg_addr_len 2'b00: 1bye 2'b01: 2bytes 2'b10: 3bytes 2'b11:4bytes, MASTER ONLY [4]: reg_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase(Dual/Quad), MASTER ONLY [5]: reg_addr_en, 1:enabel addr phase, MASTER ONLY [6]: reg_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase(Dual/Quad), MASTER ONLY [7]: reg_cmd_en, the spi commnd phase enable, MASTER ONLY	0xa9

Address Offset	Name	Type	Description	Default Value
0x09	MSPI_CTRL2	RW	[3:0]: dummy_cnt, dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: transmode, the transfer mode the transfer sequence could be: 0x0: write and read at the same time (must enable CmdEn) 0x1: write only 0x2: read only (must enable CmdEn) 0x3: write,read 0x4: read,write 0x5: write,dummy,read 0x6: read,dummy,write (must enable CmdEn) 0x7: None Data (must enable CmdEn) 0x8: Dummy,write 0x9: Dummy,read 0xa: Dummy, write and read 0xb~0xf: reserved	0x77
0x0a	MSPI_REG_CTRL0	RW	[0]: cmd1_en [1]: reg_token_val_sel 1: 8'h69 0: 8'h00 [2]: reg_token_en [3]: reg_ddr_mode [5:4]: csht_high	0x0
0x0b	MSPI_XIP_WR_TCEM_SET	RW	[7:0]: xip_wr_tcem_set actual = xip_wr_tcem_set * 4	0x0
0x0c	MSPI_ADDR0	RW	[7:0]: spi_addr0, spi address byte0, MASTER ONLY	0x0
0x0d	MSPI_ADDR1	RW	[7:0]: spi_addr1, spi address byte1, MASTER ONLY	0x0
0X0e	MSPI_ADDR2	RW	[7:0]: spi_addr2, spi address byte2, MASTER ONLY	0x0
0x0f	MSPI_ADDR3	RW	[7:0]: spi_addr3, spi address byte3, MASTER ONLY	0x0
0x10	MSPI_TX_CNT0	RW	[7:0]: tx_cnt0, transfer count for write data, byte0	0x00
0x11	MSPI_TX_CNT1	RW	[7:0]: tx_cnt1, transfer count for write data, byte1	0x00

Address Offset	Name	Type	Description	Default Value
0x12	MSPI_TX_CNT2	RW	[7:0]: tx_cnt2, transfer count for write data, byte2	0x00
0x14	MSPI_RX_CNT0	RW	[7:0]: rx_cnt0, transfer count for read data, byte0	0x00
0x15	MSPI_RX_CNT1	RW	[7:0]: rx_cnt1, transfer count for read data, byte1	0x00
0x16	MSPI_RX_CNT2	RW	[7:0]: rx_cnt2, transfer count for read data, byte2	0x00
0x18	MSPI_CTRL3	RW	[0]: spi_lsb, transfer data with least significant bit first [1]: spi_3line, MOSI is bi-directional signal in regular mode [3:2] spi_mode spi_mode[0]: SPI_CLK Phase; spi_mode[1]: SPI_CLK Polarity [4]: no_used [5]: dmatx_sof_clrtxfifo_en, auto clr txfifo when txdma start [6]: dmarx_eof_clrrxfifo_en, auto clr rxfifo when rxdma end [7]: auto_hready_en, auto control hready while access data register	0x90
0x19	MSPI_TXFIFO_THRES	RW	[5:0]: txfifo threshold	0x00
0x1a	MSPI_RXFIFO_THRES	RW	[5:0]: rxfifo threshold	0x00
0x1b	MSPI_PEM_CTRL0	RW	[0]: reg_event_en [1]: reg_task_en [2]: reg_pem_event_sel	0x00
0x1c	MSPI_CTRL4	RW	[0]: dma_trig_spi_en [1]: txdma_req_after_cmd [2]: xip_stop, stop xip [3]: xip_enable, enable xip [4]: dummy_cnt_add	0x0a
0x1d	MSPI_XIP_PAGE_SIZE	RW	[7:0]: page_size, page boundary size = 2^page_size	0x00
0x1e	MSPI_XIP_TIMEOUT_CNT	RW	[7:0]: timeout_cnt, timeout time sel	0x60
0x1f	MSPI_XIP_RD_TCEM_SET	RW	[7:0]: xip_rd_tcem_set actual = xip_rd_tcem_set * 4	

Address Offset	Name	Type	Description	Default Value
0x22	MSPI_XIP_ADDR_OFFSET	RW	[7:0]: xip_addr_offset, address offset = xip_addr_offset << 24	0x00
0x24	MSPI_TXFIFO_STATUS	R	[6:0]: txfifo_entries [7]: txfifo_full	0x0
0x25	MSPI_RXFIFO_STATUS	R	[6:0]: rxfifo_entries [7]: rxfifo_empty	0x0
0x28	MSPI_STATUS	RW/R	[2]: spi_soft_reset(RW), spi soft reset, high valid [3]: xip_reg_arb_err(R), xip mode and reg mode conflict flag [4]: rxfifo_clr_level(RW), rxfifo is in clear status [5]: txfifo_clr_level(RW), txfifo is in clear status [7]: busy(R), SPI is transferring	0x0
0x2a	MSPI_INT_STATUS0	W1C	[2]: rxf_thres_int_stus, RX FIFO Threshold interrupt [3]: txf_thres_int_stus, TX FIFO Threshold interrupt [4]: trans_end_int_stus, End of SPI Transfer interrupt	0x00
0x80	CIPHER_KEY_BYTE0	R	[7:0]: cipher_key_byte0, cipher_key[7:0]	0x00
0x81	CIPHER_KEY_BYTE1	R	[7:0]: cipher_key_byte2, cipher_key[15:8]	0x00
0x82	CIPHER_KEY_BYTE2	R	[7:0]: cipher_key_byte2, cipher_key[23:16]	0x00
0x83	CIPHER_KEY_BYTE3	R	[7:0]: cipher_key_byte3, cipher_key[31:24]	0x00
0x84	CIPHER_DATA_NONCE	R	[7:0]: cipher_data_nonce, cipher_data_nonce	0x00
0x85	CIPHER_CTRL	RW	[0]: cipher_rden, ram cipher read enable [1]: cipher_wren, ram cipher write enable	

Address Offset	Name	Type	Description	Default Value
0x90	MSPI_XIP_RD_FMT	RW	[1:0]: xip0_rd_data_lane, xip0 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip0_rd_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip0_rd_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad) [5] xip0_rd_addr_en, 1:enable addr phase, MASTER ONLY [6]: xip0_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip0_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0x91	MSPI_XIP_RD_TRANSMODE	RW	[3:0]: xip0_rd_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip0_rd_transmode	0x97
0x92	MSPI_XIP_RD_CTRL0	RW	[0]: cmd1_en [1]: xip_token_val_sel, 1:8'h69, 0:8'h00 [2]: xip_token_en [3]: xip0_rd_dummy_cnt_add [4]: xip0_ddr_mode [5]: xip0_page_mode_en, xip page mode enable [6]: xip0_timeout_mode_en 0: xip timeout disable 1: xip timeout enable [7]: xip0_tcem_mode_en 0: xip tcem disable 1: xip tcem enable	0x40
0x93	MSPI_XIP_RD_CMD	RW	[7:0]: xip0_rd_cmd, read command used for xip	0x3b

Address Offset	Name	Type	Description	Default Value
0x94	MSPI_XIP_WR_FMT	RW	[1:0]: xip0_wr_data_lane, xip0 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip0_wr_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip0_wr_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip0_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip0_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip0_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0x95	MSPI_XIP_WR_TRANS MODE	RW	[3:0]: xip0_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip0_wr_transmode	0x10
0x96	MSPI_XIP_WR_CTRL0	RW	[0]: cmd1_en	0x00
0x97	MSPI_XIP_WR_CMD	RW	[7:0]: xip0_wr_cmd, write command used for xip	0x02
0x98	MSPI_XIP1_RD_FMT	RW	[1:0]: xip0_rd_data_lane, xip0 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip0_rd_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip0_rd_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip0_rd_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip0_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip0_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0x99	MSPI_XIP1_RD_TRANS MODE	RW	[3:0]: xip0_rd_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip0_rd_transmode	0x97

Address Offset	Name	Type	Description	Default Value
0x9a	MSPI_XIP1_RD_CTRL0	RW	[0]: cmd1_en [1]: xip1_token_val_sel, 1:8'h69, 0:8'h00 [2]: xip1_token_en [3]: xip1_rd_dummy_cnt_add [4]: xip1_ddr_mode [5]: xip1_page_mode_en, xip page mode enable [6]: xip1_timeout_mode_en 0: xip timeout disable 1: xip timeout enable [7]: xip1_tcem_mode_en 0: xip tcem disable 1: xip tcem enable	0x40
0x9b	MSPI_XIP1_RD_CMD	RW	[7:0]: xip1_rd_cmd, read command used for xip	0x3b
0x9c	MSPI_XIP1_WR_FMT	RW	[1:0]: xip1_wr_data_lane, xip0 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip1_wr_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip1_wr_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip1_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip1_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip1_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0x9d	MSPI_XIP1_WR_TRANSMODE	RW	[3:0]: xip1_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip1_wr_transmode	0x10
0x9e	MSPI_XIP1_WR_CTRL0	RW	[0]: cmd1_en	0x00
0x9f	MSPI_XIP1_WR_CMD	RW	[7:0]: xip0_wr_cmd, write command used for xip	0x02

Address Offset	Name	Type	Description	Default Value
0xa0	MSPI_XIP2_RD_FMT	RW	[1:0]: xip2_rd_data_lane, xip0 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip2_rd_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip2_rd_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip2_rd_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip2_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip2_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0xa1	MSPI_XIP2_RD_TRANS MODE	RW	[3:0]: xip2_rd_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip2_rd_transmode	0x97
0xa2	MSPI_XIP2_RD_CTRL0	RW	[0]: cmd2_en [1]: xip2_token_val_sel, 1:8'h69, 0:8'h00 [2]: xip2_token_en [3]: xip2_rd_dummy_cnt_add [4]: xip2_ddr_mode [5]: xip2_page_mode_en, xip page mode enable [6]: xip2_timeout_mode_en 0: xip timeout disable 1: xip timeout enable [7]: xip2_tcem_mode_en 0: xip tcem disable 1: xip tcem enable	0x40
0xa3	MSPI_XIP2_RD_CMD	RW	[7:0]: xip2_rd_cmd, read command used for xip	0x3b

Address Offset	Name	Type	Description	Default Value
0xa4	MSPI_XIP2_WR_FMT	RW	[1:0]: xip2_wr_data_lane, xip0 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip2_wr_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip2_wr_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip2_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip2_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip2_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0xa5	MSPI_XIP2_WR_TRANSMODE	RW	[3:0]: xip2_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip2_wr_transmode	0x10
0xa6	MSPI_XIP2_WR_CTRL0	RW	[0]: cmd1_en	0x00
0xa7	MSPI_XIP2_WR_CMD	RW	[7:0]: xip2_wr_cmd, write command used for xip	0x02
0xa8	MSPI_XIP3_RD_FMT	RW	[1:0]: xip3_rd_data_lane, xip0 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip3_rd_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip3_rd_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip3_rd_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip3_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip3_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0xa9	MSPI_XIP3_RD_TRANSMODE	RW	[3:0]: xip3_rd_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip3_rd_transmode	0x97

Address Offset	Name	Type	Description	Default Value
0xaa	MSPI_XIP3_RD_CTRL0	RW	[0]: cmd3_en [1]: xip3_token_val_sel, 1:8'h69, 0:8'h00 [2]: xip3_token_en [3]: xip3_rd_dummy_cnt_add [4]: xip3_ddr_mode [5]: xip3_page_mode_en, xip page mode enable [6]: xip3_timeout_mode_en 0: xip timeout disable 1: xip timeout enable [7]: xip3_tcem_mode_en 0: xip tcem disable 1: xip tcem enable	0x40
0xab	MSPI_XIP3_RD_CMD	RW	[7:0]: xip3_rd_cmd, read command used for xip	0x3b
0xac	MSPI_XIP3_WR_FMT	RW	[1:0]: xip3_wr_data_lane, xip0 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2]: xip3_wr_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes [4]: xip3_wr_addr_fmt, 0:single mode, 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5]: xip3_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6]: xip3_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7]: xip3_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0xad	MSPI_XIP3_WR_TRANSMODE	RW	[3:0]: xip3_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4]: xip3_wr_transmode	0x10
0xae	MSPI_XIP3_WR_CTRL0	RW	[0]: cmd3_en	0x00
0xaf	MSPI_XIP3_WR_CMD	RW	[7:0]: xip3_wr_cmd, write command used for xip	0x02

Address Offset	Name	Type	Description	Default Value
0xb0	MSPI_XIP_SIZE_SET	RW	[1:0]: xip_psram0_end_addr psram0 space = {0, (xip_psram0_end_addr+1) * 16m} [3:2]: xip_psram1_end_addr psram1 space = {(xip_psram0_end_addr+1) * 16m, (xip_psram1_end_addr+1) * 16m} [5:4]: xip_psram2_end_addr psram2 space = {(xip_psram1_end_addr+1) * 16m, (xip_psram2_end_addr+1) * 16m} [7:6]: xip_psram3_end_addr psram3 space = {(xip_psram2_end_addr+1) * 16m, (xip_psram3_end_addr+1) * 16m}	0x03
0xb4	MSPI_XIP_RD_CMD1	RW	[7:0]: xip_rd_cmd1, SPI Command1	0x00
0xb5	MSPI_XIP_WR_CMD1	RW	[7:0]: xip_wr_cmd1, SPI Command1	0x00
0xb6	MSPI_XIP1_RD_CMD1	RW	[7:0]: xip1_rd_cmd1, SPI Command1	0x00
0xb7	MSPI_XIP1_WR_CMD1	RW	[7:0]: xip1_wr_cmd1, SPI Command1	0x00
0xb8	MSPI_XIP2_RD_CMD1	RW	[7:0]: xip2_rd_cmd1, SPI Command1	0x00
0xb9	MSPI_XIP2_WR_CMD1	RW	[7:0]: xip2_wr_cmd1, SPI Command1	0x00
0xba	MSPI_XIP3_RD_CMD1	RW	[7:0]: xip3_rd_cmd1, SPI Command1	0x00
0xbb	MSPI_XIP3_WR_CMD1	RW	[7:0]: xip3_wr_cmd1, SPI Command1	0x00
0xc0	MSPI_XIP_CORE0_START_L	RW	[7:0]: xip_core0_start[7:0]	0x00
0xc1	MSPI_XIP_CORE0_START_H	RW	[5:0]: xip_core0_start[13:8]	0x00
0xc2	MSPI_XIP_CORE0_SIZE_L	RW	[7:0]: xip_core0_size[7:0]	0x00
0xc3	MSPI_XIP_CORE0_SIZE_H	RW	[5:0]: xip_core0_size[13:8]	0x00
0xc4	MSPI_XIP_CORE0_OFFSET_L	RW	[7:0]: xip_core0_offset[7:0]	0x00
0xc5	MSPI_XIP_CORE0_OFFSET_H	RW	[5:0]: xip_core0_offset[13:8]	0x00

Address Offset	Name	Type	Description	Default Value
0xc8	MSPI_XIP_CORE1_START_L	RW	[7:0]: xip_core1_start[7:0]	0x00
0xc9	MSPI_XIP_CORE1_START_H	RW	[5:0]: xip_core1_start[13:8]	0x00
0xca	MSPI_XIP_CORE1_SIZE_L	RW	[7:0]: xip_core1_size[7:0]	0x00
0xcb	MSPI_XIP_CORE1_SIZE_H	RW	[5:0]: xip_core1_size[13:8]	0x00
0xcc	MSPI_XIP_CORE1_OFFSET_L	RW	[7:0]: xip_core1_offset[7:0]	0x00
0xcd	MSPI_XIP_CORE1_OFFSET_H	RW	[5:0]: xip_core1_offset[13:8]	0x00
0xd0	MSPI_XIP_CORE2_START_L	RW	[7:0]: xip_core2_start[7:0]	0x00
0xd1	MSPI_XIP_CORE2_START_H	RW	[5:0]: xip_core2_start[13:8]	0x00
0xd2	MSPI_XIP_CORE2_SIZE_L	RW	[7:0]: xip_core2_size[7:0]	0x00
0xd3	MSPI_XIP_CORE2_SIZE_H	RW	[5:0]: xip_core2_size[13:8]	0x00
0xd4	MSPI_XIP_CORE2_OFFSET_L	RW	[7:0]: xip_core2_offset[7:0]	0x00
0xd5	MSPI_XIP_CORE2_OFFSET_H	RW	[5:0]: xip_core2_offset[13:8]	0x00

11.6 LSPI

11.6.1 LSPI Diagram

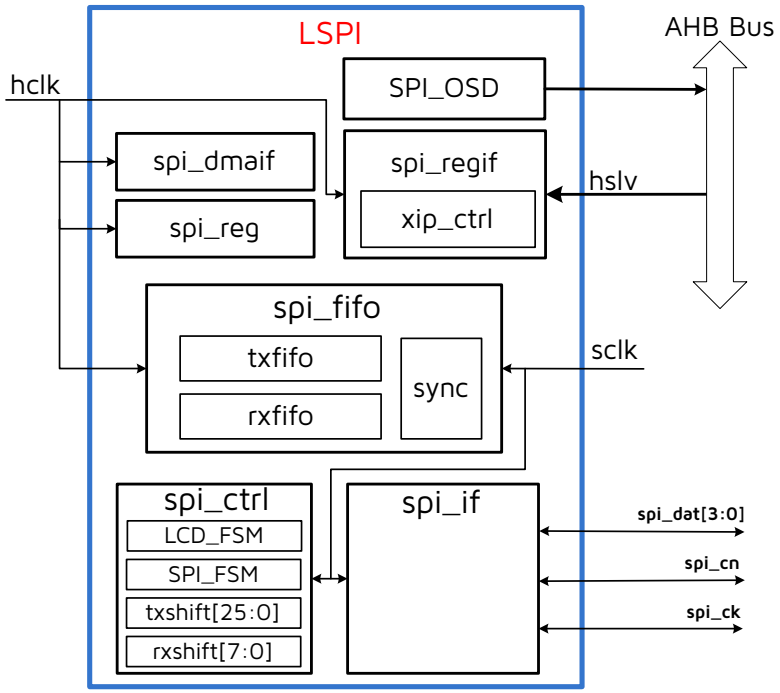
LSPI module is a controller which serves as a SPI master to access external LCD. Features of LSPI are listed as following:

- Supports SPI Master/Slave mode
- Supports Single line, Dual line, Quad line and 3 line I/O SPI interface
- Supports XIP function

- Supports DMA transmission
- Supports DMA burst transmission, tx_dma up to burst4, rx_dma up to burst2
- Supports LCD driving with SPI ports

The LSPI diagram is shown as below:

Figure 11-16 LSPI Diagram



11.6.2 LSPI Register Description

The LSPI related registers are listed as below. The base address of the following registers is 0x87FFFF00.

Table 11-1 LSPI Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	LSPI_WR_RD_DATA0	RW	[7:0]: data[7:0] to transmit or received	0x00
0x01	LSPI_WR_RD_DATA1	RW	[7:0]: data[15:8] to transmit or received	0x00
0x02	LSPI_WR_RD_DATA2	RW	[7:0]: data[23:16] to transmit or received	0x00
0x03	LSPI_WR_RD_DATA3	RW	[7:0]: data[31:24] to transmit or received	0x00
0x04	LSPI_CMD	RW	[7:0]: SPI Command	0x00

Address Offset	Name	Type	Description	Default Value
0x05	LSPI_CTRL0	RW	[0]: rxf_overrun_int_en, enable the SPI Receive FIFO Overrun interrupt, SLAVE ONLY [1]: txf_underrun_int_en, enable the SPI Transmit FIFO Underrun interrupt, SLAVE ONLY [2]: rxf_thres_int_en, enable the SPI Receive FIFO Threshold interrupt [3]: txf_thres_int_en, enable the SPI Transmit FIFO Threshold interrupt [4]: trans_end_int_en, enable the End of SPI Transfer interrupt [5]: slave_cmd_int_en, enable the Slave Command Interrupt, SLAVE ONLY [6]: rx_dma_en, RX DMA enable [7]: tx_dma_en, TX DMA enable	0x00
0x06	LSPI_REG_CMD1	RW	[7:0]: SPI Command1	0x00
0x07	LSPI_TIMING	RW	[2:0] cs2sclk, the minimum time between the edge of SPI_CS and the edges of SPI_CLK. The actual duration is (SPI_CLK period*(cs2sclk+1)), MASTER ONLY [7:3] csht, the minimum time that SPI CS should stay HIGH. The actual duration is (SPI_CLK period*(csht+1)), MASTER ONLY actual_csht[6:0] = {csht_high, csht_low}	0x09
0x08	LSPI_CTRL1	RW	[1:0] reg_data_lane, reg data lane 0: single, 1: dual, 2: quad, 3: quad [3:2] reg_addr_len 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes, MASTER ONLY [4] reg_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase(Dual/Quad), MASTER ONLY [5] reg_addr_en, 1:enabel addr phase, MASTER ONLY [6] reg_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase(Dual/Quad), MASTER ONLY [7] reg_cmd_en, the spi commnd phase enable, MASTER ONLY	0xa9

Address Offset	Name	Type	Description	Default Value
0x09	LSPI_CTRL2	RW	[3:0] dummy_cnt, dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4] transmode, the transfer mode the transfer sequence could be: 0x0:write and read at the same time (must enable CmdEn) 0x1:write only 0x2:read only (must enable CmdEn) 0x3:write,read 0x4:read,write 0x5:write,dummy,read 0x6:read,dummy,write (must enable CmdEn) 0x7:None Data (must enable CmdEn) 0x8:Dummy,write 0x9:Dummy,read 0xa:Dummy, write and read 0xb~0xf:reserved	0x77
0x0a	LSPI_REG_CTRL0	RW	[0] cmd1_en [1] reg_token_val_sel, 1:8'h69, 0:8'h00 [2] reg_token_en [3] reg_ddr_mode [5:4] csht_high	0x0
0x0b	LSPI_XIP_WR_TCEM_SET	RW	[7:0] xip_wr_tcem_set, actual = xip_wr_tcem_set * 4	0x0
0x0c	LSPI_ADDR0	RW	[7:0] spi_addr0, spi address byte0/lcd_porch_line_time[7:0]	0x0
0x0d	LSPI_ADDR1	RW	[7:0] spi_addr1, spi address byte1/lcd_porch_line_time[15:8]	0x0
0x0e	LSPI_ADDR2	RW	[7:0] spi_addr2, spi address byte2/ lcd_display_line_time[7:0]	0x0
0x0f	LSPI_ADDR3	RW	[7:0] spi_addr3, spi address byte3/ lcd_display_line_time[15:8]	0x0
0x10	LSPI_TX_CNT0	RW	[7:0] tx_cnt0, transfer count for write data, byte0/ lcd_pixel_per_line[7:0]	0x00

Address Offset	Name	Type	Description	Default Value
0x11	LSPI_TX_CNT1	RW	[7:0] tx_cnt1, transfer count for write data, byte1/ {lcd_line_per_frame[5:0], lcd_pixel_per_line[9:8]}	0x00
0x12	LSPI_TX_CNT2	RW	[7:0] tx_cnt2, transfer count for write data, byte2/{4'h0, lcd_line_per_frame[9:6]}	0x00
0x14	LSPI_RX_CNT0	RW	[7:0] rx_cnt0, transfer count for read data, byte0	0x00
0x15	LSPI_RX_CNT1	RW	[7:0] rx_cnt1, transfer count for read data, byte1	0x00
0x16	LSPI_RX_CNT2	RW	[7:0] rx_cnt2, transfer count for read data, byte2	0x00
0x18	LSPI_CTRL3	RW	[0] spi_lsb, transfer data with least significant bit first [1] spi_3line, MOSI is bi-directional signal in regular mode [3:2] spi_mode, spi_mode[0]:SPI_CLK Phase; spi_mode[1]:SPI_CLK Polarity [4] spi_master, SPI master mode selection [5] dmatx_sof_clrtxfifo_en, auto clr txfifo when txdma start [6] dmarx_eof_clrrxfifo_en, auto clr rxfifo when rxdma end [7] auto_hready_en, auto control hready while access data register	0x90
0x19	LSPI_TXFIFO_THRES	RW	[5:0] txfifo threshold	0x00
0x1a	LSPI_RXFIFO_THRES	RW	[5:0] rxfifo threshold	0x00
0x1b	LSPI_PEM_CTRL0	RW	[0] reg_event_en [1] reg_task_en [2] reg_pem_event_sel	0x00
0x1c	LSPI_CTRL4	RW	[0] dma_trig_spi_en [1] txdma_req_after_cmd [2] xip_stop, stop xip [3] xip_enable, enable xip [4] dummy_cnt_add	0x0a
0x1d	LSPI_XIP_PAGE_SIZE	RW	[7:0] page_size, page boundary size = 2^page_size	0x00
0x1e	LSPI_XIP_TIMEOUT_CNT	RW	[7:0] timeout_cnt, timeout time sel	0x60
0x1f	LSPI_XIP_RD_TCEM_SET	RW	[7:0] xip_rd_tcem_set, actual = xip_rd_tcem_set * 4	0x00

Address Offset	Name	Type	Description	Default Value
0x22	LSPI_XIP_ADDR_OFFSET	RW	[7:0] xip_addr_offset, address offset = xip_addr_offset << 24	0x00
0x24	LSPI_TXFIFO_STATUS	R	[6:0] txfifo_entries [7] txfifo_full	0x00
0x25	LSPI_RXFIFO_STATUS	R	[6:0] rxfifo_entries [7] rxfifo_empty	0x00
0x28	LSPI_STATUS	RW/R	[0] set_slave_ready(RW), set this bit to indicate that spi as salve is ready for data transaction [1] clr_slave_ready(RW), clear spi slave ready [2] spi_soft_reset(RW), spi soft reset, high valid [3] xip_reg_arb_err(R), xip mode and reg mode conflict flag [4] rxfifo_clr_level(RW), rxfifo is in clear status [5] txfifo_clr_level(RW), txfifo is in clear status [6] osd_ahbmst_busy(R), osd ahbmster is in busy status [7] busy(R), SPI is transferring	0x00
0x29	LSPI_SLV_TRANS_MODE	RW	[3:0] slv_trans_mode	0x07
0x2a	LSPI_INT_STATUS0	W1C	[0] rxf_overrun_int_stus, RX FIFO Overrun interrupt, SLAVE ONLY [1] txf_underrun_int_stus, TX FIFO Underrun interrupr, SLAVE ONLY [2] rxf_thres_int_stus, RX FIFO Threshold interrupt [3] txf_thres_int_stus, TX FIFO Threshold interrupt [4] trans_end_int_stus, End of SPI Transfer interrupt [5] slave_cmd_int_stus, Slave Command Interrupt, SLAVE ONLY	0x00
0x2b	LSPI_INT_STATUS1	W1C	[0] lcd_line_int_stus, lcd line interrupt status [1] lcd_lvl_int_stus, lcd line level interrupt status [2] lcd_frame_int_stus, lcd frame interrupt status	0x00

Address Offset	Name	Type	Description	Default Value
0x2f	LSPI_LCD_CTRL2	RW	[0] lcd_single_color_mode, 1: single color mode, use lut1 [1] lcd_rgb_big_endian_mode, 1:big endian mode; 0:little endian mode [2] lcd_ram_4bit_mode [5:3] lcd_int_mask, lcd interrupt mask, [3]: lcd_line_irq_mask [4]: lcd_line_lvl_irq_mask [5]: lcd_frame_irq_mask [6] lcd_off_bimage [7] lcd_off_fimage	0x00
0x30	LSPI_LCD_CTRL	RW	[0] lcd_scan_en, scan lcd enable [2:1] lcd_rgb_mode, 0:rsvd; 1:565; 2:666; 3:888 [3] lcd_2lane_en, 1: 2 data lane enable in ram lcd mode [6] line3_dcx_en, 1:enable 3line mode [7] dcx, 1:set dcx filed to 1	0x00
0x31	LSPI_LCD_VBP_CNT	RW	[7:0] lcd_vbp_cnt, lcd vertical porch line number, actual num=lcd_vbp_cnt	0x00
0x32	LSPI_LCD_VFP_CNT	RW	[7:0] lcd_vfp_cnt, lcd front porch line number, actual num=lcd_vbp_cnt	0x00
0x33	LSPI_LCD_LINE_LVL	RW	[7:0] lcd_line_lvl, lcd line threshold to trig interrupt	0x00
0x34	LSPI_LCD_BIMAGE_ADDR0	RW	[7:5] lcd_bimage_start_addr0, background image data start address byte0	0x00
0x35	LSPI_LCD_BIMAGE_ADDR1	RW	[7:0] lcd_bimage_start_addr1, background image data start address byte1	0x00
0x36	LSPI_LCD_BIMAGE_ADDR2	RW	[7:0] lcd_bimage_start_addr2, background image data start address byte2	0x00
0x37	LSPI_LCD_BIMAGE_ADDR3	RW	[6:0] lcd_bimage_start_addr3, background image data start address byte3	0x00
0x38	LSPI_LCD_FIMAGE_ADDR0	RW	[7:4] lcd_fimage_start_addr0, front image data start address byte0	0x00
0x39	LSPI_LCD_FIMAGE_ADDR1	RW	[7:0] lcd_fimage_start_addr1, front image data start address byte1	0x00

Address Offset	Name	Type	Description	Default Value
0x3a	LSPI_LCD_FIMAGE_ADDR2	RW	[7:0] lcd_fimage_start_addr2, front image data start address byte2	0x00
0x3b	LSPI_LCD_FIMAGE_ADDR3	RW	[6:0] lcd_fimage_start_addr3, front image data start address byte3	0x00
0x3e	LSPI_LCD_LINE_CNT0	R	[7:0] lcd_line_cnt_l, lcd_line_cnt[7:0]	0x00
0x3f	LSPI_LCD_LINE_CNT1	R	[1:0] lcd_line_cnt_l, lcd_line_cnt[9:8]	0x00
0x40	LSPI_LCD_LUT_DATA0_BYTE0	RW	[7:0] lcd_lut_data0_byte0, lcd lut address0 data byte0	0x00
0x41	LSPI_LCD_LUT_DATA0_BYTE1	RW	[7:0] lcd_lut_data0_byte1, lcd lut address0 data byte1	0x00
0x42	LSPI_LCD_LUT_DATA0_BYTE2	RW	[7:0] lcd_lut_data0_byte2, lcd lut address0 data byte2	0x00
0x44	LSPI_LCD_LUT_DATA1_BYTE0	RW	[7:0] lcd_lut_data1_byte0, lcd lut address1 data byte0	0x00
0x45	LSPI_LCD_LUT_DATA1_BYTE1	RW	[7:0] lcd_lut_data1_byte1, lcd lut address1 data byte1	0x00
0x46	LSPI_LCD_LUT_DATA1_BYTE2	RW	[7:0] lcd_lut_data1_byte2, lcd lut address1 data byte2	0x00
0x48	LSPI_LCD_LUT_DATA2_BYTE0	RW	[7:0] lcd_lut_data2_byte0, lcd lut address2 data byte0	0x00
0x49	LSPI_LCD_LUT_DATA2_BYTE1	RW	[7:0] lcd_lut_data2_byte1, lcd lut address2 data byte1	0x00
0x4a	LSPI_LCD_LUT_DATA2_BYTE2	RW	[7:0] lcd_lut_data2_byte2, lcd lut address2 data byte2	0x00
0x4c	LSPI_LCD_LUT_DATA3_BYTE0	RW	[7:0] lcd_lut_data3_byte0, lcd lut address3 data byte0	0x00
0x4d	LSPI_LCD_LUT_DATA3_BYTE1	RW	[7:0] lcd_lut_data3_byte1, lcd lut address3 data byte1	0x00
0x4e	LSPI_LCD_LUT_DATA3_BYTE2	RW	[7:0] lcd_lut_data3_byte2, lcd lut address3 data byte2	0x00

Address Offset	Name	Type	Description	Default Value
0x50	LSPI_LCD_LUT_DATA 4_BYTE0	RW	[7:0] lcd_lut_data4_byte0, lcd lut address4 data byte0	0x00
0x51	LSPI_LCD_LUT_DATA 4_BYTE1	RW	[7:0] lcd_lut_data4_byte1, lcd lut address4 data byte1	0x00
0x52	LSPI_LCD_LUT_DATA 4_BYTE2	RW	[7:0] lcd_lut_data4_byte2, lcd lut address4 data byte2	0x00
0x54	LSPI_LCD_LUT_DATA 5_BYTE0	RW	[7:0] lcd_lut_data5_byte0, lcd lut address5 data byte0	0x00
0x55	LSPI_LCD_LUT_DATA 5_BYTE1	RW	[7:0] lcd_lut_data5_byte1, lcd lut address5 data byte1	0x00
0x56	LSPI_LCD_LUT_DATA 5_BYTE2	RW	[7:0] lcd_lut_data5_byte2, lcd lut address5 data byte2	0x00
0x58	LSPI_LCD_LUT_DATA 6_BYTE0	RW	[7:0] lcd_lut_data6_byte0, lcd lut address6 data byte0	0x00
0x59	LSPI_LCD_LUT_DATA 6_BYTE1	RW	[7:0] lcd_lut_data6_byte1, lcd lut address6 data byte1	0x00
0x5a	LSPI_LCD_LUT_DATA 6_BYTE2	RW	[7:0] lcd_lut_data6_byte2, lcd lut address6 data byte2	0x00
0x5c	LSPI_LCD_LUT_DATA 7_BYTE0	RW	[7:0] lcd_lut_data7_byte0, lcd lut address7 data byte0	0x00
0x5d	LSPI_LCD_LUT_DATA 7_BYTE1	RW	[7:0] lcd_lut_data7_byte1, lcd lut address7 data byte1	0x00
0x5e	LSPI_LCD_LUT_DATA 7_BYTE2	RW	[7:0] lcd_lut_data7_byte2, lcd lut address7 data byte2	0x00
0x60	LSPI_LCD_LUT_DATA 8_BYTE0	RW	[7:0] lcd_lut_data8_byte0, lcd lut address8 data byte0	0x00
0x61	LSPI_LCD_LUT_DATA 8_BYTE1	RW	[7:0] lcd_lut_data8_byte1, lcd lut address8 data byte1	0x00
0x62	LSPI_LCD_LUT_DATA 8_BYTE2	RW	[7:0] lcd_lut_data8_byte2, lcd lut address8 data byte2	0x00
0x64	LSPI_LCD_LUT_DATA 9_BYTE0	RW	[7:0] lcd_lut_data9_byte0, lcd lut address9 data byte0	0x00

Address Offset	Name	Type	Description	Default Value
0x65	LSPI_LCD_LUT_DATA 9_BYTE1	RW	[7:0] lcd_lut_data9_byte1, lcd lut address9 data byte1	0x00
0x66	LSPI_LCD_LUT_DATA 9_BYTE2	RW	[7:0] lcd_lut_data9_byte2, lcd lut address9 data byte2	0x00
0x68	LSPI_LCD_LUT_DATA 10_BYTE0	RW	[7:0] lcd_lut_data10_byte0, lcd lut address10 data byte0	0x00
0x69	LSPI_LCD_LUT_DATA 10_BYTE1	RW	[7:0] lcd_lut_data10_byte1, lcd lut address10 data byte1	0x00
0x6a	LSPI_LCD_LUT_DATA 10_BYTE2	RW	[7:0] lcd_lut_data10_byte2, lcd lut address10 data byte2	0x00
0x6c	LSPI_LCD_LUT_DATA 11_BYTE0	RW	[7:0] lcd_lut_data11_byte0, lcd lut address11 data byte0	0x00
0x6d	LSPI_LCD_LUT_DATA 11_BYTE1	RW	[7:0] lcd_lut_data11_byte1, lcd lut address11 data byte1	0x00
0x6e	LSPI_LCD_LUT_DATA 11_BYTE2	RW	[7:0] lcd_lut_data11_byte2, lcd lut address11 data byte2	0x00
0x70	LSPI_LCD_LUT_DATA 12_BYTE0	RW	[7:0] lcd_lut_data12_byte0, lcd lut address12 data byte0	0x00
0x71	LSPI_LCD_LUT_DATA 12_BYTE1	RW	[7:0] lcd_lut_data12_byte1, lcd lut address12 data byte1	0x00
0x72	LSPI_LCD_LUT_DATA 12_BYTE2	RW	[7:0] lcd_lut_data12_byte2, lcd lut address12 data byte2	0x00
0x74	LSPI_LCD_LUT_DATA 13_BYTE0	RW	[7:0] lcd_lut_data13_byte0, lcd lut address13 data byte0	0x00
0x75	LSPI_LCD_LUT_DATA 13_BYTE1	RW	[7:0] lcd_lut_data13_byte1, lcd lut address13 data byte1	0x00
0x76	LSPI_LCD_LUT_DATA 13_BYTE2	RW	[7:0] lcd_lut_data13_byte2, lcd lut address13 data byte2	0x00
0x78	LSPI_LCD_LUT_DATA 14_BYTE0	RW	[7:0] lcd_lut_data14_byte0, lcd lut address14 data byte0	0x00
0x79	LSPI_LCD_LUT_DATA 14_BYTE1	RW	[7:0] lcd_lut_data14_byte1, lcd lut address14 data byte1	0x00

Address Offset	Name	Type	Description	Default Value
0x7a	LSPI_LCD_LUT_DATA 14_BYTE2	RW	[7:0] lcd_lut_data14_byte2, lcd lut address14 data byte2	0x00
0x7c	LSPI_LCD_LUT_DATA 15_BYTE0	RW	[7:0] lcd_lut_data15_byte0, lcd lut address15 data byte0	0x00
0x7d	LSPI_LCD_LUT_DATA 15_BYTE1	RW	[7:0] lcd_lut_data15_byte1, lcd lut address15 data byte1	0x00
0x7e	LSPI_LCD_LUT_DATA 15_BYTE2	RW	[7:0] lcd_lut_data15_byte2, lcd lut address15 data byte2	0x00
0x90	LSPI_XIP_RD_FMT	RW	[1:0] xip0_rd_data_lane, xip0 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip0_rd_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip0_rd_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad) [5] xip0_rd_addr_en, 1:enable addr phase, MASTER ONLY [6] xip0_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7] xip0_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0x91	LSPI_XIP_RD_TRANS MODE	RW	[3:0] xip0_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip0_rd_transmode, xip read transmode/lcd display transmode	0x97
0x92	LSPI_XIP_RD_CTRL0	RW	[0] cmd1_en [1] xip_token_val_sel, 1:8'h69, 0:8'h00 [2] xip_token_en [3] xip0_rd_dummy_cnt_add [4] xip0_ddr_mode [5] xip0_page_mode_en [6] xip0_timeout_mode_en, 0:xip timeout disable 1:xip timeout enable [7] xip0_tcem_mode_en, 0:xip tcem disable 1:xip tcem enable	0x40

Address Offset	Name	Type	Description	Default Value
0x93	LSPI_XIP_RD_CMD	RW	[7:0] xip0_rd_cmd, read command used for xip	0x3b
0x94	LSPI_XIP_WR_FMT	RW	[1:0] xip0_wr_data_lane, xip0 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip0_wr_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip0_wr_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5] xip0_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6] xip0_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7] xip0_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0x95	LSPI_XIP_WR_TRANSMODE	RW	[7:4] xip0_wr_transmode, xip write transmode/lcd porch transmode	0x10
0x96	LSPI_XIP_WR_CTRL0	RW	[0] cmd1_en	0x00
0x97	LSPI_XIP_WR_CMD	RW	[7:0] xip0_wr_cmd, write command used for xip/lcd_cmd	0x02
0xb4	LSPI_XIP_RD_CMD1	RW	[7:0] xip_rd_cmd1, SPI Command1	0x00
0xb5	LSPI_XIP_WR_CMD1	RW	[7:0] xip0_wr_cmd, SPI Command1	0x00

11.7 GSPI

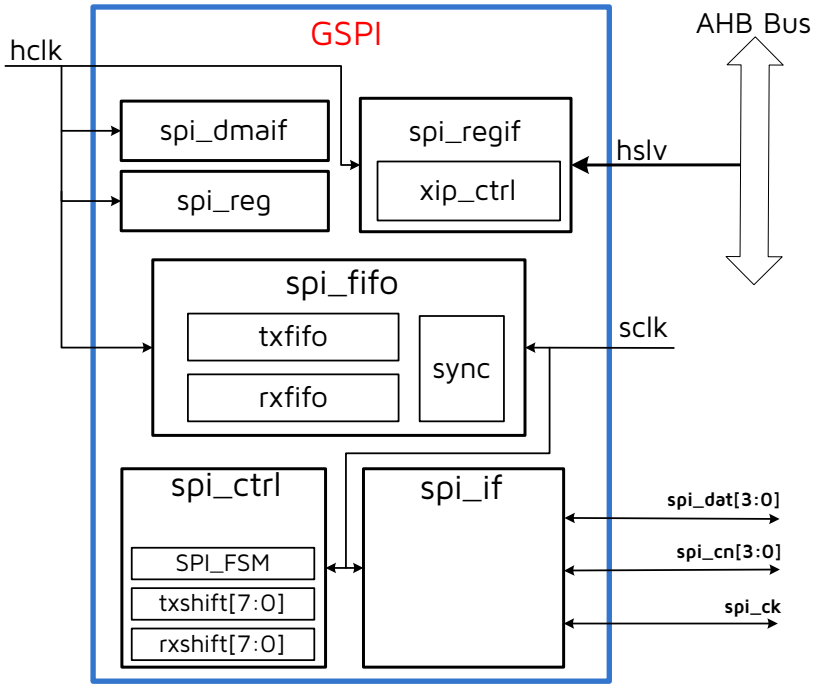
11.7.1 GSPI Diagram

The GSPI module is a controller which serves as a SPI master to access other general SPI interfaces. Features of GSPI are listed as following:

- Supports SPI Master/Slave mode
- Supports Single line, Dual line, Quad line and 3 line I/O SPI interface
- Supports XIP function
- Supports DMA transmission
- Supports multi-chip selection function

The GSPI diagram is shown as following:

Figure 11-17 GSPI Diagram



11.7.2 GSPI Register Description

The GSPI related registers are listed as below. The base address of the following registers is 0x8BFFFF00.

Table 11-1 GSPI Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	GSPI_WR_RD_DATA0	RW	[7:0] wr_rd_data0, data[7:0] to transmit or received	0x00
0x01	GSPI_WR_RD_DATA1	RW	[7:0] wr_rd_data1, data[15:8] to transmit or received	0x00
0x02	GSPI_WR_RD_DATA2	RW	[7:0] wr_rd_data2, data[23:16] to transmit or received	0x00
0x03	GSPI_WR_RD_DATA3	RW	[7:0] wr_rd_data3, data[31:24] to transmit or received	0x00
0x04	GSPI_CMD	RW	[7:0]: SPI Command	0x00

Address Offset	Name	Type	Description	Default Value
0x05	GSPI_CTRL0	RW	[0] RXFIFOORIntEn, enable the SPI Receive FIFO Overrun interrupt, SLAVE ONLY [1] TXFIFOURIntEn, enable the SPI Transmit FIFO Underrun interrupt, SLAVE ONLY [2]: RXFIFOIntEn, enable the SPI Receive FIFO Threshold interrupt [3]: TXFIFOIntEn, enable the SPI Transmit FIFO Threshold interrupt [4]: EndIntEn, enable the End of SPI Transfer interrupt [6]: rx_dma_en, RX DMA enable [7]: tx_dma_en, TX DMA enable	0x00
0x06	GSPI_REG_CMD1	RW	[7:0] reg_cmd1, SPI Command1	
0x07	GSPI_TIMING	RW	[2:0] cs2sclk, the minimum time between the edge of SPI_CS and the edges of SPI_CLK. The actual duration is (SPI_CLK period*(cs2sclk+1)), MASTER ONLY [7:3] csht_low the minimum time that SPI CS should stay HIGH.the actual duration is (SPI_CLK period*(csht+1)),MASTER ONLY actual_csht[6:0] = {csht_high, csht_low}	0x09
0x08	GSPI_CTRL1	RW	[1:0] reg_data_lane, reg data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] reg_addr_len, 2'b00:1bye, 2'b01:2bytes, 2'b10:3bytes, 2'b11:4bytes, MASTER ONLY [4] reg_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase(Dual/Quad), MASTER ONLY [5] reg_addr_en, 1:enabel addr phase, MASTER ONLY [6] reg_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase(Dual/Quad), MASTER ONLY [7] reg_cmd_en, the spi commnd phase enable, MASTER ONLY	0xa9

Address Offset	Name	Type	Description	Default Value
0x09	GSPI_CTRL2	RW	[3:0] dummy_cnt, dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4] transmode, the transfer mode the transfer sequence could be: 0x0:write and read at the same time (must enable CmdEn) 0x1:write only 0x2:read only (must enable CmdEn) 0x3:write,read 0x4:read,write 0x5:write,dummy,read 0x6:read,dummy,write (must enable CmdEn) 0x7:None Data (must enable CmdEn) 0x8:Dummy,write 0x9:Dummy,read 0xa:Dummy, write and read 0xb~0xf:reserved	0x77
0x0a	GSPI_REG_CTRL0	RW	[0] cmd1_en_ [1] reg_token_val_sel, 1:8'h69, 0:8'h00 [2] reg_token_en [3] reg_ddr_mode_ [5:4] csht_high	0x00
0x0b	GSPI_XIP_WR_TCEM_SET	RW	[7:0] xip_wr_tcem_set, actual = xip_wr_tcem_set * 4	
0x0c	GSPI_ADDR0	RW	[7:0] spi_addr0, spi address byte0, MASTER ONLY	0x00
0x0d	GSPI_ADDR1	RW	[7:0] spi_addr1, spi address byte1, MASTER ONLY	0x00
0x0e	GSPI_ADDR2	RW	[7:0] spi_addr2, spi address byte2, MASTER ONLY	0x00
0x0f	GSPI_ADDR3	RW	[7:0] spi_addr3, spi address byte3, MASTER ONLY	0x00
0x10	GSPI_TX_CNT0	RW	[7:0] tx_cnt0, transfer count for write data, byte0	0x00
0x11	GSPI_TX_CNT1	RW	[7:0] tx_cnt1, transfer count for write data, byte1	0x00
0x12	GSPI_TX_CNT2	RW	[7:0] tx_cnt2, transfer count for write data, byte2	0x00
0x14	GSPI_RX_CNT0	RW	[7:0] rx_cnt0, transfer count for read data, byte0	0x00

Address Offset	Name	Type	Description	Default Value
0x15	GSPI_RX_CNT1	RW	[7:0] rx_cnt1, transfer count for read data, byte1	0x00
0x16	GSPI_RX_CNT2	RW	[7:0] rx_cnt2, transfer count for read data, byte2	0x00
0x18	GSPI_CTRL3	RW	[0] spi_lsb, transfer data with least significant bit first [1] spi_3line, MOSI is bi-directional signal in regular mode [3:2] spi_mode, spi_mode[0]:SPI_CLK Phase; spi_mode[1]:SPI_CLK Polarity [4] spi_master, SPI master mode selection [5] dmatx_sof_clrtxfifo_en, auto clr txfifo when txdma start [6] dmarx_eof_clrrxfifo_en, auto clr rxfifo when rxdma end [7] auto_hready_en, auto control hready while access data register	0x90
0x19	GSPI_TXFIFO_THRES	RW	[5:0] txfifo threshold	0x00
0x1a	GSPI_RXFIFO_THRES	RW	[5:0] rxfifo threshold	0x00
0x1b	GSPI_PEM_CTRL0	RW	[0] reg_event_en [1] reg_task_en [2] reg_pem_event_sel	0x00
0x1c	GSPI_CTRL4	RW	[0] dma_trig_spi_en [1] txdma_req_after_cmd [2] xip_stop, stop xip [3] xip_enable, enable xip [4] dummy_cnt_add	0x0a
0x1d	GSPI_XIP_PAGE_SIZE	RW	[7:0] page_size, page boundary size = 2^page_size	0x00
0x1e	GSPI_XIP_TIMEOUT_CNT	RW	[7:0] timeout_cnt, timeout time sel	0x60
0x1f	GSPI_XIP_RD_TCEM_SET	RW	[7:0] xip_rd_tcem_set	0x00
0x22	GSPI_XIP_ADDR_OFFSET	RW	[7:0] xip_addr_offset, address offset = xip_addr_offset << 24	0x00
0x24	GSPI_TXFIFO_STATUS	R	[6:0] txfifo_entries [7] txfifo_full	0x0

Address Offset	Name	Type	Description	Default Value
0x25	GSPI_RXFIFO_STATUS	R	[6:0] rxfifo_entries [7] rxfifo_empty	0x0
0x28	GSPI_STATUS	RW/R	[0] set_slave_ready(RW), set this bit to indicate that spi as salve is ready for data transaction [1] clr_slave_ready(RW), clear spi slave ready [2] spi_soft_reset(RW), spi soft reset, high valid [3] xip_reg_arb_err(R), xip mode and reg mode conflict flag [4] rxfifo_clr_level(RW), rxfifo is in clear status [5] txfifo_clr_level(RW), txfifo is in clear status [7] busy(R), SPI is transferring	0x0
0x29	GSPI_SLV_TRANS_MODE	RW	[3:0] slv_trans_mode	
0x2a	GSPI_INT_STATUS0	W1C	[0] rxf_overrun_int_stus, RX FIFO Overrun interrupt, SLAVE ONLY [1] txf_underrun_int_stus, TX FIFO Underrun interrupr,SLAVE ONLY [2] rxf_thres_int_stus, RX FIFO Threshold interrupt [3] txf_thres_int_stus, TX FIFO Threshold interrupt [4] trans_end_int_stus, End of SPI Transfer interrupt [5] slave_cmd_int_stus, Slave Command Interrupt, SLAVE ONLY	0x00

Address Offset	Name	Type	Description	Default Value
0x90	GSPI_XIP_RD_FMT	RW	[0] xip0_rd_data_dual, spi dual I/O mode, MASTER ONLY [1] xip0_rd_data_quad, spi quad I/O mode, MASTER ONLY [3:2] xip0_rd_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip0_rd_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad) [5] xip0_rd_addr_en, 1:enable addr phase, MASTER ONLY [6] xip0_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/ Quad), MASTER ONLY [7] xip0_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0x91	GSPI_XIP_RD_TRANSMODE	RW	[3:0] xip0_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip0_rd_transmode	0x97
0x92	GSPI_XIP_RD_CTRL0	RW	[0] cmd1_en [1] xip_token_val_sel, 1:8'h69, 0:8'h00 [2] xip_token_en [3] xip0_rd_dummy_cnt_add [4] xip0_ddr_mode [5] xip0_page_mode_en [6] xip0_timeout_mode_en 0:xip timeout disable 1:xip timeout enable [7] xip0_tcem_mode_en 0:xip tcem disable 1:xip tcem enable	0x40
0x93	GSPI_XIP_RD_CMD	RW	[7:0] xip0_rd_cmd, read command used for xip	0x3b

Address Offset	Name	Type	Description	Default Value
0x94	GSPI_XIP_WR_FMT	RW	[1:0] xip0_wr_data_lane, xip0 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip0_wr_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip0_wr_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5] xip0_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6] xip0_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/ Quad), MASTER ONLY [7] xip0_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0x95	GSPI_XIP_WR_TRANSMODE	RW	[3:0] xip0_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4] xip0_wr_transmode	0x10
0x96	GSPI_XIP_WR_CTRL0	RW	[7:0] cmd1_en	
0x97	GSPI_XIP_WR_CMD	RW	[7:0] xip0_wr_cmd, write command used for xip	0x02
0x98	GSPI_XIP1_RD_FMT	RW	[1:0] xip1_rd_data_lane, xip1 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip1_rd_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip1_rd_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad) [5] xip1_rd_addr_en, 1:enable addr phase, MASTER ONLY [6] xip1_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7] xip1_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0x99	GSPI_XIP1_RD_TRANSMODE	RW	[3:0] xip1_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip1_rd_transmode	0x97

Address Offset	Name	Type	Description	Default Value
0x9a	GSPI_XIP1_RD_CTRL0	RW	[0] cmd1_en [1] xip1_token_val_sel, 1:8'h69, 0:8'h00 [2] xip1_token_en [3] xip1_rd_dummy_cnt_add [4] xip1_ddr_mode [5] xip1_page_mode_en [6] xip1_timeout_mode_en 0:xip timeout disable 1:xip timeout enable [7] xip1_tcem_mode_en 0:xip tcem disable 1:xip tcem enable	0x40
0x9b	GSPI_XIP1_RD_CMD	RW	[7:0] xip1_rd_cmd, read command used for xip	0x3b
0x9c	GSPI_XIP1_WR_FMT	RW	[1:0] xip1_wr_data_lane, xip1 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip1_wr_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip1_wr_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5] xip1_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6] xip1_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/ Quad), MASTER ONLY [7] xip1_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0x9d	GSPI_XIP1_WR_TRANSMODE	RW	[3:0] xip1_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4] xip1_wr_transmode	0x10
0x9e	GSPI_XIP1_WR_CTRL0	RW	[0] cmd1_en	0x00
0x9f	GSPI_XIP1_WR_CMD	RW	[7:0] xip1_wr_cmd, write command used for xip	0x02

Address Offset	Name	Type	Description	Default Value
0xa0	GSPI_XIP2_RD_FMT	RW	[1:0] xip2_rd_data_lane, xip2 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip2_rd_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip2_rd_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad) [5] xip2_rd_addr_en, 1:enable addr phase, MASTER ONLY [6] xip2_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7] xip2_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9
0xa1	GSPI_XIP2_RD_TRANSMODE	RW	[3:0] xip2_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip2_rd_transmode	0x97
0xa2	GSPI_XIP2_RD_CTRL0	RW	[0] cmd1_en [1] xip2_token_val_sel, 1:8'h69, 0:8'h00 [2] xip2_token_en [3] xip2_rd_dummy_cnt_add [4] xip2_ddr_mode [5] xip2_page_mode_en [6] xip2_timeout_mode_en 0:xip timeout disable 1:xip timeout enable [7] xip2_tcem_mode_en 0:xip tcem disable 1:xip tcem enable	0x40
0xa3	GSPI_XIP2_RD_CMD	RW	[7:0] xip2_rd_cmd, read command used for xip	0x3b

Address Offset	Name	Type	Description	Default Value
0xa4	GSPI_XIP2_WR_FMT	RW	[1:0] xip2_wr_data_lane, xip2 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip2_wr_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip2_wr_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5] xip2_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6] xip2_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/ Quad), MASTER ONLY [7] xip2_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0xa5	GSPI_XIP2_WR_TRANSMODE	RW	[3:0] xip2_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4] xip2_wr_transmode	0x10
0xa6	GSPI_XIP2_WR_CTRL0	RW	[0] cmd1_en	0x00
0xa7	GSP2_XIP2_WR_CMD	RW	[7:0] xip2_wr_cmd, write command used for xip	0x02
0xa8	GSPI_XIP3_RD_FMT	RW	[1:0] xip3_rd_data_lane, xip2 read data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip3_rd_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip3_rd_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad) [5] xip3_rd_addr_en, 1:enable addr phase, MASTER ONLY [6] xip3_rd_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/Quad), MASTER ONLY [7] xip3_rd_cmd_en, the spi command phase enable, MASTER ONLY	0xa9

Address Offset	Name	Type	Description	Default Value
0xa9	GSPI_XIP3_RD_TRANSMODE	RW	[3:0] xip3_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip3_rd_transmode	0x97
0xaa	GSPI_XIP3_RD_CTRL0	RW	[0] cmd1_en [1] xip3_token_val_sel, 1:8'h69, 0:8'h00 [2] xip3_token_en [3] xip3_rd_dummy_cnt_add [4] xip3_ddr_mode [5] xip3_page_mode_en [6] xip3_timeout_mode_en 0:xip timeout disable 1:xip timeout enable [7] xip3_tcem_mode_en 0:xip tcem disable 1:xip tcem enable	0x40
0xab	GSPI_XIP3_RD_CMD	RW	[7:0] xip3_rd_cmd, read command used for xip	0x3b
0xac	GSPI_XIP3_WR_FMT	RW	[1:0] xip3_wr_data_lane, xip2 write data lane 0: single, 1: dual, 2: quad, 3: octal [3:2] xip3_wr_addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes [4] xip3_wr_addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase (Dual/Quad), MASTER ONLY [5] xip3_wr_addr_en, 1:enabel addr phase, MASTER ONLY [6] xip3_wr_cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase (Dual/ Quad), MASTER ONLY [7] xip3_wr_cmd_en, the spi command phase enable, MASTER ONLY	0xa8
0xad	GSPI_XIP3_WR_TRANSMODE	RW	[3:0] xip3_wr_dummy_cnt dummy number = {dummy_cnt_add, dummy_cnt} + 1 [7:4] xip3_wr_transmode	0x10
0xaf	GSP2_XIP3_WR_CMD	RW	[7:0] xip3_wr_cmd, write command used for xip	0x02

Address Offset	Name	Type	Description	Default Value
0xb0	GSPI_XIP_SIZE_SET	RW	[1:0] xip_psram0_end_addr, psram0 space = {0, (xip_psram0_end_addr+1) * 16m} [3:2] xip_psram1_end_addr, psram1 space = {(xip_psram0_end_addr+1) * 16m, (xip_psram1_end_addr+1) * 16m} [5:4] xip_psram2_end_addr, psram2 space = {(xip_psram1_end_addr+1) * 16m, (xip_psram2_end_addr+1) * 16m} [7:6] xip_psram3_end_addr, psram3 space = {(xip_psram2_end_addr+1) * 16m, (xip_psram3_end_addr+1) * 16m}	0x03
0xb4	GSPI_XIP_RD_CMD1	RW	[7:0] xip_rd_cmd1, SPI Command1	0x00
0xb5	GSPI_XIP_WR_CMD1	RW	[7:0] xip_wr_cmd1, SPI Command1	0x00
0xb6	GSPI_XIP1_RD_CMD1	RW	[7:0] xip1_rd_cmd1, SPI Command1	0x00
0xb7	GSPI_XIP1_WR_CMD1	RW	[7:0] xip1_wr_cmd1, SPI Command1	0x00
0xb8	GSPI_XIP2_RD_CMD1	RW	[7:0] xip2_rd_cmd1, SPI Command1	0x00
0xb9	GSPI_XIP2_WR_CMD1	RW	[7:0] xip2_wr_cmd1, SPI Command1	0x00
0xba	GSPI_XIP3_RD_CMD1	RW	[7:0] xip3_rd_cmd1, SPI Command1	0x00
0xbb	GSPI_XIP3_WR_CMD1	RW	[7:0] xip3_wr_cmd1, SPI Command1	0x00

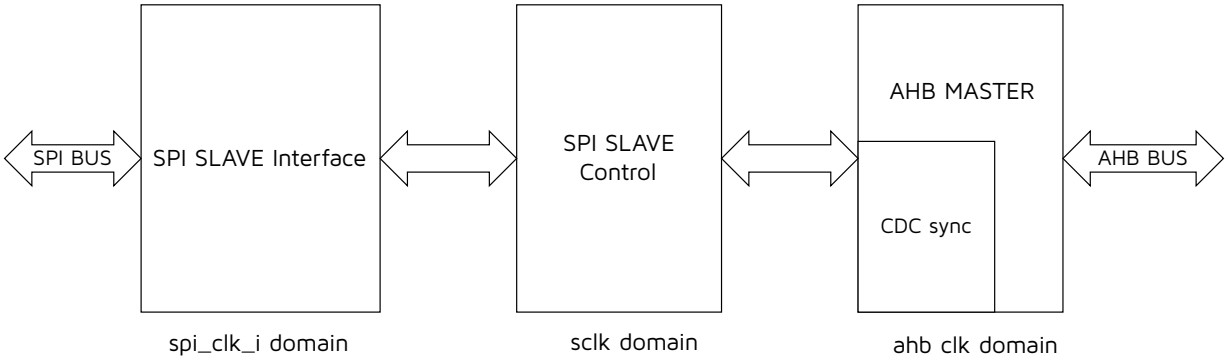
11.8 SPI Slave (SPI_SLV)

11.8.1 Diagram

SPI_SLV diagram is shown as below.

As shown in the diagram, SPI_SLAVE_Interface is used to analyze the protocol of the SPI interface, which is in spi_clk_i domain. SPI_SLAVE_Control is used to synchronize the data of spi_clk_i domain to sclk domain, and parse out the address and data segment to AHB_MASTER module. The AHB_MASTER module synchronizes the data from the sclk domain to the ahb clk domain and sends the parsed address and data to form the AHB bus.

Figure 11-18 SPI_SLV Diagram



11.8.2 Features

The SoC embeds SPI_SLV interface for debugging, features of SPI_SLV are listed as following:

- Supports SPI Slave mode
- Supports Dual I/O SPI interface

11.8.3 Function Description

This module converts SPI timing to AHB Master request.

SPI Master data should be read and written in formats specified by SPI_SLV, shown as following:

Figure 11-19 SPI_SLV Write Format

Cmd(8bit)	Addr(32bit)	Data0(1byte)	Data1(1byte)	Data

Figure 11-20 SPI_SLV Read Format

Cmd(8bit)	Addr(32bit)	Dummy(8/4cycle)	Data0(1byte)	Data1(1byte)	Data

SPI_SLV determines the format and operation by parsing the commands, shown in the following table.

Table 11-10 SPI_SLV Commands

Name	Description	Default
Cmd[7:0]	Cmd[7]: value 0:spi write, value 1:spi read Cmd[6]: value 0:addr single i/o, value 1:addr dual i/o cmd[5]: value 0:data single i/o, value 1:data dual i/o cmd[4]: value 0:addr auto increase, value 1:disable addr auto increase cmd[3]:value0:read dummy 8 cycle, value1:read dummy 4 cycle cmd[2]: value1:ahb word transfer cmd[1]: value1: ahb half word transfer cmd[0]: reserved	8'b0000_0000: spi slave write with addr single i/o and data single i/o in the addr auto increasing mode.

Address auto increase does not support ahb word/half word transfer.

SPI_CLK_in supported frequencies:

When read_dummy is 8, SPI_CLK_in frequency <= (1/2)*hclk frequency.

When read_dummy is 4, SPI_CLK_in frequency <= (1/4)*hclk frequency.

11.9 UART

11.9.1 Introduction

The SoC embeds UART (Universal Asynchronous Receiver/Transmitter) to implement full-duplex transmission and reception via UART TX and RX interface.

The UART module also supports ISO7816 protocol to enable communication with ISO/IEC 7816 integrated circuit card, especially smart card. In this mode, half-duplex communication (transmission or reception) is supported via the shared 7816_TRX interface.

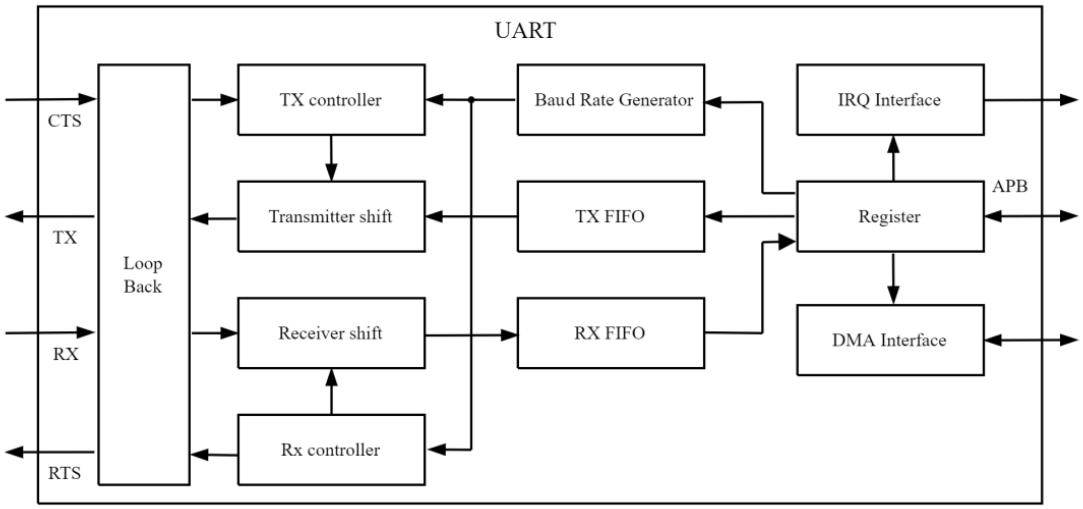
UART features include:

- Full-duplex operation
- Automatic flow control via RTS and CTS
- 8-bit UART mode, variable baud rate
- Optional even parity bit checking and generation
- Supports 1, 1.5 and 2 STOP bits
- Supports line breaks, parity errors, framing errors
- 8 bytes of transmit/receive FIFOs
- Supports ISO7816 protocol
- Supports DMA function (RX supports DMA Linked List Pointer)
- Up to 3 Mbps baud rate
- 3-channel UART (UART0, UART1, UART2)

11.9.2 Block Diagram

The figure below shows the block diagram of UART.

Figure 11-21 Block Diagram of UART



11.9.3 Function Description

11.9.3.1 Pin Configuration

The UART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- RX (Receive Data Input)

RX is the serial data input. Oversampling techniques are used for data recovery. In ISO7816 modes, this I/O is used to transmit and receive data.

- TX (Transmit Data Output)

When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High.

The following pins are required in Hardware flow control mode:

- CTS (Clear To Send)

When driven low (optional), this signal blocks the data transmission at the end of the current transfer.

- RTS (Request To Send)

When it is low, this signal indicates that the UART is ready to receive data.

11.9.3.2 Transmitter

(1) NDMA Operation

The transmitter comprises a Transmitter FIFO (TX FIFO), a Transmitter Shift, and a Transmitter Controller (TX controller). The TX FIFO holds data to be transferred through the serial interface. The TX FIFO can store up to 8 characters depending on hardware configurations and programming settings. The Transmitter Shift reads a character from the TX FIFO for the next transmission. The Transmitter Shift functions as a parallel-to-serial data converter, converting the outgoing character to serial bit streams. For each character transmission, the TX

Controller generates a START bit, an optional parity bit, and some number of STOP bits. The generation of parity bit and STOP bit can be configured by the uart_ctrl1 register. The TX FIFO is by default a 8-byte buffer called Transmitter Buffer Register.

(2) DMA Operation

When the TX FIFO under the threshold 4 characters, the UART controller asserts dma_tx_req to request a data transfer. The DMA controller should then transfer data to the TX FIFO followed by asserting dma_tx_ack. Next, the UART controller de-asserts dma_tx_req and the DMA controller de-asserts dma_tx_ack. The UART controller asserts dma_tx_req again unless the TX FIFO is full or the DMA transmission length is reached.

11.9.3.3 Receiver

(1) NDMA Operation

The receiver comprises a Receiver FIFO (RX FIFO), Receiver Shift, and a Receiver Controller (RX Controller). The RX Controller uses the oversampling clock generated by Baud Rate Generator to perform sampling at the center of each bit transmission. The received bits are shifted into the Receiver Shift for serial-to-parallel data conversion and the received character is stored into the RX FIFO. The RX FIFO is by default a 8byte buffer called the Receiver Buffer Register. The RX controller also detects some error conditions for each data transmission including parity error, framing error, or line break.

(2) DMA Operation

When the RX FIFO reaches the threshold 4 characters, the UART controller asserts dma_rx_req to request a data transfer. The DMA controller should then transfer data from the RX FIFO followed by asserting dma_rx_ack. Next, the UART controller de-asserts dma_rx_req and the DMA controller de-asserts dma_rx_ack. The UART controller asserts dma_rx_req again unless the RX FIFO is empty. DMA relies on rxdone to read parts of the data below 4 characters.

11.9.3.4 Baud Rate Generator

The Baud Rate Generator takes the UART clock as the source clock (pclk) and divides it with a divisor. The divisor consists of uart_clk_div and bpwc register. The uart_clk_div value is 15-bit in size and stored in two separate registers.

The formula for the divisor value is as follows:

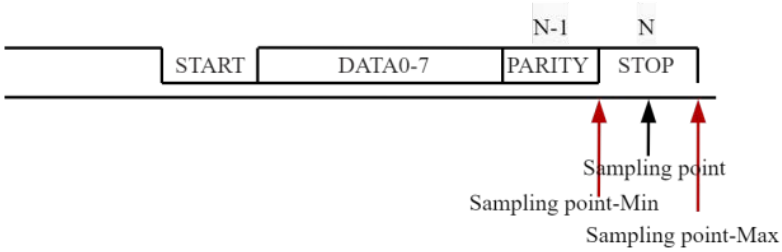
$$uart\_sclk = pclk/(uart\_clk\_div[14:0]+1)$$

$$Baudrate = uart\_sclk/(bpwc+1),\ bpwc > 2$$

Suppose that:

- T1 is the period of one bit transmission as perceived by the Rx Controller.
- T2 is the period of one bit transmission of the transmitter.
- N is the bit number for one frame of data – the START bit, data bits, parity bit, and the STOP bit(s).

Figure 11-22 UART Protocol Formats and Sampling Point



Then, the clock period tolerance for is as follows:

$$(N-1)\times T_1 \le (N-0.5)\times T_2$$

$$N\times T_1 \ge \left(N-\left(0.5-\frac{1}{bpwc+1}\right)\right)\times T_2$$

The calculation formula is obtained by conversion:

$$\left(1-\frac{\left(0.5-\frac{1}{bpwc+1}\right)}{N}\right)\times T_2 \le T_1 \le \left(\frac{N-0.5}{N-1}\right)\times T_2$$

Since T is the inverse of the baud rate, the actual baud rate generated by this controller in relation to the actual baud rate of the transmitter (the tolerance factor) can be within the range below:

$$\left(1-\frac{\left(0.5-\frac{1}{bpwc+1}\right)}{N}\right) \le \frac{\text{Actual baud rate}}{\text{Actual transmission baud rate}} \le \left(\frac{N-0.5}{N-1}\right)$$

If the character has one START bit, 8 data bits, one parity bit and one STOP bit, then N is 11 (1 + 8 + 1 + 1). The tolerance factor is from 0.9602 to 1.05. The table below shows clock tolerance factors as percentage of the actual Transmitter Baud Rates for typical values of N and bpwc register.

Table 11-11 Clock Variation Tolerance Factor

Typical Register Value	N = 10	N = 11	Conditions
bpwc = 15, uart_clk_div = 12	0.9563 - 1.056	0.9602 - 1.05	pclk: 24MHz baud rate: 115200bps
bpwc = 7, uart_clk_div = 25	0.9625 - 1.056	0.9659 - 1.05	

11.9.3.5 Loopback Mode

The UART provides a loopback mode for diagnostic testing without connecting an external device. When the loopback mode is enabled, the behavior of the controller is as follows:

- The output signals (TX, RTS) are disconnected from the TX/RX Controller and driven HIGH to avoid confusing the other end of the serial connection in case the connection exists.
- The input signals (RX, CTS) are disconnected from the TX/RX Controller and ignored.

- The TX Controller output values originally intended for the TX output signals are routed internally to replace the input signal of RX for the RX Controller, so every bit sent by the TX Controller is looped back and received by the RX controller. Note that CTS and RTS are similar.

11.9.3.6 Error Conditions

An ERROR event, in the form of a framing error, is generated if a valid stop bit is not detected in a frame. Second ERROR event, in the form of a break condition, is generated if the RX line is held active low for longer than the length of a data frame. Another ERROR event, Parity check bit error. The above ERROR event generates an rx_err interrupt.

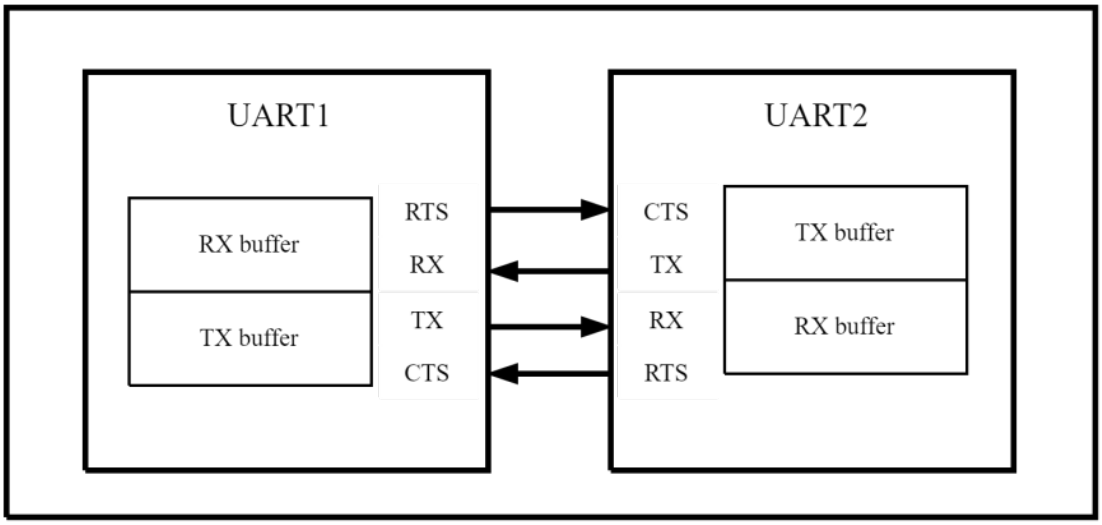
When an rx err interrupt is detected, perform the following operations:

Disable DMA and clear RX FIFO(irq_sts[2]).

11.9.3.7 Hardware Flow Control

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The figure below shows how to connect 2 devices in this mode:

Figure 11-23 Hardware Flow Control between 2 UARTs



If RX buffer of the UART is close to threshold, the UART sends a signal (configurable high or low level) via pin RTS to inform other device that it should stop sending data. Similarly, if the UART receives a signal from pin CTS, it indicates that RX buffer of other device is close to full and the UART should stop sending data.

The threshold can be configured using the register uart_ctrl2[3:0].

If the flow control is not enabled, the interface behaves as if the CTS and RTS lines are kept active all the time.

The RTS has two modes of control: manual and automatic.

In Manual mode, it is controlled via the register uart ctrl[6:5].

Automatic hardware flow control can be triggered in two ways:

- Automatic trigger RTS when the RX FIFO reaches the threshold(uart_ctrl2[3:0]).
- If rxdone_rts_en is configured to 1 and the rxdone is triggered at the same time, causing the RTS to be triggered.

Please refer to Register uart_ctrl2 for the use of the flow control function.

If rts_en is 0, rts function is off, rxdone_rts_en/rxtimeout_rts_en are both off (rxtimeout_rts_en is off to avoid rxdone_irq generation).

If rts_en is 1, the rts function is on, and rxdone_rts_en/rxtimeout_rts_en are both enabled.

(The rxtimeout_rts_en is enabled to stop sending data when the sender's CTS pin receives an active level on the RTS pin; If this is turned off at this time, no data is sent, resulting in an rxdone interrupt, and when rxdone_irq is cleared, the rx_fifo data is cleared, resulting in an rts failure. The rxdone_rts_en enable is to prevent the two sets of data from being so close that the software can't handle them.)

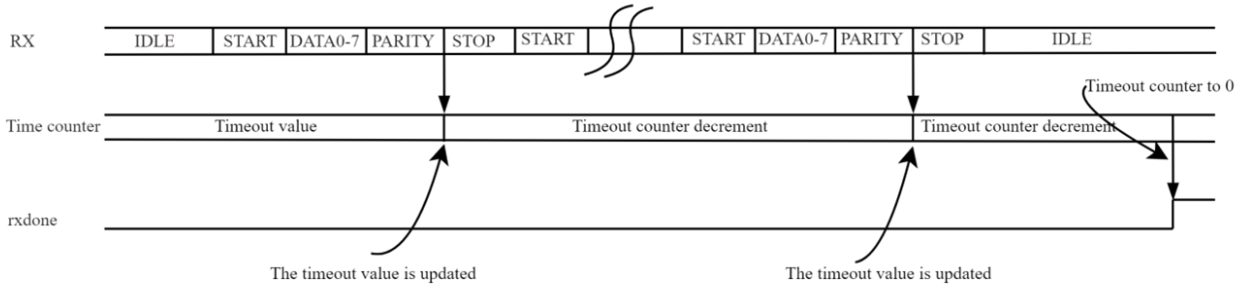
11.9.3.8 Receiver Timeout

Receiver timeout is used to handle when the data received per frame does not reach the threshold. Because data read from the Receiver Buffer Register is a multiple of 4 at a time, The rxdone interrupt is required to process the remaining data below the threshold.

NOTE:

- The DMA Operation threshold is fixed at 4.
- The NDMA Operation threshold can be configured through the register uart_ctrl3[3:0].

Figure 11-24 Timeout Flag Used for Data Transmission



The Time out counter inside the UART is updated at the STOP bit, and when receiving data stops, the Timeout counter decreases to 0 and generates a rxdone interrupt.

Note: If the register uart ctrl0[6] is configured to 1 and the RTS is triggered at the same time, causing the timeout counter to pause.

The configurable total timeout is determined via registers r_rxtimeout_l and r_rxtimeout_h[1:0].

Total timeout = r_rxtimeout_l * (r_rxtimeout_h + 1)

• The r_rxtimeout_l register:

The setting is transfer one bytes need cycles base on uart_clk. For example, if transfer one bytes (1start bit+8bits data+1 priority bit+2stop bits) total 12 bits, this register setting should be (register uart ctrl0[3:0]+1)*12.

• The r_rxtimeout_h[1:0] register:

2'b00: rx timeout time is r_rxtimeout[7:0]

2'b01: rx timeout time is r_rxtimeout[7:0]*2

2'b10: rx timeout time is r_rxtimeout[7:0]*3

3'b11: rx timeout time is r_rxtimeout[7:0]*4

The register r_rxtimeout (r_rxtimeout_l and r_rxtimeout_h) is for rx dma to decide the end of each transaction. Supposed the interval between each byte in one transaction is very short.

The minimum time supported via function timeout the time required for a single transmission of 1byte data, The maximum time is the maximum value supported via register r_rxtimeout. But registers r_rxtimeout_l and r_rxtimeout_h[1:0] still expect to follow our recommended approach.

11.9.4 UART Register Description

The UART related registers are listed in tables below.

For UART0 related register, the base address is 0x8140080; for UART1 related register, the base address is 0x81400c0; for UART2 related register, the base address is 0x81402c0.

Table 11-12 UART Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	UART_DATA_BUF0	Volatile	Write/read buffer[7:0]	0x00
0x01	UART_DATA_BUF1	Volatile	Write/read buffer[15:8]	0x00
0x02	UART_DATA_BUF2	Volatile	Write/read buffer[23:16]	0x00
0x03	UART_DATA_BUF3	Volatile	Write/read buffer[31:24]	0x00
0x04	UART_CLK_DIV_L	RW	uart_cli_div[7:0]:uart clk div register	0xff
0x05	UART_CLK_DIV_H	RW	uart_cli_div[15:8]:uart_sclk = sclk/ (uart_clk_div[14:0]+1) uart_clk_div[15]:1:enable clock divider, 0: disable.	0x0f
0x06	UART_CTRL0	RW	[3:0] bpwc_o, bpwc, bit width, should be larger than 2 Baudrate = uart_sclk/(bpwc+1) [4] auto_rxclr_en, DMA and ndma mode: auto clr function switch; 1:enable,0:disable [5] ndma_rxdone_en, NDMA mode: rxdone (timeout) function switch; 1:enable,0:disable;dma mode must disable [6] rxtimeout_rts_en, RTS controls timeout stop enabling signal [7] p7816_en_o, 7816 enable	0x7f

Address Offset	Name	Type	Description	Default Value
0x07	UART_CTRL1	RW	[0] tx_cts_polarity, cts select, 0: cts_i, 1: cts _i inverter [1] tx_cts_enable, cts enable, 1: enable, 0, disable [2] parity_enable, Parity, 1: enable, 0:disable [3] parity_polarity, even parity or odd [5:4] stop_sel, stop bit, 00: 1 bit, 01, 1.5bit 1x: 2bits [6] ttl_enable, ttl enable [7] loopback_o, uart tx, rx loopback	0x0e
0x08	UART_CTRL2	RW	[3:0] rts trig level [4] rts parity [5] rts manual value [6] rts manual enable [7] rts enable	0xa5
0x09	UART_CTRL3	RW	[3:0] rx_irq_trig level [7:4] tx_irq_trig level	0x44
0x0a	UART_RXTIMEOUT_O_L	RW	r_rxtimeout_o[7:0]:The setting is transfer one bytes need cycles base on uart_clk. For example, if transfer one bytes (1start bit+8bits data+1 priority bit+2stop bits) total 12 bits, this register setting should be (bpwc+1)*12.	0xc0
0x0b	UART_RXTIMEOUT_O_H	RW	[1:0] r_rxtimeout_o[9:8]:R_rxtimeout 2'b00:rx timeout time is r_rxtimeout[7:0] 2'b01:rx timeout time is r_rxtimeout[7:0]*2 2'b10:rx timeout time is r_rxtimeout[7:0]*3 3'b11: rx timeout time is r_rxtimeout[7:0]*4 R_rxtimeout is for rx dma to decide the end of each transaction. Supposed the interval between each byte in one transaction is very short. [2] mask_rx_irq, rx interrupt enable [3] mask_tx_irq, tx interrupt enable [4] mask_rxdone [5] mask_txdone [6] mask_err_irq [7] rsvd	0x01

Address Offset	Name	Type	Description	Default Value
0x0c	UART_BUFCNT	Volatile	[3:0] r_buf_cnt [7:4] t_buf_cnt	0x00
0x0d	UART_STATUS	Volatile	[2:0] rbcnt [3] irq [6:4] R: wbcnt [7] rxdone	0x00
0x0e	UART_TXRX_STATUS	Volatile	[1:0] R: rx_rem_cnt_d [2] rxbuf_irq; W:[2] write 1 to clear rx [3] txbuf_irq; W:[3] write 1 to clear tx [4] R: rxdone_irq; W:[4] write 1 to clear rxdone_irq [5] txdone; W:[5] write 1 to clear txdone [6] R: rx_err [7] R: Timeout	0x00
0x0f	UART_STATE	Volatile	[2:0] tstate_i, tx state machine; 0 -idle;1-start;2-byte;3-parity;4-stop;5-pop byte [3] rx_full [7:4] rx state machine, 0 -idle;1-start;2-bit;3-parity;4-stop;5-check parity;6-prepare;7-end bit; 8-lc parity; 9-wait; 10-push	0x00
0x10	UART_CTRL4	RW	[0] rxdone_rts_en 1'b1:rxdone work on rts; 1'b0:rxdone doesn't work on rts [1] timeout_en 1'b1:enable rxtimeout; 1'b0 disable rxtimeout [2] rx_timeout_reload_sel 1'b0: wr_o; 1'b1: REC_PREPARE state pulse [3] rts_stop_timeout_en Enables the counter to stop by rts [4] pem_event_en, pem event enable [5] rsvd [6] uart_en, uart enable [7] rsvd	0x4b

Address Offset	Name	Type	Description	Default Value
0x11	UART_RXTIMEOUT_O_EXP	RW	[7:0] r_rxtimeout_exp r_rxtimeout_o[9:8]:R_rxtimeout 2'b00:rx timeout time is r_rxtimeout[7:0] * r_rxtimeout_exp 2'b01:rx timeout time is r_rxtimeout[7:0]*2 * r_rxtimeout_exp 2'b10:rx timeout time is r_rxtimeout[7:0]*3 * r_rxtimeout_exp 3'b11: rx timeout time is r_rxtimeout[7:0]*4 * r_rxtimeout_exp R_rxtimeout is for rx dma to decide the end of each transaction. Supposed the interval between each byte in one transaction is very short.	0x00
0x11	UART_PEM_CTRL	RW	[4:0] pem_task_en	0x00

11.10 USB

11.10.1 Introduction

The SoC has a full-speed (12 Mbps) USB interface for communicating with other compatible digital devices. The USB interface acts as a USB peripheral, responding to requests from a master host controller. The chip contains internal 1.5kohm pull up resistor for the DP pin.

Telink USB interface supports the Universal Serial Bus Specification, Revision v2.0 (USB v2.0 Specification).

USB features include:

- Control/Bulk/Interrupt/Isochronous
- Control endpoint 0 and 8 configurable data endpoints
- The allowable maximum control transfer data payload sizes for full-speed is 8, 16, 32, or 64 bytes.
- Eight bidirectional endpoints (The maximum number of output endpoints is 2), Excluding control endpoint 0.
- Software controlled on-chip pull-up on D+
- Supports USB suspend, resume, and remote wake-up
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/ decoding and bit-stuffing
- Vendor CMD accesses registers via the AHB master

The chip supports 9 endpoints, including control endpoint 0 and 8 configurable data endpoints. Endpoint 1, 2, 3, 4, 7 and 8 can be configured as input endpoint, while endpoint 5 and 6 can be configured as output

endpoint. In audio class application, only endpoint 6 supports iso out mode, while endpoint 7 supports iso in mode. In other applications, each endpoint can be configured as bulk, interrupt and iso mode. For control endpoint 0, the chip's hardware vendor command is configurable.

Optional suspend mode:

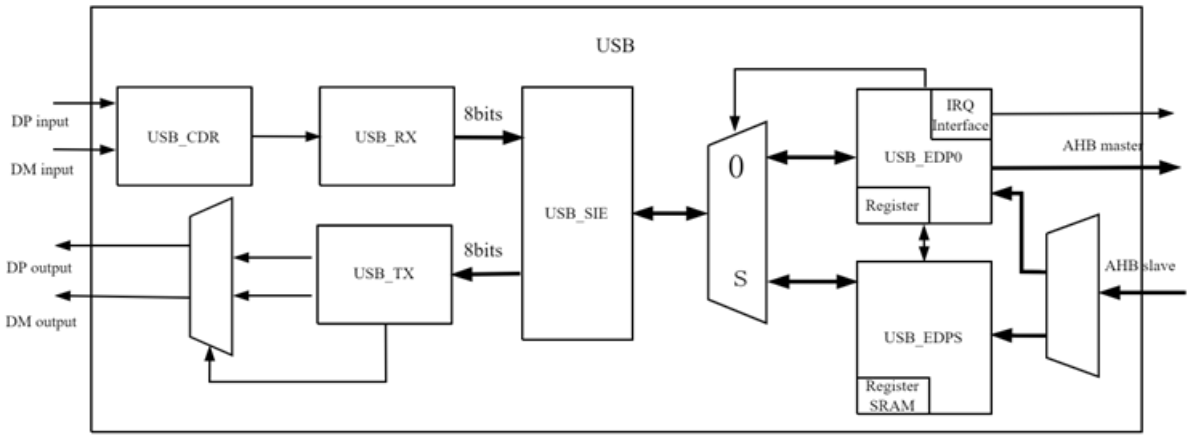
- Selectable as USB suspend mode or chip suspend mode, support remote wakeup.
- Current draw in suspend mode complied with USB v2.0 Specification.
- USB pins (DM, DP) can be used as GPIO function in suspend mode.
- Resume and detach detect: Recognize USB device by detecting the voltage on the DP pin with configurable 1.5K pull-up resistor.
- USB pins configurable as wakeup GPIOs.

The USB interface belongs to an independent power domain, and it can be configured to power down independently.

11.10.2 Block Diagram

The figure below shows the block diagram of USB.

Figure 11-25 Block Diagram of USB



11.10.3 USB Register Description

The USB related registers are listed in table below, the base address is 0x80110800.

Table 11-1 USB Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	EDP0PTR	Volatile	[3:0]: reg_ptr, Endpoint 0 buffer point	0x00
0x01	EDP0DAT	Volatile	[7:0]: buff, Endpoint 0 buffer data access address	0x00

Address Offset	Name	Type	Description	Default Value
0x02	EDPOCT	Volatile	[0]: ack_data, Ack data [1]: stall_data, Stall data [2]: ack_status, Ack status [3]: stall_status, Stall status	0x00
0x03	EDPOST	R	[0]: irq_reset, W: Clear usb reset edge interrupted [1]: irq_250us, W:Clear usb 250us edge interrupted [2]: suspend_i, USB suspend status read only: suspend [3]: irq_sof, W:Clear usb sof edge interrupted [4]: irq_setup, setup interrupt flag, W: Clear irq_setup interrupted [5]: irq_data, data interrupt flag, W: Clear irq_data interrupted [6]: irq_status, status interrupt flag, W: Clear irq_status interrupted [7]: irq_setinf, set interface interrupt flag, W: Clear irq_setinf interrupted	0x00
0x04	EDP0MODE	RW	[0]: r_en_sadr, enable auto decoding set_address command [1]: r_en_cfg, enable auto decoding set_config command [2]: r_en_inf, enable auto decoding set_interface command [3]: r_en_sta, enable auto decoding get_status command [4]: r_en_frm, enable auto decoding sync_frame command [5]: r_en_desc, enable auto decoding get_descriptor command [6]: r_en_fea, enable auto decoding set_feature command [7]: r_en_hw, enable auto decoding standard command	0xff
0x05	USBCT	RW	[0]: r_clk_sel_0, use auto calibrate clock if 1, use system clock if 0 [1]: low_speed, low speed mode if 1; full speed mode if 0 [2]: r_clk_sel_2, low jitter mode if 1 [3]: test_mode, usb test mode [7:4]: r_clk_sel_o, 2 for select 48M RC clock; 1 for 400M RC	0x01
0x06	CALCYCL	R	[7:0]: r_clk_div_il, r_clk_div_i[7:0]	0x00
0x07	CALCYCH	R	[2:0]: r_clk_div_ih, r_clk_div_i[10:8]	0x00

Address Offset	Name	Type	Description	Default Value
0x08	EDP0UDC	R	[2:0]: udc_cnt, number of data transferred	0x00
0x09	EDP0SIZE	/	[1:0]: r_cwptr_mux 2'b11--b4byte, 2'b10--32byte, 2'b01--16byte, 2'b00--8byte [3:2]: r_clens_mux 2'b11--7 bit width, 2'b10--6 bit width, 2'b01--5 bit width, 2'b00--4 bit width [4]: r_lvl0, lvl[0]:0-->irq_reset_edge; 1-->usb_reset_i [5]: r_lvl1, lvl[1]:0-->irq_250us_edge; 1-->usb_250us_i [6]: rsvd [7]: r_lvl3, lvl[3]:0-->irq_sof_edge; 1-->usb_sof_i	0x03
0x0a	MDEV	RW	[0]: r_mdev, self power, 1: self power, 0: bus power [1]: set_wakeup_feature, set_wakeup_feature by sw [2]: wakeup_feature_o, wakeup feature read only [3]: r_vend, r_vnd[0] vendor cmd offset (byte1[7] == r_vnd[0] means vendor cmd) [4]: r_vend_disable, 1 for disable vendor cmd [6:5]: mode_sel, 2'b0 --byte; 2'b1--halfword; 2'b2---word	0x18
0x0b	EDP0SIE	R	[6:0]: sie_adr_i, usb_address [7]: r_config, config_now	0x00
0x0c	SUSPENDCYC	RW	[4:0]: r_suspend_cnt, suspend_cnt [5]: r_edp0_stall, Avoid bugs with 8 multiples of in transfers [7:6]: rsvd	0x18
0x0d	INFALT	R	[7:0]: r_infalt, Interface and alternate setting number in last SET_INTERFACE command	0x00
0x0e	EDPS_EN	RW	[7:0]:edps_en	0xff

Address Offset	Name	Type	Description	Default Value
0x0f	IRQ_MASK	RW	[0]: r_mask0, mask[0]: irq_reset [1]: r_mask1, mask[1]: irq_250us [2]: r_mask2, mask[2]: irq_suspend [3]: r_mask3, mask[3]:irq_sof [4]: r_mask4, mask[4]:irq_setup [5]: r_mask5, mask[5]:irq_data [6]: r_mask6, mask[6]:irq_status [7]: r_mask7, mask[7]:irq_setinf	0x04
0x10	EDPSPTR	Volatile	[7:0]: rd_ptrl	0x00
0x11	EDPS1PTR	Volatile	[7:0]: rd_ptrl	0x00
0x12	EDPS2PTR	Volatile	[7:0]: rd_ptrl	0x00
0x13	EDPS3PTR	Volatile	[7:0]: rd_ptrl	0x00
0x14	EDPS4PTR	Volatile	[7:0]: rd_ptrl	0x00
0x15	EDPS5PTR	Volatile	[7:0]: rd_ptrl	0x00
0x16	EDPS6PTR	Volatile	[7:0]: rd_ptrl	0x00
0x17	EDPS7PTR	Volatile	[7:0]: rd_ptrl	0x60
0x18	EDPSPTRH	Volatile	[2:0]: rd_ptrh	0x00
0x19	EDPS1PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x1a	EDPS2PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x1b	EDPS3PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x1c	EDPS4PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x1d	EDPS5PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x1e	EDPS6PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x1f	EDPS7PTRH	Volatile	[2:0]: rd_ptrh	0x00
0x20	EDPSDATA	Volatile	[7:0]: sr_q	0x00
0x21	EDPS1DATA	Volatile	[7:0]: sr_q	0x00
0x22	EDPS2DATA	Volatile	[7:0]: sr_q	0x00
0x23	EDPS3DATA	Volatile	[7:0]: sr_q	0x00
0x24	EDPS4DATA	Volatile	[7:0]: sr_q	0x00

Address Offset	Name	Type	Description	Default Value
0x25	EDPS5DATA	Volatile	[7:0]: sr_q	0x00
0x26	EDPS6DATA	Volatile	[7:0]: sr_q	0x00
0x27	EDPS7DATA	Volatile	[7:0]: sr_q	0x00
0x28	EDPSCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1 [7]: edp8_dma_eof, Launch EOF for FIFO mode (W) (no support)	0x00
0x29	EDP1SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2a	EDP2SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2b	EDP3SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2c	EDP4SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2d	EDP5SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00

Address Offset	Name	Type	Description	Default Value
0x2e	EDP6SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1 [6]: rd_mono_aout, MONO mode [7]: rd_en_aout, Audio ISO out enable	-
0x2f	EDP7SCT	Volatile	[0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1 [6]: rd_mono_aout, MONO mode [7]: rd_en_ain, Audio ISO in enable	-
0x30	EDPSADR	RW	Endpoint 8(0) buffer address low	0x80
0x31	EDPS1ADR	RW	Endpoint 1 buffer address low	0x00
0x32	EDPS2ADR	RW	Endpoint 2 buffer address low	0x08
0x33	EDPS3ADR	RW	Endpoint 3 buffer address low	0x10
0x34	EDPS4ADR	RW	Endpoint 4 buffer address low	0x40
0x35	EDPS5ADR	RW	Endpoint 5 buffer address low	0xc0
0x36	EDPS6ADR	RW	Endpoint 6 buffer address low	0x20
0x37	EDPS7ADR	RW	Endpoint 7 buffer address low	0x30
0x38	EDPSADRH	RW	[1:0] Endpoint 8(0) buffer address high	0x00
0x39	EDPS1ADRH	RW	[1:0] Endpoint 1 buffer address high	0x00
0x3a	EDPS2ADRH	RW	[1:0] Endpoint 2 buffer address high	0x00
0x3b	EDPS3ADRH	RW	[1:0] Endpoint 3 buffer address high	0x00
0x3c	EDPS4ADRH	RW	[1:0] Endpoint 4 buffer address high	0x00
0x3d	EDPS5ADRH	RW	[1:0] Endpoint 5 buffer address high	0x00
0x3e	EDPS6ADRH	RW	[1:0] Endpoint 6 buffer address high	0x00
0x3f	EDPS7ADRH	RW	[1:0] Endpoint 7 buffer address high	0x00
0x40	USBSO	RW	Enable endpoint ISO mode	0xc0

Address Offset	Name	Type	Description	Default Value
0x41	USBIRQ	RW	[7:0]: r_irq, Endpoint data transfer interrupt	0x00
0x42	USBMASK	RW	Endpoint interrupt mask	0xff
0x43	USBMAX0	RW	Maximum endpoint 8 transfer number: max_size = {USBMAX0[7:0],5'h0}	0x10
0x44	USBMIN0	RW	Minimum threshold to ACK endpoint 8 transfer (the buffer must have USBMIN8 data to ack to IN YOKEN)	0x40
0x45	USBFIFO	RW	[0]: r_fifo0, Endpoint 0 FIFO mode: the pointer of endpoint8 auto as a circuit buffer [1]: full0, Full flag [2]: r_mode00 [3]: edp8_eof [6:4]: edp8_dma_eof [7]: rsvd	-
0x46	USBMAX	RW	[7:0]: max_in, Max data in for endpoint buffer (except 7): Max_data_size =USBMAX*8	0x08
0x47	USBTICK	Volatile	[7:0] r_tick, Just a tick that increase on posedge of the sclk_usb	0x00
0x48	USBRAM	RW	[0]: sr_cen, CEN in power down mode [1]: sr_clk, CLK in power down mode [2]: r_ram2, Reserved [3]: wen_i, WEN in power down mode [4]: r_ram4, CEN in function mode	0x18
0x49	USBMIN1	RW	[2:0] usb_blk1, Minimum threshold to ACK endpoint 8 transfer [3] r_edps_map_manual [4] r_edps_map_auto [5] r_edps_sm_map_en [6] r_edps_map_tgl_en [7] r_get_sta_map_en	0x00
0x4a	EDPS_MAP0	RW	[3:0] r_edps8_map [7:4] r_edps1_map	0x55

Address Offset	Name	Type	Description	Default Value
0x4b	EDPS_MAP1	RW	[3:0] r_edps2_map [7:4] r_edps3_map	0x55
0x4c	EDPS_MAP2	RW	[3:0] r_edps4_map [7:4] r_edps5_map	0x15
0x4d	EDPS_MAP3	RW	[3:0] r_edps6_map [7:4] r_edps7_map	0x67
0x4e	SOF_FRAME0	R	[7:0] sie_framel, Bit7-0 of Frame number	0x00
0x4f	SOF_FRAME1	R	[2:0] sie_framel, Bit10-8 of Frame number	0x00
0x50	EDPS_MAXH0	RW/R	[0] r_maxh0 (RW), Maximum endpoint 8 transfer number:max_size = {USBMAXH0[0],USBMAX0[7:0],3'h0} [1] r_en_ain_maxh0 (RW), Maximum endpoint 7 transfer number(r_en_ain set 1): max_size = {USBAINMAXH0[0],10'h3ff} [5] full (R), Read-write pointer equality [6] full_nz (R), the read and write Pointers are equal and not 0 [7] full_thrd (R), the read and write Pointers are equal and not 0, and reach register r_eptr	0x20
0x51	EDPS_EDP_R	R	[3:0] sie_edp, the endpoint number before the Map [7:4] sie_edp_nomap, the endpoint number after the Map	0x00
0x52	USB_PID_L	RW	[7:0] usb_pid_l	0x20
0x53	USB_PID_H	RW	[7:0] usb_pid_h	0x53
0x54	EDPS_EPTRL	RW	[7:0] r_eptr_l, Bit7-0 of the configuration register of the empty packet flag bit	0x40
0x55	EDPS_EPTRH	RW	[2:0] r_eptr_h, Bit10-8 of the configuration register of the empty packet flag bit	0x00
0x56	EDPS_UDC_PTRL	R	[7:0] r_udc_ptr_l, Bit 7-0 of Host pointer	0x00
0x57	EDPS_UDC_PTRH	R	[2:0] r_udc_ptr_h, Bit 10-8 of Host pointer	0x00
0x58	EDPS_S_PTRL	R	[7:0] r_s_ptr_l, Bit 7-0 of MCU pointer	0x00
0x59	EDPS_S_PTRH	R	[2:0] r_s_ptr_h, Bit 10-8 of MCU pointer	0x00

Address Offset	Name	Type	Description	Default Value
0x5a	EDPS_MAP_EN	RW	[7:0] r_edps_map_en	0x00
0x5b	EDPS_LOGIC_EN	RW	[7:0] r_edps_logic_en	0xff
0x5c	EDPS_FULL_THRD	R	[0] edps_full_thrd8, Edps8 empty flag, read/write pointer is needed to reach register r_eptr [1] edps_full_thrd1, Edps1 empty flag, read/write pointer is needed to reach register r_eptr [2] edps_full_thrd2, Edps2 empty flag, read/write pointer is needed to reach register r_eptr [3] edps_full_thrd3, Edps3 empty flag, read/write pointer is needed to reach register r_eptr [4] edps_full_thrd4, Edps4 empty flag, read/write pointer is needed to reach register r_eptr [5] edps_full_thrd5, Edps5 empty flag, read/write pointer is needed to reach register r_eptr [6] edps_full_thrd6, Edps6 empty flag, read/write pointer is needed to reach register r_eptr [7] edps_full_thrd7, Edps7 empty flag, read/write pointer is needed to reach register r_eptr	0x00
0x5d	PEM_CTRL	RW	[0] pem_event_en [1] pem_event_sel 1'b1: edps irq; 1'b0: edp0 irq	0x00
0x5e	PEM_CTRL1	W	[7:0] pem_task_en	0x00
0x60	TSTAMP0	R	[7:0] Bit 7-0 of system timer tick value that is latched via SOF	0x00
0x61	TSTAMP1	R	[7:0] Bit 15-8 of system timer tick value that is latched via SOF	0x00
0x62	TSTAMP2	R	[7:0] Bit 23-16 of system timer tick value that is latched via SOF	0x00
0x63	TSTAMP3	R	[7:0] Bit 31-24 of system timer tick value that is latched via SOF	0x00

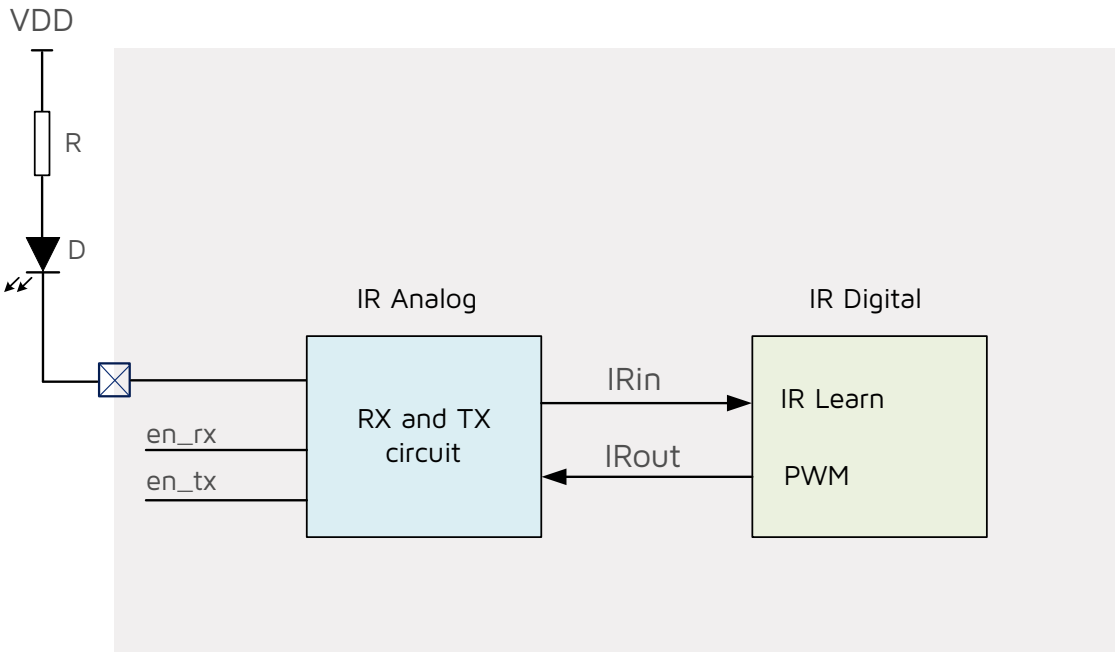
12 IR (Infrared)

12.1 Overview of IR

The SoC embeds an IR (Infrared) module which includes analog part and digital part. Outside the chip, only a current limiting resistor and an IR diode are needed to implement the IR function.

The block diagram of the IR module is shown as below.

Figure 12-1 Block Diagram of IR module



12.2 IR Analog (RX and TX circuit)

12.2.1 Function Configuration

- IR RX (receive) function: write analog register 0x0f<3> as 1, 0x11<7> as 0 to enable the IR RX function.
- IR TX (transmit) function: write analog register 0x0f<3> as 0, 0x11<7> as 1 to enable the IR TX function.

The receiving signal of the IR digital part is IRin inside the chip, and the transmitting signal of the IR digital part is IRout inside the chip. These two signals are realized as internal routings in the chip.

12.2.2 Current Limitation

Due to the internal circuit of the chip, the power pad and the package bonding wire have current limitations. The IR RX and TX circuit integrated into the chip needs to limit the transient current when the IR TX function is turned on, which can be realized by changing the resistance value of the current limiting resistor R. Currently, we need to limit the transient current of the IR TX to less than 200mA on the design.

12.3 IR Digital (IR Learn and PWM)

The function of IR learn part is mainly counting, there is an internal 24bit counter which can count the pulse width duration (hight) and cycle duration of the IR signal carrier.

12.3.1 Working Principle

The source of the IR signal is selected through register IR_CTRL1[7], when IR_CTRL1[7] is 0, the input of GPIO is selected as the IR signal, when IR_CTRL1[7] is 1, the IR output signal of ANALOG is selected as the IR signal.

After register IR_CTRL0[0] is set to 1, the condition that triggers the 24bit counter to start counting is determined according to the configuration of register IR_CTRL0[1:0]. When register IR_CTRL0[1:0 are] is configured to 0, the 24bit counter starts counting when the first rising edge of the IR signal arrives.

- When the level of IR signal goes from high to low, the current counter value is stored in rxfifo as the carrier's high, and if register INT_MASK[0] is set to 1 before that, a high interrupt is generated, and the status of the high interrupt can be read by reading register INT_STAT[0];
- When the level of IR signal is from low to high, the current counter value is stored in rxfifo as the carrier's cycle, and the counter is reset to 1. If register INT_MASK[1] is set to 1 before that, a cycle interrupt is generated, and the status of the cycle interrupt can be read by reading register INT_STAT[1];
- When the IR signal is low and the counter value reaches the preset timeout threshold, the current counter value is stored in rxfifo as the cycle of the carrier and the counter is reset to 0. If register INT_MASK[2] is set to 1 before this, a timeout interrupt is generated and the status of the timeout interrupt can be read by reading register INT_STAT[2]. The timeout interrupt is generated by reading INT_STAT[2].

NOTE:

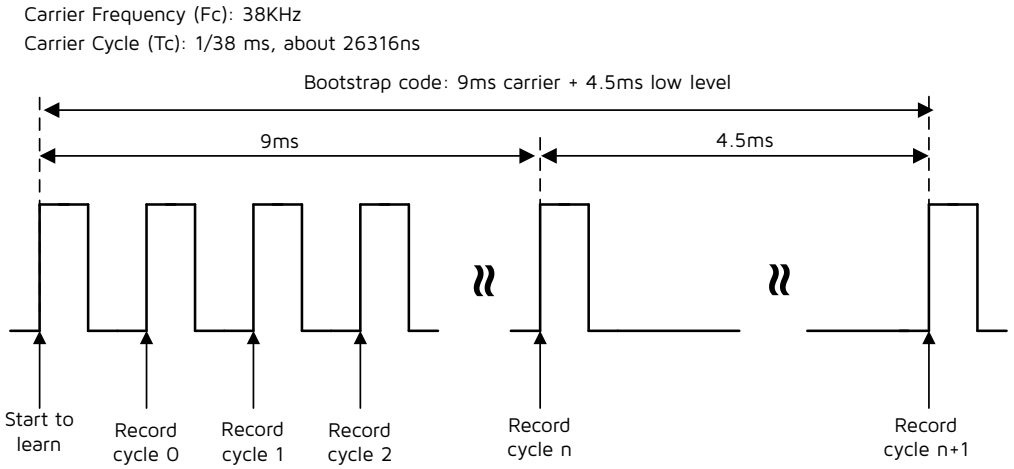
• If the timeout threshold is set to more than 16bit, IR_CTRL1[5] must be configured to 0, i.e., the data is stored into the rxfifo with a bit width of 24bit

12.3.2 Working Mode

The module supports two modes to read the data in rxfifo: non-DMA mode (read by rx_buf interrupt method), and DMA mode (read by DMA).

The following is an example of bootstrap code learning for the NEC protocol:

Figure 12-2 Bootstrap Code Learning for NEC Protocol



The detailed learning procedure is as follows.

1. At the rising edge of the first pulse, the module starts learning and the counter starts counting. At each subsequent rising edge moment, the value of cycle is recorded;
2. Assuming the carrier frequency Fc = 38KHz for the NEC protocol, the carrier cycle Tc is approximately 26316ns; Assuming PCLK = 24MHz, Tclk is approximately 41ns; If cycle * Tclk == Tc, it can be determined that the current cycle is a carrier cycle. If cycle * Tclk! = Tc, then you can determine that the current cycle is a non-carrier cycle.
3. As shown in the figure, cycle0, cycle1, cycle2... cycleen are all carrier cycles; because cycle n+1 is much larger than the value of cycle0~cyclen, it is a non-carrier cycle. If the time sum of all carrier cycles (cycle0, cycle1, cycle2... cycleen) time sum is about equal to 9ms and when cycle n+1 is about equal to 4.5ms, it can be judged that the current wave is the bootstrap code of NEC protocol;
4. The learning of data codes 0 and 1 of the NEC protocol is similar;
5. When the value of the off-carrier cycle is equal to the threshold of the preset timeout, the signal number composed of this off-carrier cycle can be judged as the end code.

12.4 IR Register Description

The IR related analog registers are listed below.

Table 12-1 IR Related Analog Registers

Address	Type	Description	Reset Value
afe_0x2<7:5>	R/W	ir_r_trim_30k	100
afe_0xf<3>	R/W	IR_receiver_en	0
afe_0x11<7>	R/W	IR_transmitter_en IR_transmitter enable: 1: enable 0:disable (default)	0

Address	Type	Description	Reset Value
afe_0x14<2:0>	R/W	ir_rtrim<3:0>	0
afe_0x14<3>	R/W	ir_en_ex_diod	1000
afe_0x14<7:4>	R/W	ir_htrim<3:0>	100

The IR related digital registers are listed in the following table. The base address for the following IR related registers is 0x801404C0.

Table 12-2 IR Related Digital Registers

Address Offset	Name	Type	Description	Default Value
0x00	IR_CTRL0	W	[0]: en 1: enable module 0: disable module [1]: rxfifo_clr 1: clear all counters within rxfifo 0: do not clear all counters within rxfifo	0x00

Address Offset	Name	Type	Description	Default Value
0x01	IR_CTRL1	R/W	[1:0]: ir_mode 2'b00: after enabling the module, the counter waits for the first rising edge to enter the HIGH state to start counting. 2'b01: after enabling the module, the counter enters the HIGH state immediately. 2'b10: after enabling the module, the counter waits for the first falling edge to enter the LOW state to start counting. 2'b11: after enabling the module, the counter enters the LOW state immediately. [2]: ir_inv 1: IR input signal is inverted 0: IR input signal is not inverted [3]: high_wr_en 1: high is stored in rxfifo 0: high is not stored in rxfifo [4]: cycle_wr_en 1: cycle is stored in rxfifo 0: cycle is not stored in rxfifo [5]: data_mode, the bit width of data stored in rxfifo 1: store 16bits, two 16bit data spliced into one word 0: store 24bits, high complement zero to form a word [6]: timeout_dis 1: disable timeout mechanism 0: enable timeout mechanism [7]: ir_sel. 1: select the IR input from analogue 0: select the IR input from pad	0x00
0x02	IR_TO_0	R/W	[7:0]: ir_timeout[7:0] timeout threshold byte0	0xff
0x03	IR_TO_1	R/W	[7:0]: ir_timeout[15:8] timeout threshold byte1	0xff
0x04	IR_TO_2	R/W	[7:0]: ir_timeout[23:16] timeout threshold byte2	0x00

Address Offset	Name	Type	Description	Default Value
0x05	INT_MASK	R/W	[0]: high interrupt enable [1]: cycle interrupt enable [2]: time out interrupt enable [3]: rx_buf interrupt enable	0x00
0x06	INT_STAT	VOLATILE	[0]: high_irq, High interrupt request status, write 1 to clear 0 [1]: cycle_irq, Cycle interrupt request status, write 1 to clear 0 [2]: timeout_irq Timeout interrupt request status, write 1 to clear 0 [3]: rx_buf_irq rx_buf interrupt request status, write 1 to clear 0, volatile	0x00
0x07	IR_FIFO_CTRL	R/W	[2:0]: fifo_lvl, threshold to trigger rx_buf interrupt request [3]: auto_rxclr_en, auto clear all counters in rxfifo in dma mode 1: enable, 0: disable [4]: pem_event_en [5]: pem_task0_en [6]: pem_task1_en	0x00
0x08	IR_STAT	VOLATILE	[3:0]: rx_buf_cnt [4]: rx_full [5]: rx_empty [6]: rx_done	0x00
0x0c	IR_RDAT_0	R	[7:0]: rx_rdat[7:0]	0x00
0x0d	IR_RDAT_1	R	[7:0]: rx_rdat[15:8]	0x00
0x0e	IR_RDAT_2	R	[7:0]: rx_rdat[23:16]	0x00
0x0f	IR_RDAT_3	R	[7:0]: rx_rdat[31:24]	0x00

13 Return to Zero (RZ)

13.1 Overview

The RZ module uses a single-wire return-to-zero code protocol to communicate and transmit, and can drive pixel ICs in series or parallel;

Both the data and the feature code of the zeroing code consist of a segment of high level and a segment of low level;

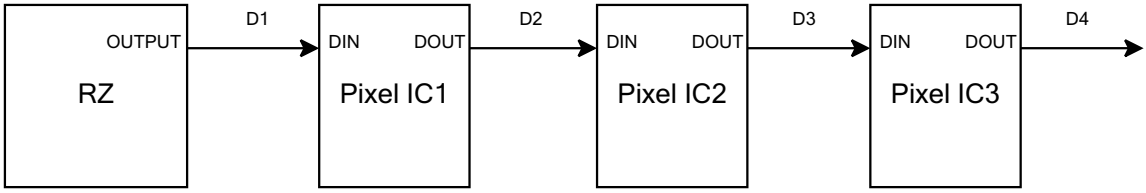
The pixel ICs are addressed in two ways:

- Serial sequential addressing;
- Parallel random addressing.

13.2 Mechanism of Serial Sequential Addressing

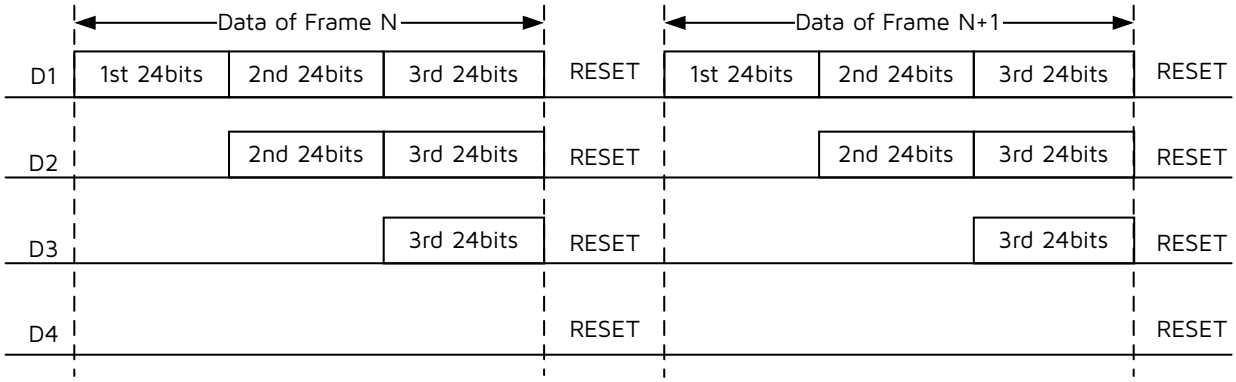
13.2.1 Diagram of series connection of Pixel ICs

Figure 13-1 Series Connection of Pixel ICs



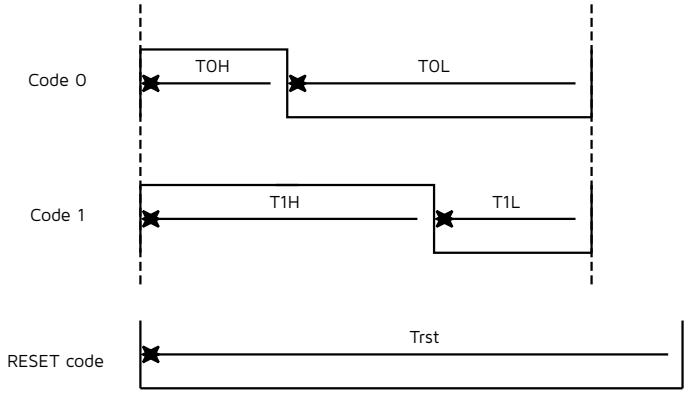
As shown in the above figure, three Pixel ICs work in series, D1 is the data sent by the RZ module, and D2, D3, and D4 are the RZ code data forwarded by the series chips. Assuming that the data required by each chip is 24 bits, the data transmission can be obtained as shown below. The RZ module sends multiple frames of data, each of which contains 3 bits of 24-bit data, and a RESET code is sent immediately after each frame of data.

Figure 13-2 RZ Data Transmission



13.2.2 Timing Sequence of RZ

Figure 13-3 Timing Sequence of RZ for Serial Sequential Addressing



As shown in the figure above, the RZ code protocol for driving a series-operated Pixel IC consists of three code elements: code 0, code 1, and RESET code, and the three code elements are differentiated by the duration of the high and low levels, which can be configured through registers T0H and T0L for the duration of the high and low levels of code 0, through registers T1H and T1L for the duration of the high and low levels of code 1, and through registers TSRH and TSRL for the duration of the high and low levels of the RESET code.

13.2.3 Example of Mechanism of the RZ module to Drive series Pixel ICs

13.2.3.1 No global data in the data sent by the RZ module

Assuming that there is no global data in the data to be sent by the RZ module, the RZ module drives a total of two Pixel ICs connected in series, and the data required to be ground for each Pixel IC is 36bits; therefore, register RZ_CTRL1[7] is configured as 0, register GLOBAL_DATA_NUM is configured as 0, register PIXEL_NUM is configured as 1, and register PIXEL_DATA_NUM is configured as 35.

1. Case1

Assuming that the data stored in memory is aligned at 8 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 0, and both registers RZ_CTRL[4] and RZ_CTRL[3] are configured as 1. Then, the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-4 Data filling in memory for case 1

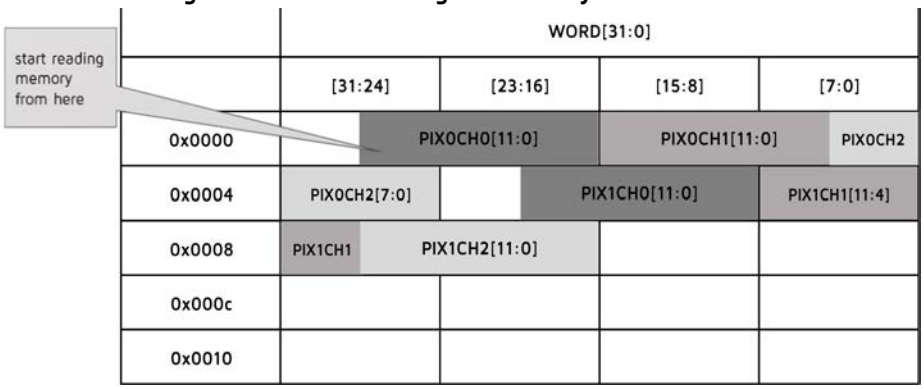


Figure 13-5 Data output sequence of from RZ module for case 1

2. Case2

Assuming that the data stored in memory is aligned at 8 bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 0, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 0. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-6 Data filling in memory for case 2

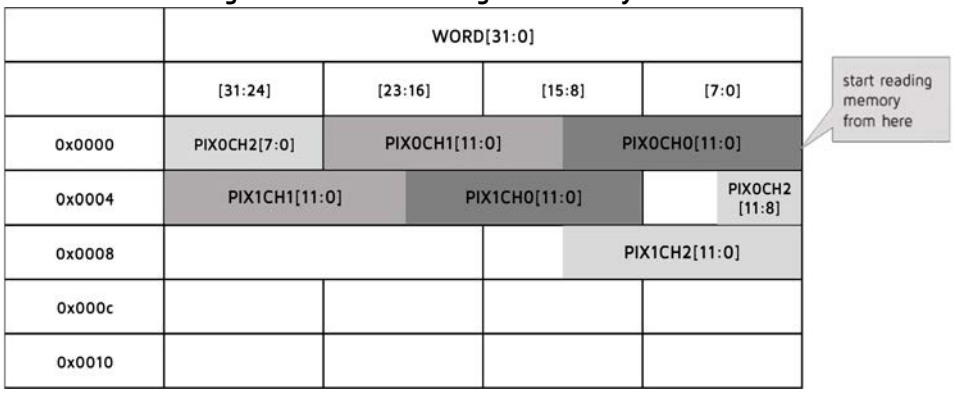


Figure 13-7 Data output sequence of from RZ module for case 2

3. Case3

Assuming that the data stored in memory is aligned at 32 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 1. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-8 Data filling in memory for case 3

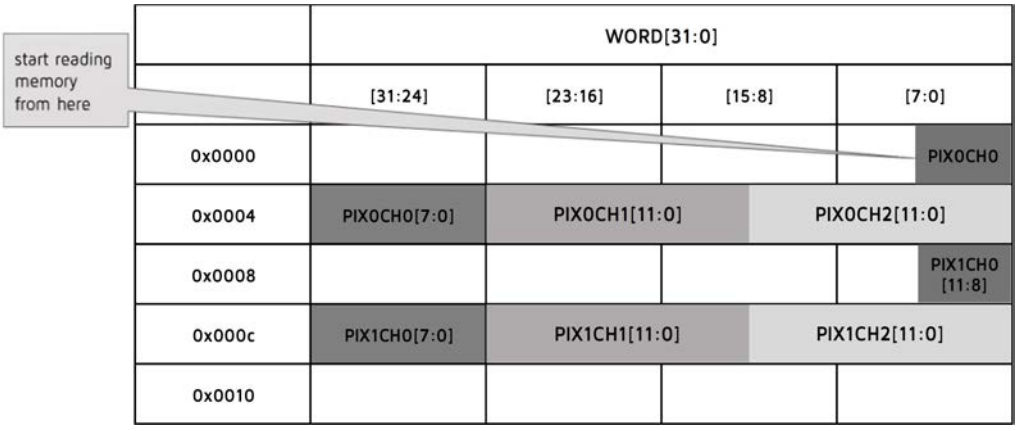


Figure 13-9 Data output sequence of from RZ module for case 3

4. Case4

Assuming that the data stored in memory is aligned at 32 bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured as 0. Then, the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-10 Data filling in memory for case 4

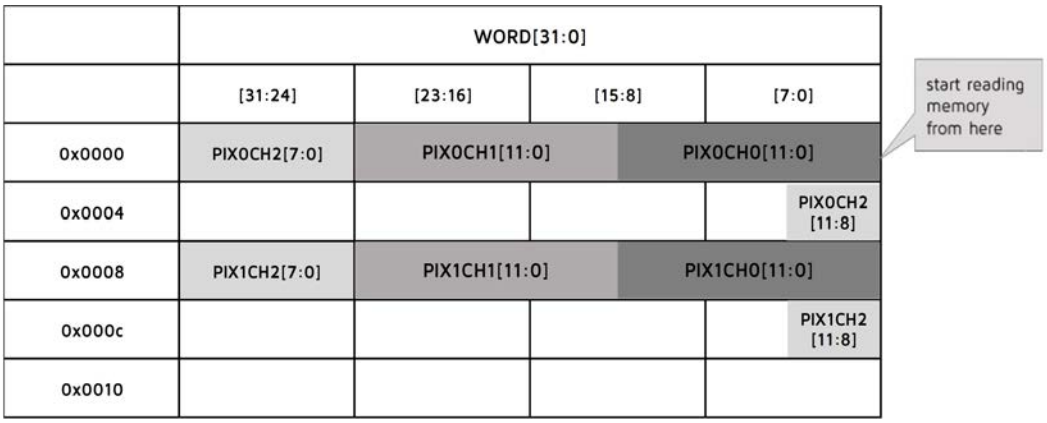


Figure 13-11 Data output sequence of from RZ module for case 4

13.2.3.2 There is global data in the data sent by the RZ module 1

Assuming that there is global data in the data to be sent by the RZ module and that global data is inserted after each Pixel IC data, the global data is 10 bits, the RZ module drives a total of two Pixel ICs in series, and the data required for each Pixel IC is 36 bits; therefore, register RZ_CTRL1[7] is configured to 0, register GLOBAL_DATA_NUM is configured as 10, register RZ_CTRL[6] is configured as 0, register PIXEL_NUM is configured as 1, register PIXEL_DATA_NUM is configured as 35.

1. Case1

Assuming that the data stored in memory is aligned at 8 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 0, and both registers RZ_CTRL[4] and RZ_CTRL[3] are configured as 1. Then, the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-12 Data filling in memory for case 1

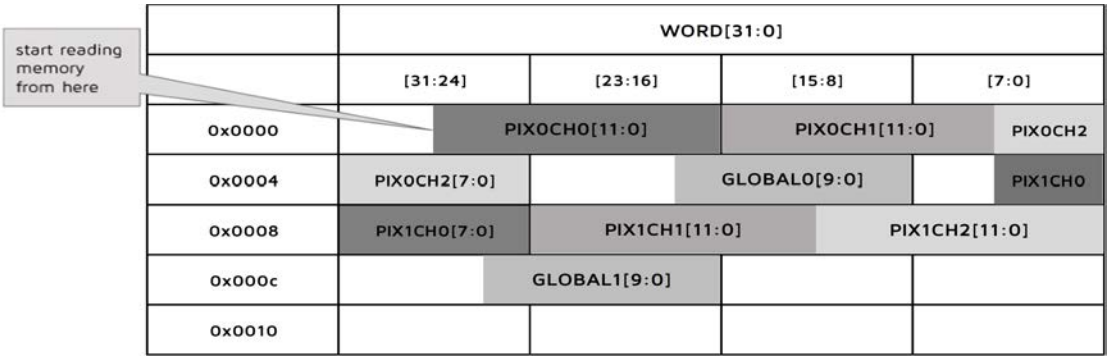


Figure 13-13 Data output sequence of from RZ module for case 1

2. Case2

Assuming that the data stored in memory is aligned at 8 bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 0, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 0. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-14 Data filling in memory for case 2

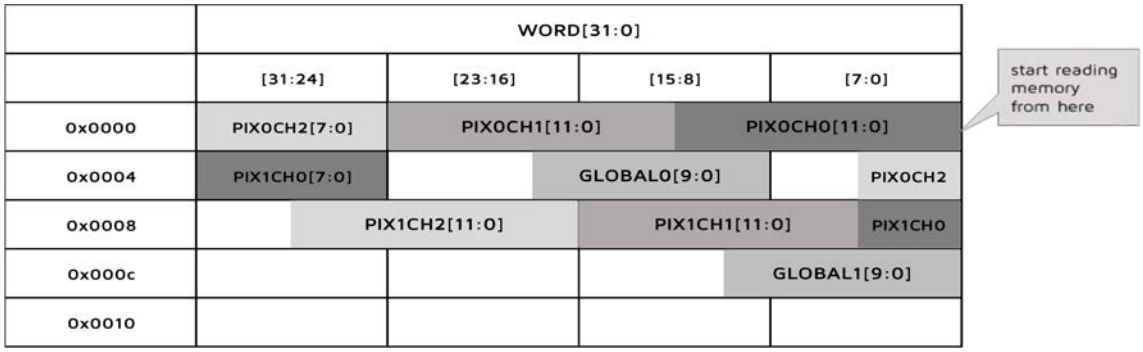


Figure 13-15 Data output sequence of from RZ module for case 2

3. Case3

Assuming that the data stored in memory is aligned at 32 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 1. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-16 Data filling in memory for case 3

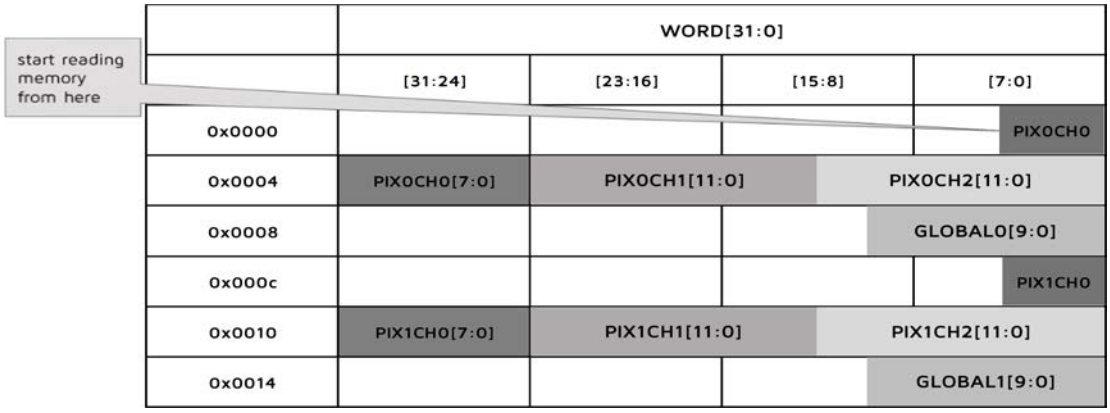


Figure 13-17 Data output sequence of from RZ module for case 3

4. Case4

Assuming that the data stored in memory is aligned at 32bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured as 0. Then, the schematic diagram of the data filling in memory and the timing diagram of the data output from the RZ module are as the following two respectively shown:

Figure 13-18 Data filling in memory for case 4

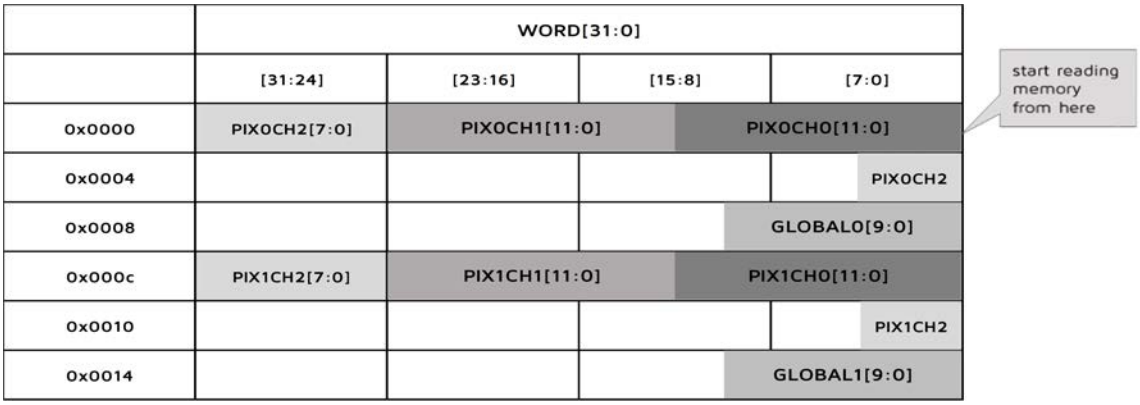


Figure 13-19 Data output sequence of from RZ module for case 4

13.2.3.3 There is global data in the data sent by the RZ module 2

Assuming that there is global data in the data to be sent by the RZ module and that global data is inserted after all Pixel IC data, the global data is 10 bits, the RZ module drives a total of two Pixel ICs in series, and the data required to be ground for each Pixel IC is 36bits; therefore, register RZ_CTRL1[7] is configured as 0,

register GLOBAL_DATA_NUM is configured as 10, register RZ_CTRL[6] is configured as 1, register PIXEL_NUM is configured as 1, and register PIXEL_DATA_NUM is configured as 35.

1. Case1

Assuming that the data stored in memory is aligned at 8 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 0, and both registers RZ_CTRL[4] and RZ_CTRL[3] are configured as 1. Then, the diagram of the data filling in memory and the timing diagram of the data output from the RZ module are as the following two figures:

Figure 13-20 Data filling in memory for case 1

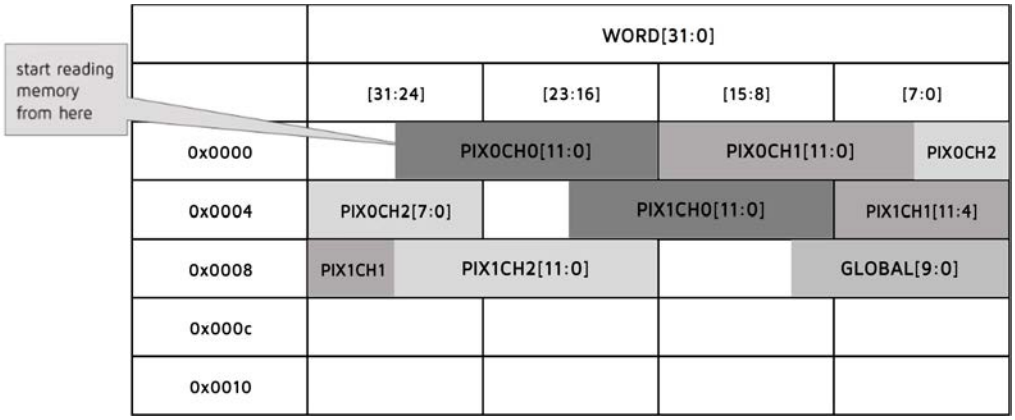


Figure 13-21 Data output sequence of from RZ module for case 1

2. Case2

Assuming that the data stored in memory is aligned at 8 bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 0, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 0. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-22 Data filling in memory for case 2

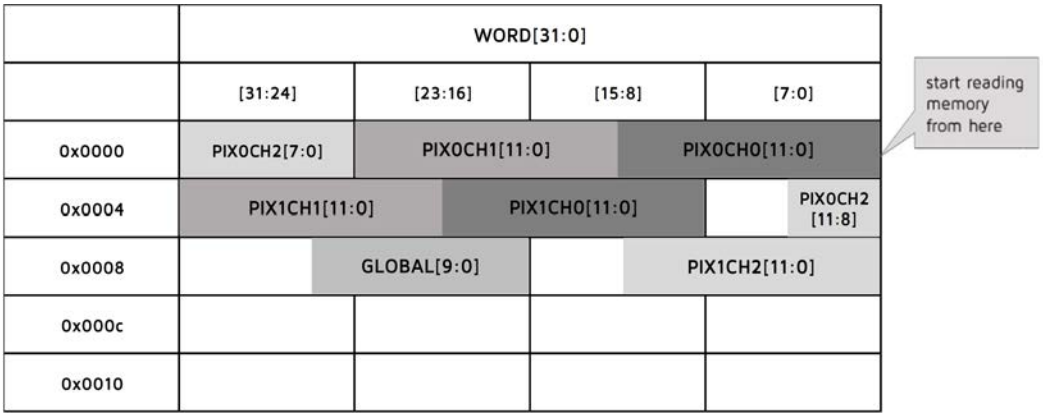


Figure 13-23 Data output sequence of from RZ module for case 2

3. Case3

Assuming that the data stored in memory is aligned at 32 bits, the MSB method is used to read the data in memory; accordingly, register RZ_CTRL[5] is configured to 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 1. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-24 Data filling in memory for case 3

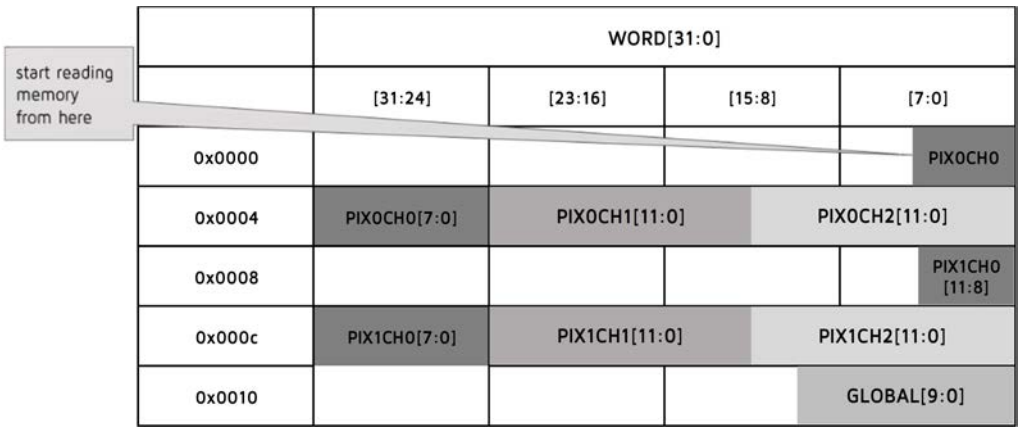


Figure 13-25 Data output sequence of from RZ module for case 3

4. Case4

Assuming that the data stored in memory is aligned at 32 bits and the LSB method is used to read the data in memory; accordingly, register RZ_CTRL[5] is configured to 1 and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 0. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-26 Data filling in memory for case 4

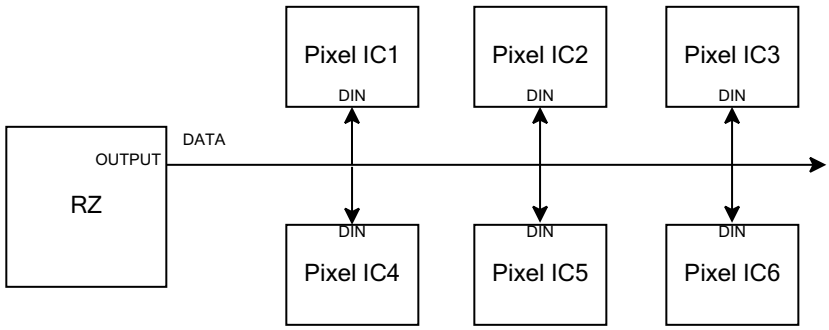
	WORD[31:0]				
	[31:24]	[23:16]	[15:8]	[7:0]	start reading memory from here
0x0000	PIX0CH2[7:0]	PIX0CH1[11:0]		PIX0CH0[11:0]	
0x0004				PIX0CH2 [11:8]	
0x0008	PIX1CH2[7:0]	PIX1CH1[11:0]		PIX1CH0[11:0]	
0x000c				PIX1CH2 [11:8]	
0x0010				GLOBAL[9:0]	

Figure 13-27 Data output sequence of from RZ module for case 4

13.3 Mechanism of Parallel Random Addressing

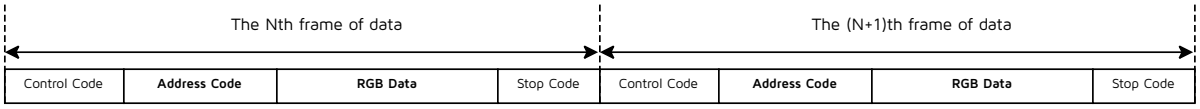
13.3.1 Diagram of parallel connection of Pixel ICs

Figure 13-28 Parallel Connection of Pixel ICs



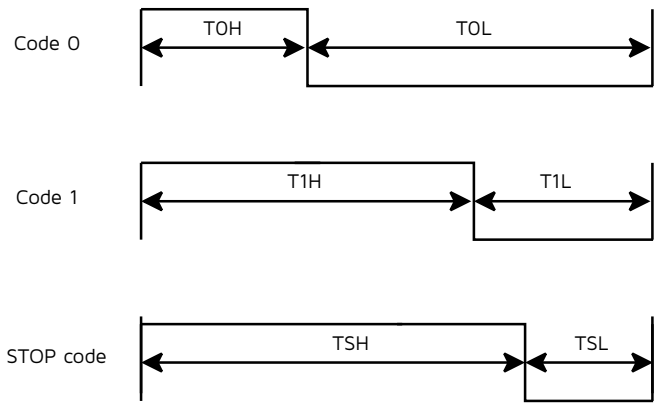
As shown in the figure above, 6 Pixel ICs are working in parallel, all the chips can receive the data sent by the RZ module. When the RZ module drives the chips working in parallel, the data format of each frame sent by the RZ is shown in the figure below. Each frame contains Control Code, Address Code, RGB data and Stop Code, so each chip can recognize whether the current RGB data belongs to it or not by Address Code.

Figure 13-29 Frame data format sent by RZ module



13.3.2 Timing Sequence of RZ

Figure 13-30 Timing Sequence of RZ for Parallel Random Addressing



As shown in the figure above, the RZ code protocol that drives the parallel operation Pixel IC contains three code elements: code 0, code 1 and STOP code, and the three code elements are distinguished by the duration of the high and low levels, which can be configured through registers T0H and T0L for the duration of the high and low levels of the code 0, through registers T1H and T1L for the duration of the high and low levels of the code 1, and through registers TSRH and TSRL for the duration of the high and low levels of the STOP code.

13.3.3 Example of Mechanism of the RZ module to Drive Parallel Pixel ICs

Because there is no global data in the frame data of the parallel mode, the register GLOBAL_DATA_NUM is configured as 0. Also, the sum of the bit numbers of Control Code, Address Code, and RGB data is counted as the PIXEL_DATA_NUM of each chip.Assuming that the RZ module drives a total of two Pixel ICs in parallel. Each frame contains 2 bits of Control Code, 9 bits of Address Code, and 36 bits of RGB data; therefore, register RZ_CTRL1[7] is configured as 1, register PIXEL_NUM is configured as 1, and register PIXEL_DATA_NUM is configured as 46.

1. Case1

Assuming that the data stored in memory is aligned at 8 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 0, and both registers RZ_CTRL[4] and RZ_CTRL[3] are configured as 1. Then, the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-31 Data filling in memory for case 1

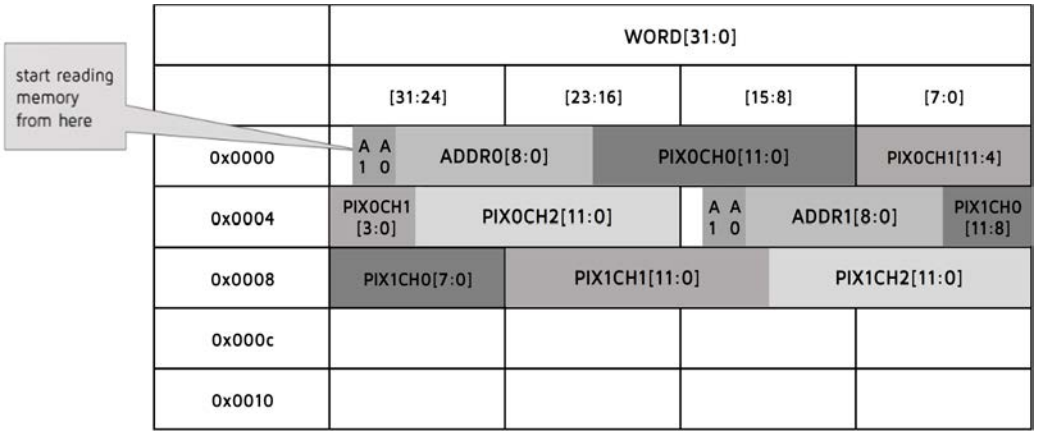
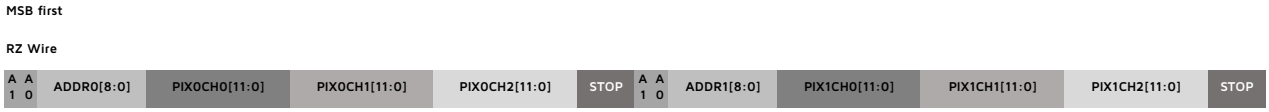


Figure 13-32 Data output sequence of from RZ module for case 1



2. Case2

Assuming that the data stored in memory is aligned at 8 bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 0, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 0. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-33 Data filling in memory for case 2

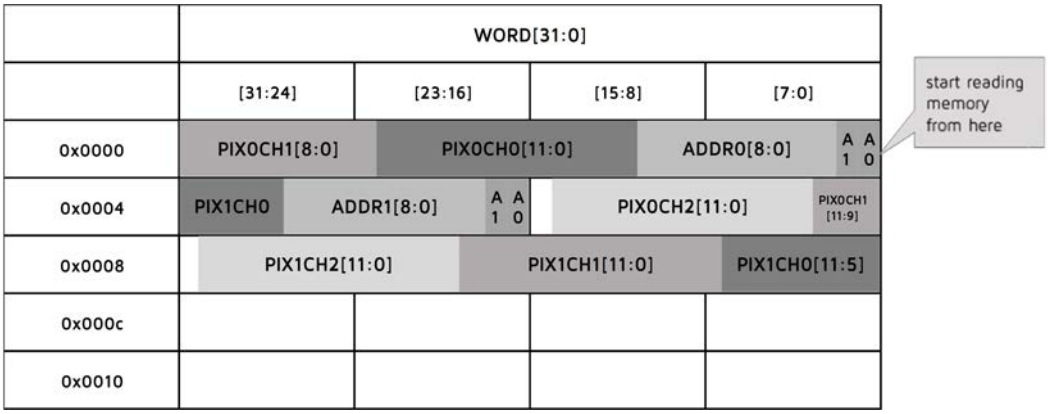
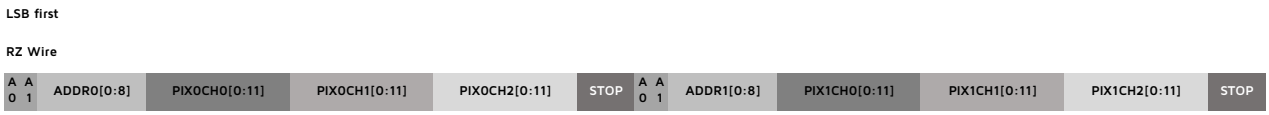


Figure 13-34 Data output sequence of from RZ module for case 2



3. Case3

Assuming that the data stored in memory is aligned at 32 bits, the MSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured to 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured to 1. Then the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-35 Data filling in memory for case 3

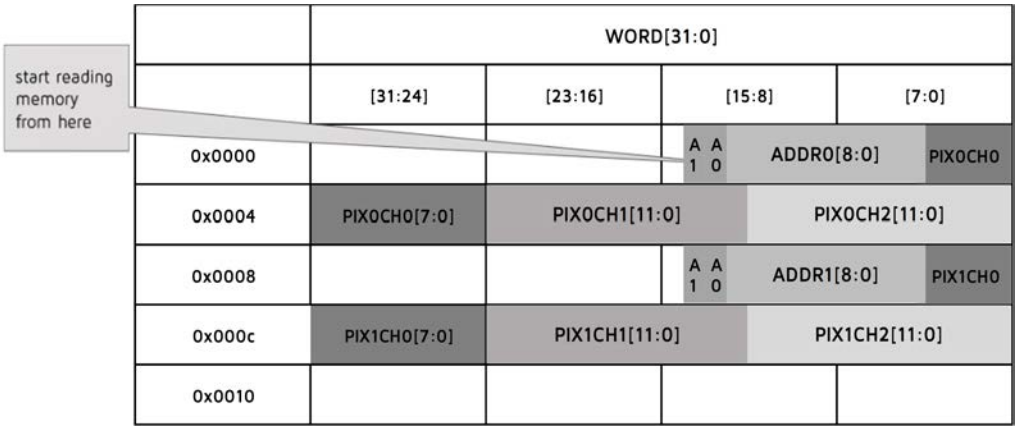
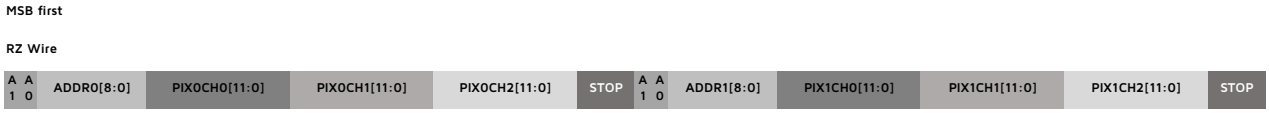


Figure 13-36 Data output sequence of from RZ module for case 3



4. Case4

Assuming that the data stored in memory is aligned at 32 bits, the LSB method is used to read the data in memory; therefore, register RZ_CTRL[5] is configured as 1, and registers RZ_CTRL[4] and RZ_CTRL[3] are both configured as 0. Then, the diagram of the data filling in memory and the timing sequence of the data output from the RZ module are as the following two figures:

Figure 13-37 Data filling in memory for case 4

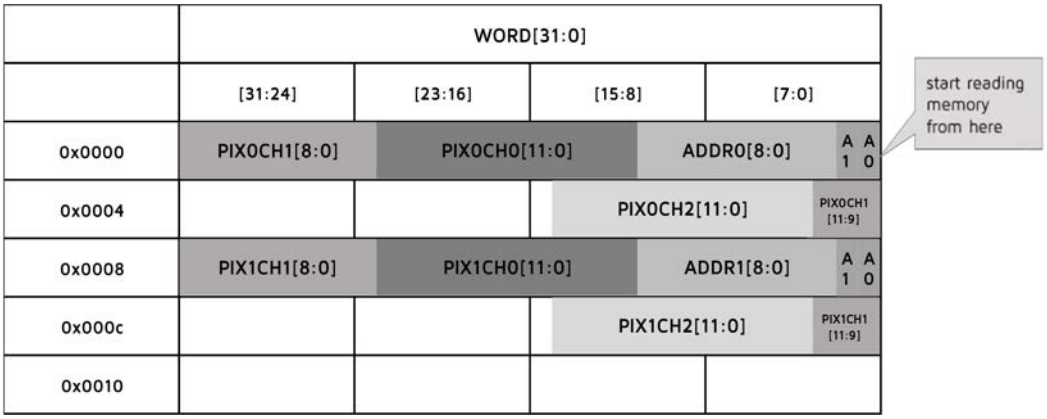
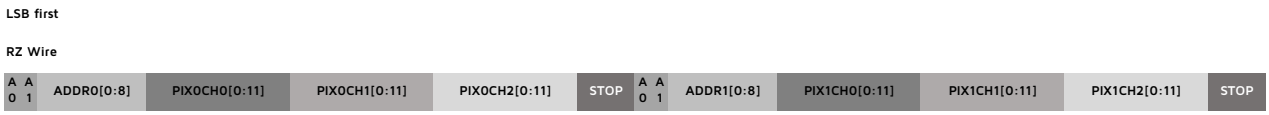


Figure 13-38 Data output sequence of from RZ module for case 4



13.4 Variable Symbol Timing (Jitters on T0L & T1L or T0H & T1H)

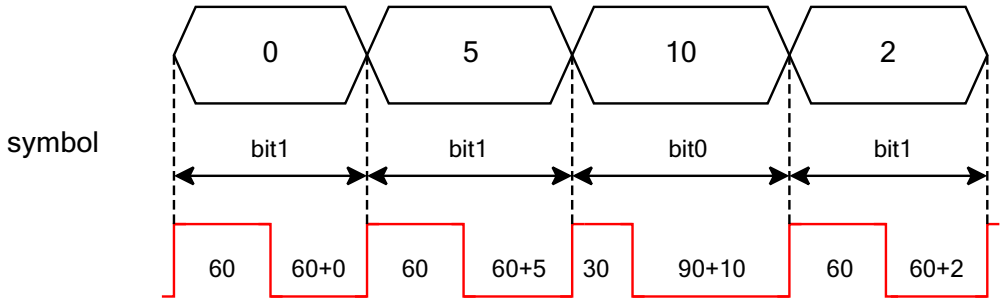
An example of the working mechanism of Variable symbol timing

Jitter_mag is a random number generated by the random number generator, and the range of variation of the random number is selected by configuring register RZ_CTRL2[2:0].

1. Case1

Assuming that the jitter on T0L and T1L is enabled, the random number varies from 0 to 31; therefore, register RZ_CTRL[0] is configured as 1 and register RZ_CTRL2[2:0] is configured as 4. Assuming that registers T0H, T0L, T1H, and T1L are configured to be 30, 90, 60, and 60, respectively, and that the data required to be sent is 4 'b1101, the timing sequence shown below can be obtained.

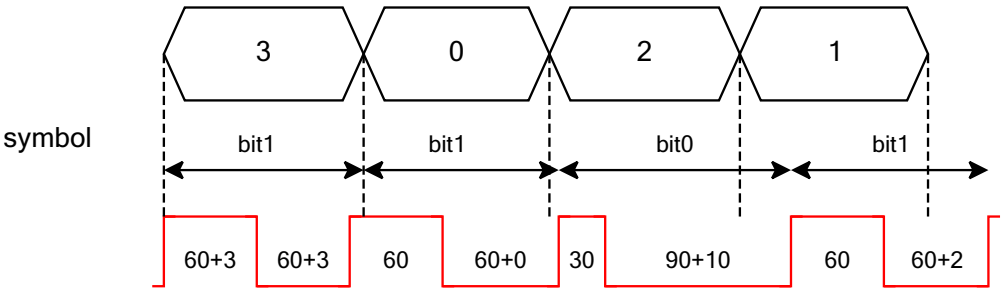
Figure 13-39 Variable Symbol Timing for case 1



2. Case2

Assuming that jitter on T0L and T1L and T0H and T1H is enabled, the random number varies from 0 to 7; therefore, registers RZ_CTRL[0] and RZ_CTRL[1] are both configured as 1, and register RZ_CTRL2[2:0] is configured as 2. Assuming that registers T0H, T0L, T1H, and T1L have been configured to be 30, 90, 60, and 60, respectively. The data to be sent is 4'b1101, the timing sequence shown below can be obtained.

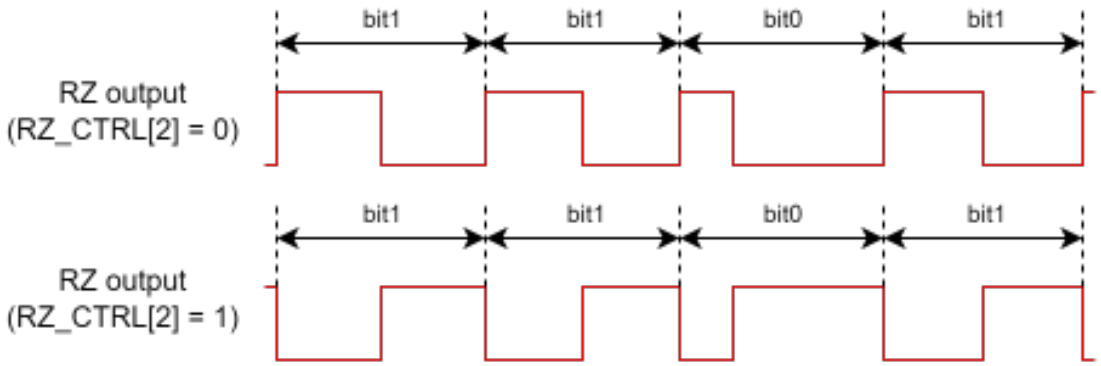
Figure 13-40 Variable Symbol Timing for case 2



13.5 Output Polarity

reg_Jitter_L_en = 1, reg_Jitter_H_en = 1

Figure 13-41 RZ Output Polarity



13.6 RZ Register Description

The RZ related registers are listed in the following table. The base address for the following RZ related registers is 0x80140740.

Table 13-1 RZ Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	RZ_CTRL0	W	[0]: tx_clr, clear tx_fifo [7:1]: reserved	0x00

Address Offset	Name	Type	Description	Default Value
0x01	RZ_CTRL1	R/W	[0]: Jitter_L_en 1: enable jitter on TOL&T1L 0: disable jitter on TOL&T1L [1]: Jitter_H_en 1: enable jitter on TOH&T1H 0: disable jitter on TOH&T1H [2]: pola, output polarity 1: change output polarity 0: not change output polarity [3]: bit_msb 1: MSB of bit in byte first on wire 0: LSB of bit in byte first on wire [4]: big_endian_mode 1: MSB of byte in word first on wire 0: LSB of byte in word first on wire [5] align_32bits_mode 1: each pixel data at 32bits boundary 0: each pixel data at 8bits boundary [6] global_data_mode 1: the global data is after the pixel data of all pixel chips 0: the global data is after the pixel data of each pixel chip [7] addr_mode 1: random addressing mode 0: sequential addressing mode	0x00

Address Offset	Name	Type	Description	Default Value
0x02	RZ_CTRL2	R/W	[2:0] Jitter range, units: Tpclk 0: Jitter range is 0~1 1: Jitter range is 0~3 2: Jitter range is 0~7 3: Jitter range is 0~15 4: Jitter range is 0~31 5: Jitter range is 0~63 6: Jitter range is 0~127 7: Jitter range is 0~255 [5:3]: fifo_lvl fifo_lvl actual range: 1~8, configuration range: 0~7. That is, when fifo_lvl is configured as 0, the actual value of fifo_lvl is 1. The default value of fifo_lvl is 1. [7]: auto_txclr_en	0x00
0x03	RZ_STS	R	[3:0]: tx_buf_cnt [4]: tx_empty [5]: tx_full [6]: RZ_en, RZ enable status	0x00
0x04	T0H_L	R/W	[7:0]: T0H[7:0]	0x00
0x05	T0H_H	R/W	[2:0]: T0H[10:8]	0x00
0x06	T0L_L	R/W	[7:0]: T0H[7:0]	0x00
0x07	T0L_H	R/W	[2:0]: T0H[10:8]	0x00
0x08	T1H_L	R/W	[7:0]: T0H[7:0]	0x00
0x09	T1H_H	R/W	[2:0]: T0H[10:8]	0x00
0x0a	T1L_L	R/W	[7:0]: T0H[7:0]	0x00
0x0b	T1L_H	R/W	[2:0]: T0H[10:8]	0x00
0x0c	TSH_L	R/W	multiplexed as TRH_L [7:0]: TSRH[7:0]	0x00

Address Offset	Name	Type	Description	Default Value
0x0d	TSH_H	R/W	multiplexed as TRH_H [7:0]: TSRH[15:8] When the module works in serial mode, TSRH is T_reset_H; When the module works in parallel mode, TSRH is T_stop_H.	0x00
0x0e	TSL_L	R/W	multiplexed as TRL_L [7:0]: TSRL[7:0]	0x00
0x0f	TSL_H	R/W	multiplexed as TRL_H [7:0]: TSRL[15:8] When the module works in serial mode, TSRL is T_reset_L; When the module works in parallel mode, TSRL is T_stop_L.	0x00
0x10	PIXEL_NUM_L	R/W	[7:0]: pixel_num[7:0]	0x00
0x11	PIXEL_NUM_H	R/W	[0]: pixel_num[8] pixel_num actual range: 1~512, actual configuration range: 0~511. That is: when the number of pixel chips driven is 1, pixel_num is configured as 0, and so on.	0x00
0x12	GLOBAL_DATA_NUM_L	R/W	[7:0]: global_data_num[7:0]	0x00
0x13	GLOBAL_DATA_NUM_H	R/W	[0]: global_data_num[8]	0x00
0x14	PIXEL_DATA_NUM_L	R/W	[7:0]: pixel_data_num[7:0]	0x00
0x15	PIXEL_DATA_NUM_H	R/W	[0]: pixel_data_num[8] pixel_data_bit_num actual range: 1~512, actual configuration range: 0~511. That is: when the number of data bit to be sent by each pixel chip is 1, pixel_data_num is configured as 0, and so on.	0x00

Address Offset	Name	Type	Description	Default Value
0x16	MASK	R/W	[0]: mask_lvl [1]: mask_txdone [2]: mask_error [3]: pem_event_en [7:4]: reserved	0x00
0x17	INT	W1C	[0]: int_lvl, interrupt generated when the number of bytes in the fifo is less than fifo_lvl [1]: int_txdone, interrupt of txdone [2]: int_error, error interrupt generated when DMA response is not timely	0x00
0x18	TX_DAT_0	R/W	[7:0]: tx_rdat[7:0]	0x00
0x19	TX_DAT_1	W	[7:0]: tx_rdat[15:8]	0x00
0x1a	TX_DAT_2	W	[7:0]: tx_rdat[23:16]	0x00
0x1b	TX_DAT_3	W	[7:0]: tx_rdat[31:24]	0x00

14 Quadrature Decoder

The SoC embeds one quadrature decoder (QDEC) which is designed mainly for applications such as wheel. The QDEC implements debounce function to filter out jitter on the two phase inputs, and generates smooth square waves for the two phase.

14.1 Input Pin Selection

The QDEC supports two phase input; each input is selectable from the 8 pins of PortD, PortC, PortB and PortA via setting address 0x42[2:0] (for channel a)/0x43[2:0] (for channel b).

Table 14-1 Input Pin Selection

Address 0x42[2:0]/0x43[2:0]	Pin
0	PA[2]
1	PA[3]
2	PB[6]
3	PB[7]
4	PC[2]
5	PC[3]
6	PD[6]
7	PD[7]

NOTE: *To use corresponding IO as QDEC input pin, it's needed first to enable GPIO function, enable "IE" (1) and disable "OEN" (1) for this IO.*

14.2 Common Mode and Double Accuracy Mode

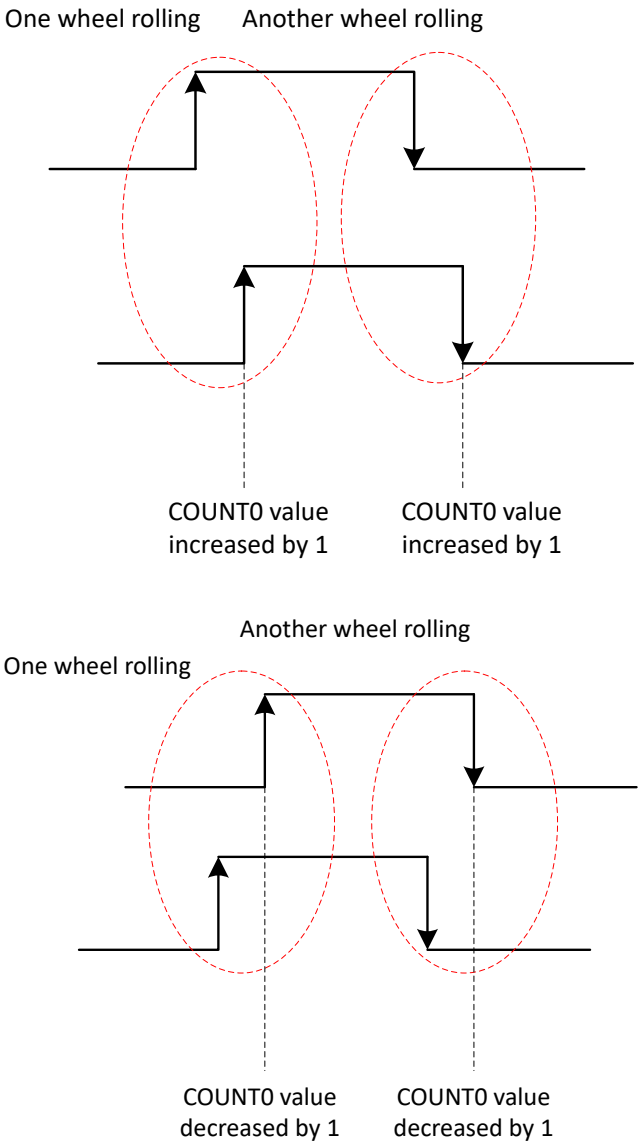
The QDEC embeds an internal hardware counter, which is not connected with bus.

Address 0x80140247 serves to select common mode or double accuracy mode.

For each wheel rolling step, two pulse edges (rising edge or falling edge) are generated.

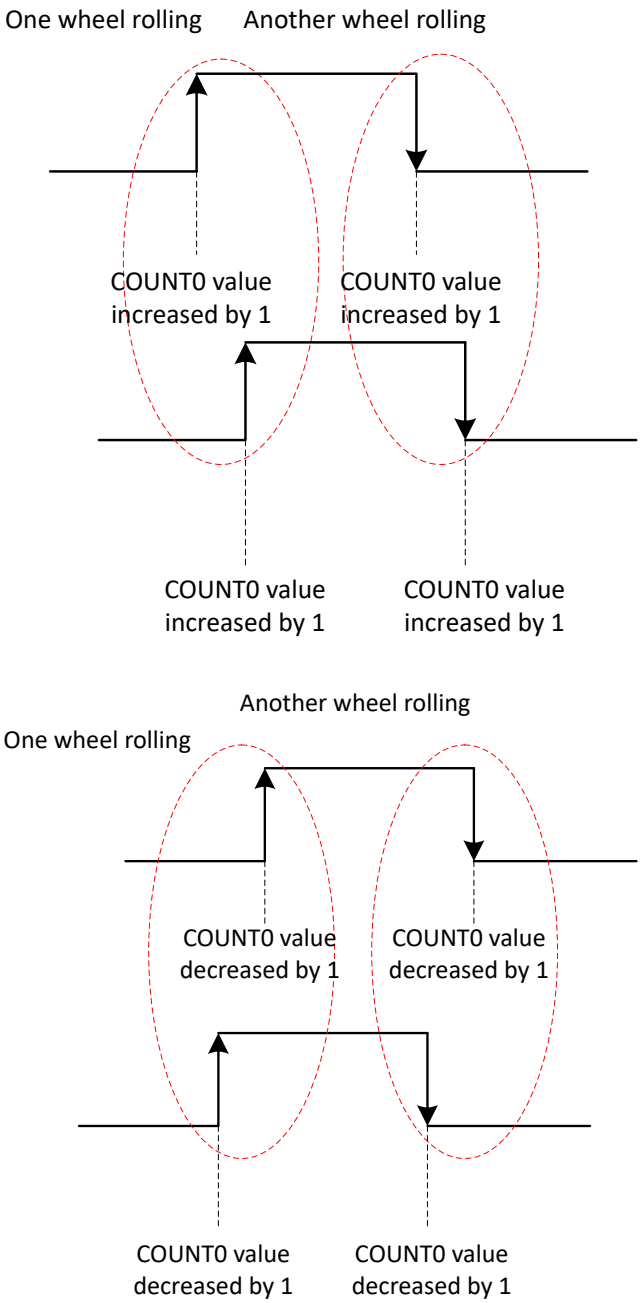
If address 0x80140247 is cleared to select common mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 only when the same rising/falling edges are detected from the two phase signals.

Figure 14-1 Common Mode



If address 0x47[0] is set to 1'b1 to select double accuracy mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 on each rising/falling edge of the two phase signals; the COUNT0 is increased/decreased by 2 for one wheel rolling.

Figure 14-2 Double Accuracy Mode

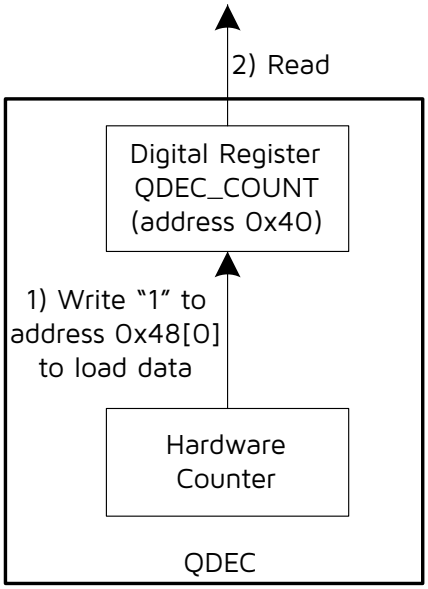


14.3 Read Real Time Counting Value

Neither can Hardware Counter value be read directly via software, nor can the counting value in address 0x40 be updated automatically.

To read real time counting value, first write address 0x48[0] with 1'b1 to load Hardware Counter data into the QDEC_COUNT register, then read address 0x40.

Figure 14-3 Read Real Time Counting Value



14.4 QDEC Reset

Address 0x801401f3[5] serves to reset the QDEC. The QDEC Counter value is cleared to zero.

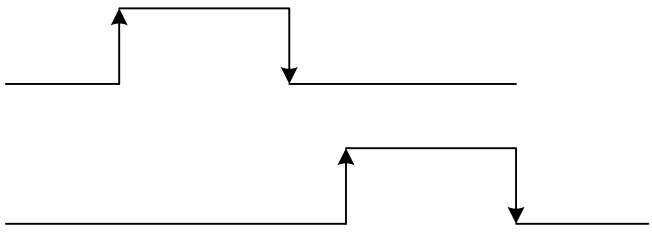
14.5 Other Configuration

The QDEC supports hardware debouncing. Address 0x41[2:0] serves to set filtering window duration. All jitter with period less than the value is filtered out and thus does not trigger count change.

Address 0x41[4] serves to set input signal initial polarity.

Address 0x41[5] serves to enable shuttle mode. Shuttle mode allows non-overlapping two phase signals as shown in the following figure.

Figure 14-4 Shuttle Mode



14.6 Timing Sequence

Figure 14-5 Timing Sequence Chart

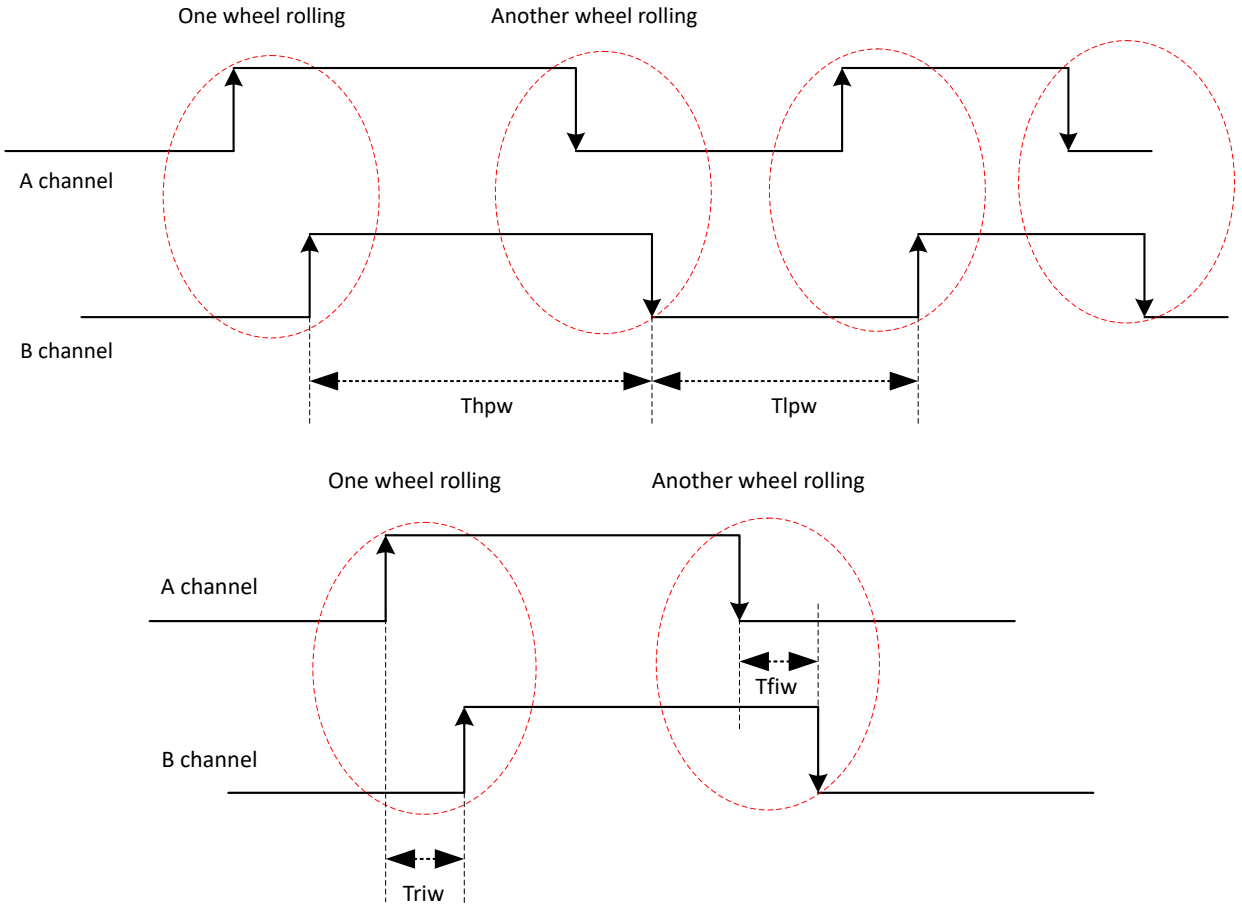


Table 14-2 Timing Interval and Minimum Value

Time Interval	Min Value
Thpw (High-level pulse width)	2^(n+1) *clk_32kHz *3 (n=0x41[2:0])
Tlpw (Low-level pulse width)	2^(n+1) *clk_32kHz *3 (n=0x41[2:0])
Triw (Interval width between two rising edges)	2^(n+1) *clk_32kHz (n=0x41[2:0])
Tfiw (Interval width between two falling edges)	2^(n+1) *clk_32kHz (n=0x41[2:0])

The QDEC module works based on 32 kHz clock to ensure it can work in suspend mode. QDEC module supports debouncing function, and any signal with width lower than the threshold (i.e. "2^(n+1) *clk_32kHz *3 (n=0x41[2:0])) is regarded as jitter. Therefore, effective signals input from Channel A and B should contain high/low level with width Thpw/Tlpw more than the threshold. The 2^n *clk_32kHz clock is used to synchronize input signal of QDEC module, so the interval between two adjacent rising/falling edges from Channel A and B, which are marked as Triw and Tfiw, should exceed "2^(n+1) *clk_32kHz".

Only when the timing requirements above are met, can QDEC module recognize wheel rolling times correctly.

14.7 QDEC Register Description

The QDEC related registers are listed in the following table. The base address for the following registers is 0x80140240.

Table 14-3 QDEC Related Registers

Address offset	Name	Type	Description	Reset Value
0x00	QDEC_COUNT0	R	QDEC Counting value (read to clear): Pulse edge number	0x00
0x01	QDEC_DBNTIME	RW	[5]: shuttle mode 1 - enable shuttle mode [4]: pola, input signal pola 0 - no signal is low, 1 - no signal is high [2:0]: filter time (can filter 2^n *clk_32k*2 width deglitch)	0x00
0x02	QDEC_CHANNEL_A0	RW	[2:0]: QDEC input pin select for channel A, choose 1 of 8 pins for input channel A 7~0: {PD[7:6], PC[3:2], PB[7:6], PA[3:2]}	0x00
0x03	QDEC_CHANNEL_B0	RW	[2:0]: QDEC input pin select for channel B, choose 1 of 8 pins for input channel B 7~0: {PD[7:6], PC[3:2], PB[7:6], PA[3:2]}	0x01
0x04	QDEC_MASK	RW	[0]Interrupt mask 1: enable 0: mask	0x00
0x05	QDEC_INT0	RW	[0]Interrupt flag Write 1 to clear	0x00
0x06	QDEC_READ	R	[7:0] dat_o	0x00
0x07	QDEC_DOUBLE0	RW	[0]: Enable double accuracy mode	0x01
0x08	QDEC_COUNT0_RELOAD	RW	[0]: write 1 to load data When load completes it is 0.	0x00

15 Peripheral Event Matrix (PEM)

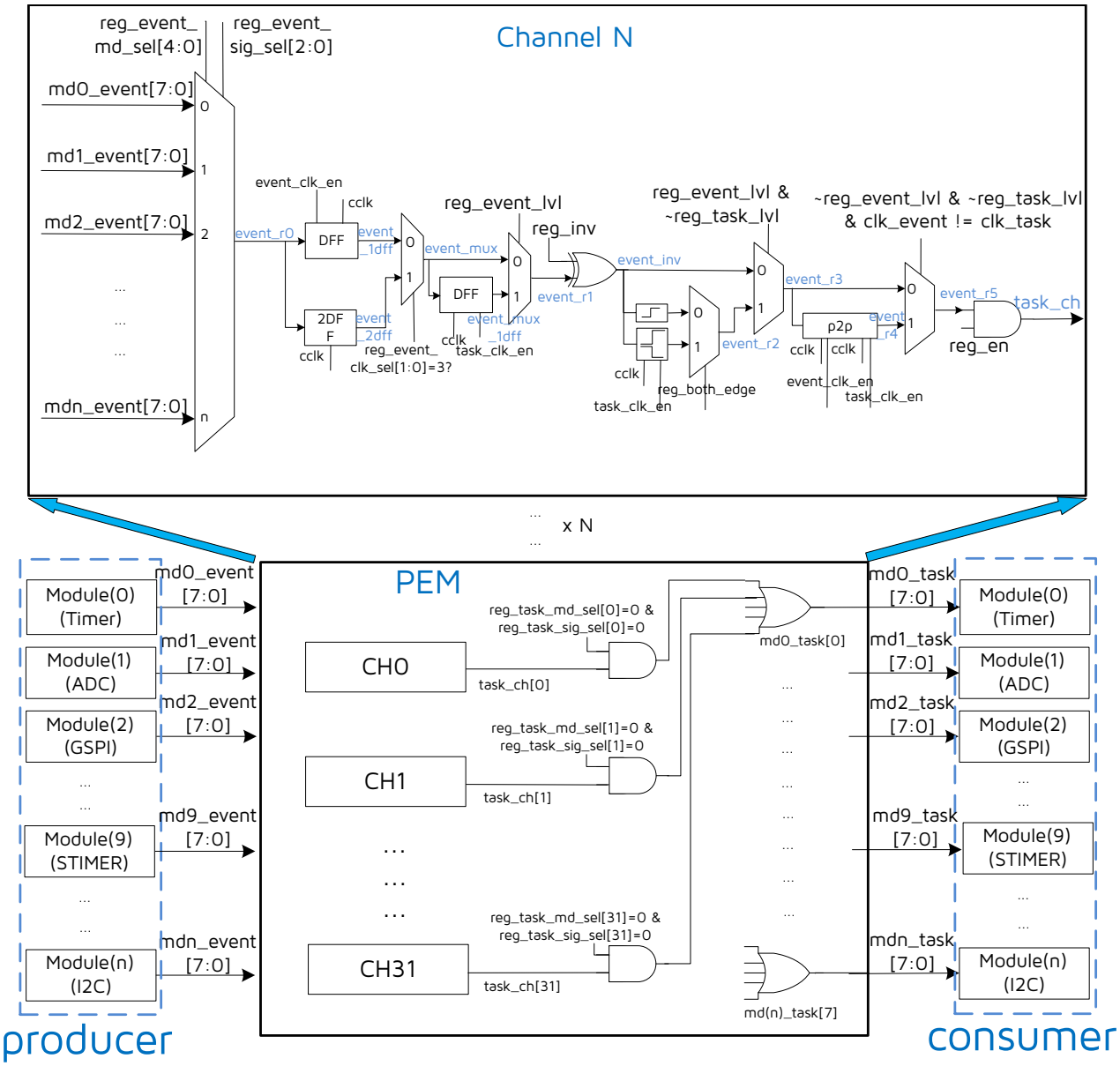
15.1 Introduction of PEM

The SoC supports the PEM (Peripheral Event Matrix) function which is to realize the interconnection between the peripherals, any peripheral A's event signal (similar to the interrupt signal) is routed to any peripheral B's task input, the peripheral B treats the task signal as enable or trigger signal.

15.2 Block Diagram of PEM

The block diagram of PEM is shown as below.

Figure 15-1 Block Diagram of PEM



15.3 PEM Function Description

The PEM routes the event signal of peripheral A to the task signal of peripheral B, and the peripheral B treats the task signal as enable or trigger signal.

- Event signal: comes from the peripheral, similar as interrupt signal.
- Task signal: it can choose any one of the event signals, the peripheral treats task signal as enable or trigger signal.

In the block diagram above, each peripheral can have multiple event and task signals, the peripheral that generates the event is called producer, the peripheral that receives the task is called consumer, and the same peripheral can be both producer and consumer.

By using multiple channels, different events can be routed to the same task and the same event can be routed to different tasks.

15.4 Event Task List

The Event and Task for PEM is listed as below.

Table 15-1 Event Task List

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
0	MSPI		2	rxf_threshold	level	hclk	0	trigger SPI transmission	pulse	hclk
			3	txf_threshold	level	hclk				
			4	trans_done	pulse	hclk				
1	LSPI	0	0	rxf_overrun	pulse	hclk	0	trigger SPI transmission	pulse	hclk
			1	txf_underrun	pulse	hclk				
			2	rxf_threshold	level	hclk				
			3	txf_threshold	level	hclk				
			4	trans_done	pulse	hclk				
			5	slave_cmd	pulse	hclk				
			6	lcd_line_done	pulse	hclk				
			7	lcd_line_lvl	pulse	hclk				
		1	0	lcd_frame_done	pulse	hclk				

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
2	GSPI		0	rxf_overrun	pulse	hclk	0	trigger SPI transmission	pulse	hclk
			1	txf_underrun	pulse	hclk				
			2	rxf_threshold	level	hclk				
			3	txf_threshold	level	hclk				
			4	trans_done	pulse	hclk				
			5	slave_cmd	pulse	hclk				
3	OSR_IP		0	pke_irq	level	hclk				
			1	trng_irq	level	hclk				
			2	hash_irq	level	hclk				
			3	ske_irq	level	hclk				
			4	chacha20_irq	level	hclk				

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
4	GPIO	0	0	pa0_input	level	aclk	0	pa0_toggle	pulse	pclk
			1	pa1_input	level	aclk	1	pa1_toggle	pulse	pclk
			2	pa2_input	level	aclk	2	pa2_toggle	pulse	pclk
			3	pa3_input	level	aclk	3	pa3_toggle	pulse	pclk
			4	pa4_input	level	aclk	4	pa4_toggle	pulse	pclk
			5	pa5_input	level	aclk	5	pa5_toggle	pulse	pclk
			6	pa6_input	level	aclk	6	pa6_toggle	pulse	pclk
			7	pa7_input	level	aclk	7	pa7_toggle	pulse	pclk
		1	0	pb0_input	level	aclk	0	pb0_toggle	pulse	pclk
			1	pb1_input	level	aclk	1	pb1_toggle	pulse	pclk
			2	pb2_input	level	aclk	2	pb2_toggle	pulse	pclk
			3	pb3_input	level	aclk	3	pb3_toggle	pulse	pclk
			4	pb4_input	level	aclk	4	pb4_toggle	pulse	pclk
			5	pb5_input	level	aclk	5	pb5_toggle	pulse	pclk
			6	pb6_input	level	aclk	6	pb6_toggle	pulse	pclk
			7	pb7_input	level	aclk	7	pb7_toggle	pulse	pclk
		2	0	pc0_input	level	aclk	0	pc0_toggle	pulse	pclk
			1	pc1_input	level	aclk	1	pc1_toggle	pulse	pclk
			2	pc2_input	level	aclk	2	pc2_toggle	pulse	pclk
			3	pc3_input	level	aclk	3	pc3_toggle	pulse	pclk
			4	pc4_input	level	aclk	4	pc4_toggle	pulse	pclk
			5	pc5_input	level	aclk	5	pc5_toggle	pulse	pclk
			6	pc6_input	level	aclk	6	pc6_toggle	pulse	pclk
			7	pc7_input	level	aclk	7	pc7_toggle	pulse	pclk

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
4	GPIO	3	0	pd0_input	level	aclk	0	pd0_toggle	pulse	pclk
			1	pd1_input	level	aclk	1	pd1_toggle	pulse	pclk
			2	pd2_input	level	aclk	2	pd2_toggle	pulse	pclk
			3	pd3_input	level	aclk	3	pd3_toggle	pulse	pclk
			4	pd4_input	level	aclk	4	pd4_toggle	pulse	pclk
			5	pd5_input	level	aclk	5	pd5_toggle	pulse	pclk
			6	pd6_input	level	aclk	6	pd6_toggle	pulse	pclk
			7	pd7_input	level	aclk	7	pd7_toggle	pulse	pclk
		4	0	pe0_input	level	aclk	0	pe0_toggle	pulse	pclk
			1	pe1_input	level	aclk	1	pe1_toggle	pulse	pclk
			2	pe2_input	level	aclk	2	pe2_toggle	pulse	pclk
			3	pe3_input	level	aclk	3	pe3_toggle	pulse	pclk
			4	pe4_input	level	aclk	4	pe4_toggle	pulse	pclk
			5	pe5_input	level	aclk	5	pe5_toggle	pulse	pclk
			6	pe6_input	level	aclk	6	pe6_toggle	pulse	pclk
			7	pe7_input	level	aclk	7	pe7_toggle	pulse	pclk
		5	0	pf0_input	level	aclk	0	pf0_toggle	pulse	pclk
			1	pf1_input	level	aclk	1	pf1_toggle	pulse	pclk
			2	pf2_input	level	aclk	2	pf2_toggle	pulse	pclk
			3	pf3_input	level	aclk	3	pf3_toggle	pulse	pclk
			4	pf4_input	level	aclk	4	pf4_toggle	pulse	pclk
			5	pf5_input	level	aclk	5	pf5_toggle	pulse	pclk
			6	pf6_input	level	aclk	6	pf6_toggle	pulse	pclk
			7	pf7_input	level	aclk	7	pf7_toggle	pulse	pclk

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
4	GPIO	6	0	pg0_input	level	aclk	0	pg0_toggle	pulse	pclk
			1	pg1_input	level	aclk	1	pg1_toggle	pulse	pclk
			2	pg2_input	level	aclk	2	pg2_toggle	pulse	pclk
			3	pg3_input	level	aclk	3	pg3_toggle	pulse	pclk
			4	pg4_input	level	aclk	4	pg4_toggle	pulse	pclk
			5	pg5_input	level	aclk	5	pg5_toggle	pulse	pclk
		7	0	gpio_irq	level	aclk				
			1	gpio2risc0	level	aclk				
			2	gpio2risc1	level	aclk				
		8	0	gpio_irq_group0	level	aclk				
			1	gpio_irq_group1	level	aclk				
			2	gpio_irq_group2	level	aclk				
			3	gpio_irq_group3	level	aclk				
			4	gpio_irq_group4	level	aclk				
			5	gpio_irq_group5	level	aclk				
			6	gpio_irq_group6	level	aclk				
			7	gpio_irq_group7	level	aclk				
5	DMA	0	0	ch0_tc	pulse	hclk	0	ch0_en	pulse	hclk
			1	ch1_tc	pulse	hclk	1	ch1_en	pulse	hclk
			2	ch2_tc	pulse	hclk	2	ch2_en	pulse	hclk
			3	ch3_tc	pulse	hclk	3	ch3_en	pulse	hclk
			4	ch4_tc	pulse	hclk	4	ch4_en	pulse	hclk
			5	ch5_tc	pulse	hclk	5	ch5_en	pulse	hclk
			6	ch6_tc	pulse	hclk	6	ch6_en	pulse	hclk
			7	ch7_tc	pulse	hclk	7	ch7_en	pulse	hclk

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
5	DMA	1	0	ch0_abt	pulse	hclk				
			1	ch1_abt	pulse	hclk				
			2	ch2_abt	pulse	hclk				
			3	ch3_abt	pulse	hclk				
			4	ch4_abt	pulse	hclk				
			5	ch5_abt	pulse	hclk				
			6	ch6_abt	pulse	hclk				
			7	ch7_abt	pulse	hclk				
		2	0	ch0_err	pulse	hclk				
			1	ch1_err	pulse	hclk				
			2	ch2_err	pulse	hclk				
			3	ch3_err	pulse	hclk				
			4	ch4_err	pulse	hclk				
			5	ch5_err	pulse	hclk				
			6	ch6_err	pulse	hclk				
			7	ch7_err	pulse	hclk				
		3	0	ch0_wbufov	pulse	hclk				
			1	ch1_wbufov	pulse	hclk				
			2	ch2_wbufov	pulse	hclk				
			3	ch3_wbufov	pulse	hclk				
			4	ch4_wbufov	pulse	hclk				
			5	ch5_wbufov	pulse	hclk				
			6	ch6_wbufov	pulse	hclk				
			7	ch7_wbufov	pulse	hclk				

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
5	DMA	4	0	ch0_pdcyc	pulse	hclk				
			1	ch1_pdcyc	pulse	hclk				
			2	ch2_pdcyc	pulse	hclk				
			3	ch3_pdcyc	pulse	hclk				
			4	ch4_pdcyc	pulse	hclk				
			5	ch5_pdcyc	pulse	hclk				
			6	ch6_pdcyc	pulse	hclk				
			7	ch7_pdcyc	pulse	hclk				
6	MISC		0	qdec_int_pos						
			1	qdec_wakeup	level	aclk				
			7	pm_irq	level	aclk				
7	CPU		0	wfi_mode	level	cclk				
8	Timer		0	irq_mode0	pulse	pclk	0	timer0_en	pulse	pclk
			1	irq_capt0	pulse	pclk	1	timer0_disable	pulse	pclk
			2	irq_comp0	pulse	pclk	2	timer1_en	pulse	pclk
			3	irq_mode1	pulse	pclk	3	timer1_disable	pulse	pclk
			4	irq_capt1	pulse	pclk	4	wd_en	pulse	pclk
			5	irq_comp1	pulse	pclk	5	wd_disable	pulse	pclk
							6	capt0	pulse	pclk
							7	capt1	pulse	pclk
9	STimer		0	irq_trig_pos	pulse	pclk	0	stimer_en	pulse	pclk
			1	irq_cal_tgl_pul	pulse	pclk	1	stimer_disable	pulse	pclk
			2	irq_capt	pulse	pclk	2	capt	pulse	pclk
			3	irq_ov	pulse	pclk				
10	SAR_ADC		0	rx_threshold	level	pclk	0	sigle adc_trig	pulse	pclk
			1	rx_data_fifo_wr	pulse	pclk				

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
11	Audio		0	txfifo_irq	pulse	pclk	0	i2s0_en/ sdm_en	pulse	pclk
			1	rxfifo_irq	pulse	pclk	1	i2s0_disable/ sdm_disable	pulse	pclk
			2	txfifo_th_irq	pulse	pclk	2	i2s1_en	pulse	pclk
			3	rxfifo_th_irq	pulse	pclk	3	i2s1_disable	pulse	pclk
							4	i2s2_en	pulse	pclk
							5	i2s2_disable	pulse	pclk
							6	codec_en	pulse	pclk
							7	codec_disable	pulse	pclk
12	IR_learn		0	irq_high	pulse	pclk	0	ir_learn_en	pulse	pclk
			1	irq_cycle	pulse	pclk	1	ir_learn_disable	pulse	pclk
			2	irq_timeout	pulse	pclk				
			3	irq_rxbuf	pulse	pclk				
13	PWM_0		0	pwm0_period_start	pulse	pclk	0	pwm0_en	pulse	pclk
			1	pwm1_period_start	pulse	pclk	1	pwm1_en	pulse	pclk
			2	pwm2_period_start	pulse	pclk	2	pwm2_en	pulse	pclk
			3	pwm3_period_start	pulse	pclk	3	pwm3_en	pulse	pclk
			4	pwm4_period_start	pulse	pclk	4	pwm4_en	pulse	pclk
			5	pwm5_period_start	pulse	pclk	5	pwm5_en	pulse	pclk
			6	pwm6_period_start	pulse	pclk	6	pwm6_en	pulse	pclk

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
14	PWM_1	0	0	pwm0_cycdone	pulse	pclk	0	pwm0_disable	pulse	pclk
			1	pwm1_cycdone	pulse	pclk	1	pwm1_disable	pulse	pclk
			2	pwm2_cycdone	pulse	pclk	2	pwm2_disable	pulse	pclk
			3	pwm3_cycdone	pulse	pclk	3	pwm3_disable	pulse	pclk
			4	pwm4_cycdone	pulse	pclk	4	pwm4_disable	pulse	pclk
			5	pwm5_cycdone	pulse	pclk	5	pwm5_disable	pulse	pclk
			6	pwm6_cycdone	pulse	pclk	6	pwm6_disable	pulse	pclk
		1	0	pwm0_done	pulse	pclk				
			1	pwm0_fifo_done	pulse	pclk				
			2	pwm0_lvl	pulse	pclk				
15	RZ		0	irq_txbuf	pulse	pclk				
			1	irq_txdone	pulse	pclk				
			2	tx_empty	level	pclk				
			3	seten	pulse	pclk				
16	Algm		0	rx_buf_irq	level	pclk	0	algm_en	pulse	pclk
			1	tx_buf_irq	level	pclk	1	rx_fifo_clr	pulse	pclk
			2	rx_done	level	pclk	2	tx_fifo_clr	pulse	pclk
			3	tx_done	level	pclk				
			4	tx_empty	level	pclk				
17	UART0		0	rx_buf_irq	level	pclk	0	uart_en	pulse	pclk
			1	tx_buf_irq	level	pclk	1	uart_disable	pulse	pclk
			2	rx_done	level	pclk	2	rx_fifo_clr	pulse	pclk
			3	tx_done	level	pclk	3	tx_fifo_clr	pulse	pclk
			4	rx_err	level	pclk	4	uart div cnt clr	pulse	pclk

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
18	UART1		0	rx_buf_irq	level	pclk	0	uart_en	pulse	pclk
			1	tx_buf_irq	level	pclk	1	uart_disable	pulse	pclk
			2	rx_done	level	pclk	2	rx_fifo_clr	pulse	pclk
			3	tx_done	level	pclk	3	tx_fifo_clr	pulse	pclk
			4	rx_err	level	pclk	4	uart div cnt clr	pulse	pclk
19	UART2		0	rx_buf_irq	level	pclk	0	uart_en	pulse	pclk
			1	tx_buf_irq	level	pclk	1	uart_disable	pulse	pclk
			2	rx_done	level	pclk	2	rx_fifo_clr	pulse	pclk
			3	tx_done	level	pclk	3	tx_fifo_clr	pulse	pclk
			4	rx_err	level	pclk	4	uart div cnt clr	pulse	pclk
20	I2C	0	0	rx_buf_irq	level	pclk	0	i2c_master_en	pulse	pclk
			1	tx_buf_irq	level	pclk	1	i2c_master_disable	pulse	pclk
			2	rx_done	level	pclk	2	i2c_slave_en	pulse	pclk
			3	tx_done	level	pclk	3	i2c_slave_disable	pulse	pclk
			4	rx_end	level	pclk	4	rx_fifo_clr	pulse	pclk
			5	tx_end	level	pclk	5	tx_fifo_clr	pulse	pclk
			6	trx_stop	level	pclk				
			7	trx_start	level	pclk				
		1	0	nak_irq	level	pclk				
			1	ss_rw_irq	level	pclk				
			2	ss_scl_irq	level	pclk				

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
22	USB	0	0	reset	level	hclk	0	edps0_ack(8)	pulse	hclk
			1	250us	level	hclk	1	edps1_ack	pulse	hclk
			2	suspend	level	hclk	2	edps2_ack	pulse	hclk
			3	sof	level	hclk	3	edps3_ack	pulse	hclk
			4	setup	level	hclk	4	edps4_ack	pulse	hclk
			5	data	level	hclk	5	edps5_ack	pulse	hclk
			6	status	level	hclk	6	edps6_ack	pulse	hclk
			7	setintf	level	hclk	7	edps7_ack	pulse	hclk
		1	0	edps0_data_irq(8)	level	hclk				
			1	edps1_data_irq	level	hclk				
			2	edps2_data_irq	level	hclk				
			3	edps3_data_irq	level	hclk				
			4	edps4_data_irq	level	hclk				
			5	edps5_data_irq	level	hclk				
			6	edps6_data_irq	level	hclk				
			7	edps7_data_irq	level	hclk				
23	ZB (Zigbee Baseband)	0	0	zb_rx	level	hclk	0	hit_stimer	pulse	hclk
			1	zb_tx	level	hclk				
			2	rx_timeout	level	hclk				
			3	rx_fifo_full	level	hclk				
			4	rx_crc2	level	hclk				
			5	cmd_done	level	hclk				
			6	fsm_timeout	level	hclk				
			7	tx_retrycnt	level	hclk				

No.	Module	Sub Sel.	Event No.	Event	Type	Clock	Task No.	Task	Type	Clock
23	ZB	1	0	tx_ds	level	hclk				
			1	rx_dr	level	hclk				
			2	rx_fst_timeout	level	hclk				
			3	rx_invld_pid	level	hclk				
			4	s_tx_timeout	level	hclk				
			5	wifi_deny	level	hclk				
			6	supp_of	level	hclk				
			7	rxdma_of	level	hclk				
		2	0	tr_turnaround	level	hclk				
			1	rxcrypt_error	level	hclk				
			2	txcrypt_err	level	hclk				
			3	hit_sync	level	hclk				
			4	header_done	level	hclk				
			5	pkt_unmatch	level	hclk				
			6	pkt_match_irq	level	hclk				
			7	zb_freq_hop	level	hclk				
		3	0	zb_freq_fixed	level	hclk				
			1	fcal_done	level	hclk				
			2	cal_done	level	hclk				
		4	0	ch0_status	pulse	hclk	0	ch0_en	pulse	hclk
			1	ch1_status	pulse	hclk	1	ch1_en	pulse	hclk
		5	0	irq_trig_pos	pulse	hclk	0	stimer_en	pulse	hclk
			1	irq_cal_tgl_pul	pulse	hclk	1	stimer_disable	pulse	hclk

15.5 PEM Register Description

The PEM related registers are listed in the following table. The base address for the following PEM related registers is 0x80142000.

Table 15-2 PEM Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	PEM_CH0_CTRL0	RW	[4:0] event_module_sel	0x00
0x01	PEM_CH0_CTRL1	RW	[4:0] task_module_sel	0x00
0x02	PEM_CH0_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x03	PEM_CH0_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x04	PEM_CH1_CTRL0	RW	[4:0] event_module_sel	0x00
0x05	PEM_CH1_CTRL1	RW	[4:0] task_module_sel	0x00
0x06	PEM_CH1_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x07	PEM_CH1_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x08	PEM_CH2_CTRL0	RW	[4:0] event_module_sel	0x00
0x09	PEM_CH2_CTRL1	RW	[4:0] task_module_sel	0x00
0x0a	PEM_CH2_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00

Address Offset	Name	Type	Description	Default Value
0x0b	PEM_CH2_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x0c	PEM_CH3_CTRL0	RW	[4:0] event_module_sel	0x00
0x0d	PEM_CH3_CTRL1	RW	[4:0] task_module_sel	0x00
0x0e	PEM_CH3_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x0f	PEM_CH3_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x10	PEM_CH4_CTRL0	RW	[4:0] event_module_sel	0x00
0x11	PEM_CH4_CTRL1	RW	[4:0] task_module_sel	0x00
0x12	PEM_CH4_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x13	PEM_CH4_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x14	PEM_CH5_CTRL0	RW	[4:0] event_module_sel	0x00
0x15	PEM_CH5_CTRL1	RW	[4:0] task_module_sel	0x00

Address Offset	Name	Type	Description	Default Value
0x16	PEM_CH5_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x17	PEM_CH5_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x18	PEM_CH6_CTRL0	RW	[4:0] event_module_sel	0x00
0x19	PEM_CH6_CTRL1	RW	[4:0] task_module_sel	0x00
0x1a	PEM_CH6_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x1b	PEM_CH6_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x1c	PEM_CH7_CTRL0	RW	[4:0] event_module_sel	0x00
0x1d	PEM_CH7_CTRL1	RW	[4:0] task_module_sel	0x00
0x1e	PEM_CH7_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x1f	PEM_CH7_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00

Address Offset	Name	Type	Description	Default Value
0x20	PEM_CH8_CTRL0	RW	[4:0] event_module_sel	0x00
0x21	PEM_CH8_CTRL1	RW	[4:0] task_module_sel	0x00
0x22	PEM_CH8_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x23	PEM_CH8_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x24	PEM_CH9_CTRL0	RW	[4:0] event_module_sel	0x00
0x25	PEM_CH9_CTRL1	RW	[4:0] task_module_sel	0x00
0x26	PEM_CH9_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x27	PEM_CH9_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x28	PEM_CH10_CTRL0	RW	[4:0] event_module_sel	0x00
0x29	PEM_CH10_CTRL1	RW	[4:0] task_module_sel	0x00
0x2a	PEM_CH10_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00

Address Offset	Name	Type	Description	Default Value
0x2b	PEM_CH10_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x2c	PEM_CH11_CTRL0	RW	[4:0] event_module_sel	0x00
0x2d	PEM_CH11_CTRL1	RW	[4:0] task_module_sel	0x00
0x2e	PEM_CH11_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x2f	PEM_CH11_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x30	PEM_CH12_CTRL0	RW	[4:0] event_module_sel	0x00
0x31	PEM_CH12_CTRL1	RW	[4:0] task_module_sel	0x00
0x32	PEM_CH12_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x33	PEM_CH12_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x34	PEM_CH13_CTRL0	RW	[4:0] event_module_sel	0x00
0x35	PEM_CH13_CTRL1	RW	[4:0] task_module_sel	0x00

Address Offset	Name	Type	Description	Default Value
0x36	PEM_CH13_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x37	PEM_CH13_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x38	PEM_CH14_CTRL0	RW	[4:0] event_module_sel	0x00
0x39	PEM_CH14_CTRL1	RW	[4:0] task_module_sel	0x00
0x3a	PEM_CH14_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x3b	PEM_CH14_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x3c	PEM_CH15_CTRL0	RW	[4:0] event_module_sel	0x00
0x3d	PEM_CH15_CTRL1	RW	[4:0] task_module_sel	0x00
0x3e	PEM_CH15_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x3f	PEM_CH15_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00

Address Offset	Name	Type	Description	Default Value
0x40	PEM_CH16_CTRL0	RW	[4:0] event_module_sel	0x00
0x41	PEM_CH16_CTRL1	RW	[4:0] task_module_sel	0x00
0x42	PEM_CH16_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x43	PEM_CH16_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x44	PEM_CH17_CTRL0	RW	[4:0] event_module_sel	0x00
0x45	PEM_CH17_CTRL1	RW	[4:0] task_module_sel	0x00
0x46	PEM_CH17_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x47	PEM_CH17_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x48	PEM_CH18_CTRL0	RW	[4:0] event_module_sel	0x00
0x49	PEM_CH18_CTRL1	RW	[4:0] task_module_sel	0x00
0x4a	PEM_CH18_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00

Address Offset	Name	Type	Description	Default Value
0x4b	PEM_CH18_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x4c	PEM_CH19_CTRL0	RW	[4:0] event_module_sel	0x00
0x4d	PEM_CH19_CTRL1	RW	[4:0] task_module_sel	0x00
0x4e	PEM_CH19_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x4f	PEM_CH19_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x50	PEM_CH20_CTRL0	RW	[4:0] event_module_sel	0x00
0x51	PEM_CH20_CTRL1	RW	[4:0] task_module_sel	0x00
0x52	PEM_CH20_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x53	PEM_CH20_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x54	PEM_CH21_CTRL0	RW	[4:0] event_module_sel	0x00
0x55	PEM_CH21_CTRL1	RW	[4:0] task_module_sel	0x00

Address Offset	Name	Type	Description	Default Value
0x56	PEM_CH21_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x57	PEM_CH21_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x58	PEM_CH22_CTRL0	RW	[4:0] event_module_sel	0x00
0x59	PEM_CH22_CTRL1	RW	[4:0] task_module_sel	0x00
0x5a	PEM_CH22_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x5b	PEM_CH22_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x5c	PEM_CH23_CTRL0	RW	[4:0] event_module_sel	0x00
0x5d	PEM_CH23_CTRL1	RW	[4:0] task_module_sel	0x00
0x5e	PEM_CH23_CTRL2	RW	[2:0] event_sig_sel [5:3] task_sig_sel [7:6] event_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00
0x5f	PEM_CH23_CTRL3	RW	[0] both_edge [2] inv [3] ch_en [4] event_lvl [5] task_lvl [7:6] task_clk_sel, 0:cclk; 1:hclk; 2:pclk; 3:aclk	0x00

16 PWM

The SoC supports 7-channel PWM (Pulse-Width-Modulation) output. Each PWM#n (n=0~6) has its corresponding inverted output at PWM#n_N pin. Each PWM channel has independent counter and 3 status including "Phase", "Pulse" and "Remaining". There are two waveforms for PWM#n. When register PWM_PHASE_MODE#n is set to 1, the Phase state is located in every PWM cycle, and when register PWM_PHASE_MODE#n is set to 0, the Phase state is located in before all PWM cycles.

16.1 Enable PWM

The register PWM_EN[6:1] and PWM_EN0[0] serves to enable PWM6~PWM0 respectively via writing "1" for the corresponding bits.

16.2 Set PWM Clock

PWM clock derives from system clock. Register PWM_CLKDIV serves to set the frequency dividing factor for PWM clock. Formula below applies:

$$F_{PWM} = F_{System\_clock} / (PWM\_CLKDIV+1)$$

16.3 PWM Waveform, Polarity and Output Inversion

16.3.1 Waveform of Signal Frame

When PWM_PHASE_MODE#n is 1'b0, the Phase state is before all PWM cycles, so each PWM cycle consists of Pulse state and Remaining state.

When PWM#n is enabled, the counter of PWM#n starts counting, and PWM#n enters the Phase state and outputs a low signal by default. When 'counter == PWM_PHASE#n', the counter is reset to 0, PWM#n completes the output of the Phase state signal, PWM#n enters into the Pulse state, and outputs high level signal by default. When 'counter == PWM_TCMP#n', PWM#n enters the Remaining state and outputs low level signal by default. When 'counter == PWM_TMAX#n', the counter is reset to 0, PWM#n completes one cycle of signal output, enters the Pulse state of the next cycle, and outputs a high level signal by default, and so on.

If 'PWM_PHASE#n == 0', there is no Phase state before all PWM cycles; if 'PWM_PHASE#n > 0', there is Phase state before all PWM cycles.

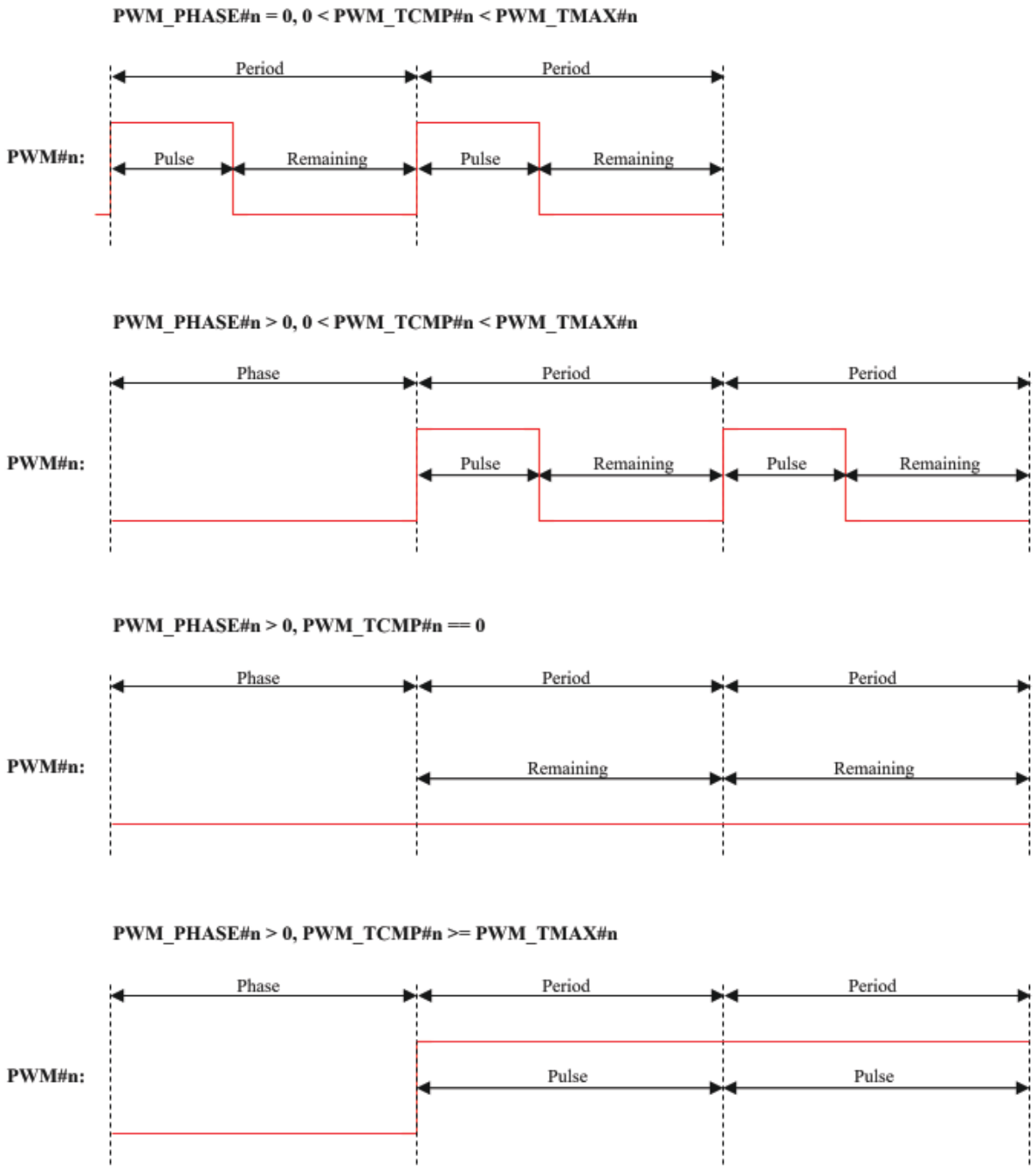
If 'PWM_TCMP#n == 0', there is no Pulse state for every PWM cycle.

If 'PWM_TCMP#n >= PWM_TMAX#n', there is no Remaining state for every PWM cycle.

If '0 < PWM_TCMP#n < PWM_TMAX#n', there are Pulse state and Remaining state for every PWM cycle.

The detailed waveform format is shown as below.

Figure 16-1 PWM Waveform 1



When PWM_PHASE_MODE#n is 1'b1, the Phase state is located in every PWM cycle, therefore every PWM cycle consists of Phase state, Pulse state and Remaining state.

When PWM#n is enabled, the counter of PWM#n starts counting, and PWM#n enters the Phase state and outputs a low signal by default. When 'counter == PWM_PHASE#n', PWM#n enters the Pulse state and outputs high level signal by default. When 'counter == (PWM_PHASE#n + PWM_TCMP#n)', PWM#n enters the Remaining state, and outputs low level signal by default. When 'counter == PWM_TMAX#n', the counter is reset to 0, PWM#n completes one cycle of signal output, and enters the Phase state of the next cycle, and outputs a low level signal by default.

If 'PWM_TCMP#n == 0', there is no Phase state and Pulse state for each PWM cycle.

If 'PWM_TCMP#n >= PWM_TMAX#n', there is no Phase state and Remaining state for every PWM cycle.

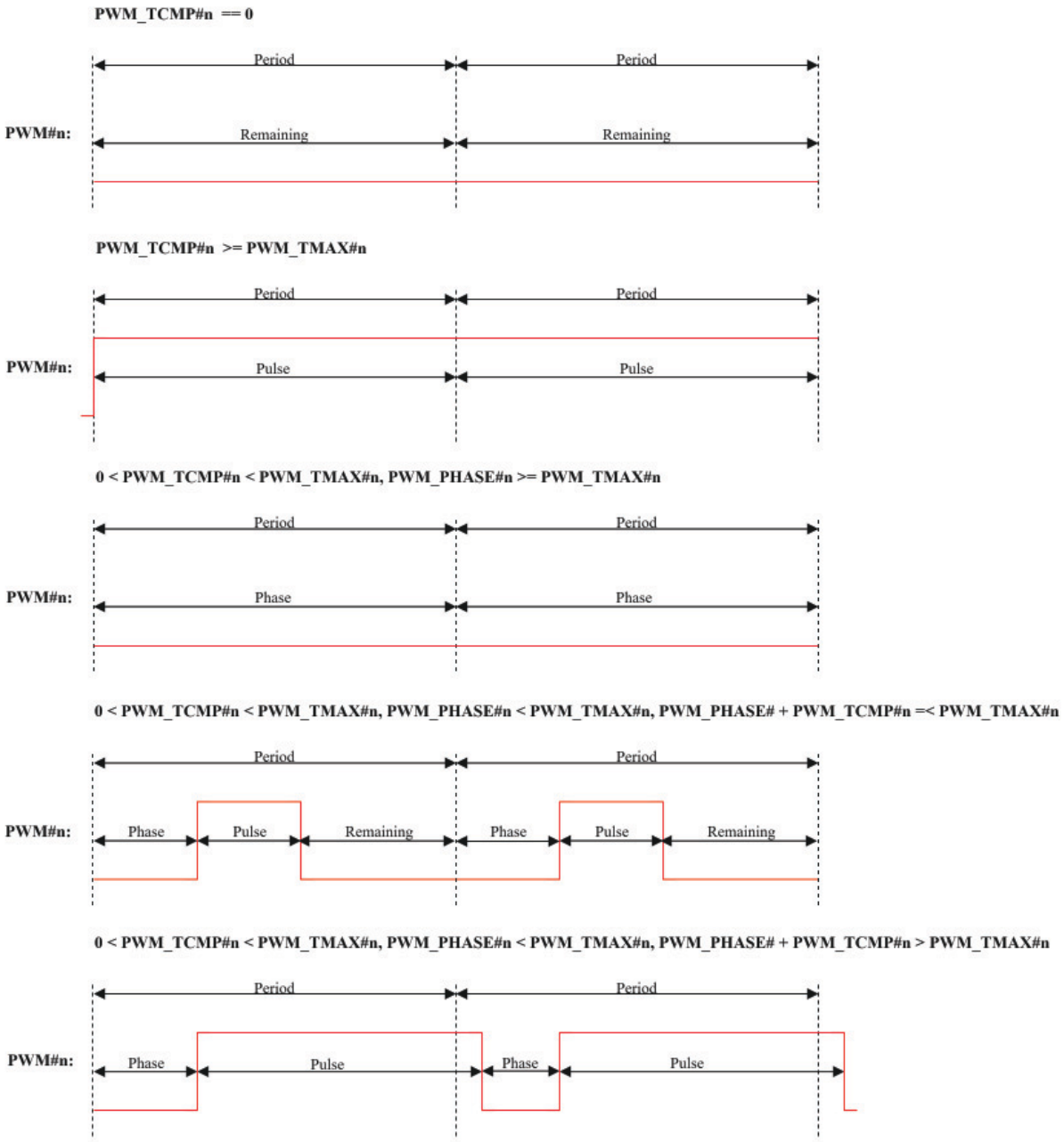
When '0 < PWM_TCMP#n < PWM_TMAX#n', if 'PWM_PHASE#n >= PWM_TMAX#n', there is no Pulse state and no Remaining state for every PWM cycle.

When '0 < PWM_TCMP#n < PWM_TMAX#n', if 'PWM_PHASE#n < PWM_TMAX#n' and '(PWM_PHASE#n + PWM_TCMP#n) < PWM_TMAX#n', there are Phase state, Pulse state and Remaining state for each PWM cycle.

When '0 < PWM_TCMP#n < PWM_TMAX#n', if 'PWM_PHASE#n < PWM_TMAX#n', and '(PWM_PHASE#n + PWM_TCMP#n) >= PWM_TMAX#n', then there is no Remaining state for each PWM cycle. It is worth noting that at this time, the Pulse state of the previous cycle overwrites the part of the Phase state of the next cycle.

The detailed waveform format is shown as below.

Figure 16-2 PWM Waveform 2



16.3.2 Invert PWM Output

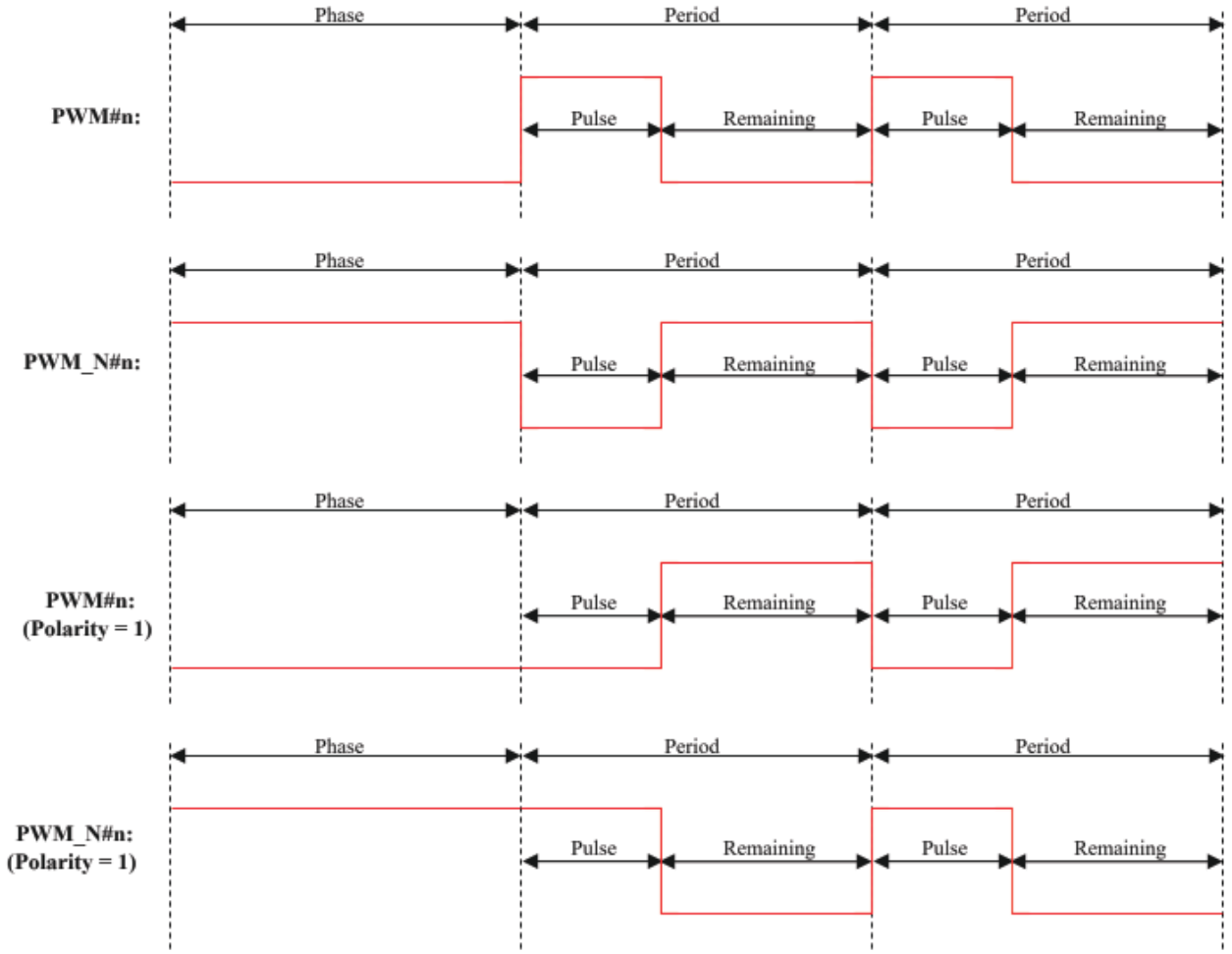
The PWM#n and PWM#n_N output could be inverted independently via register PWM_CC0 and PWM_CC1. When the inversion bit is enabled, waveform of the corresponding PWM channel is inverted completely.

16.3.3 Polarity for Signal Frame

By default, the PWM#n outputs High level at Pulse status and Low level at Remaining status. When the corresponding polarity bit is enabled via register PWM_CC2[6:0], PWM#n outputs Low level at Pulse status and High level at Remaining status. The output of PWM#n at Phase status is not affected by the corresponding polarity bit. The corresponding polarity bit can only be enabled when PWM_PHASE_MODE#n is set to 1'b0.

The PWM output waveform is shown as below.

Figure 16-3 PWM Output Waveform Chart



16.4 PWM Mode

16.4.1 Select PWM Modes

The PWM0 supports five modes, including Continuous mode (normal mode, default), Counting mode, IR mode, IR FIFO mode, IR DMA FIFO mode.

PWM1~PWM6 only support Continuous mode.

When PWM_PHASE_MODE#n is 1'b1, PWM0~PWM6 only support Continuous mode.

Register PWM_MODE serves to select PWM0 mode.

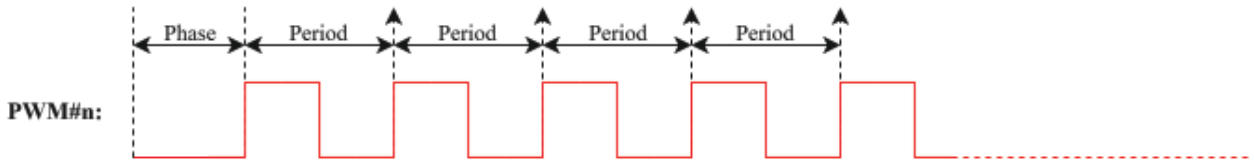
16.4.2 Continuous Mode

PWM0~PWM6 all support Continuous mode. In this mode, PWM#n continuously sends out signal frames. PWM#n should be disabled via PWM_EN/PWN_EN0 to stop it; when stopped, the PWM output turns low immediately.

When PWM_PHASE_MODE#n is 1'b0, during Continuous mode, waveform could be changed freely via PWM_TCMP#n and PWM_TMAX#n. New configuration for PWM_TCMP#n and PWM_TMAX#n takes effect when 'counter == PWM_TMAX#n'.

After each signal frame is finished, corresponding PWM cycle done interrupt flag bit (PWM_INT1[1:7]) is automatically set to 1'b1. If the interrupt is enabled by setting PWM_MASK1[1:7] as 1'b1, a frame interruption is generated. User needs to write 1'b1 to the flag bit to manually clear it.

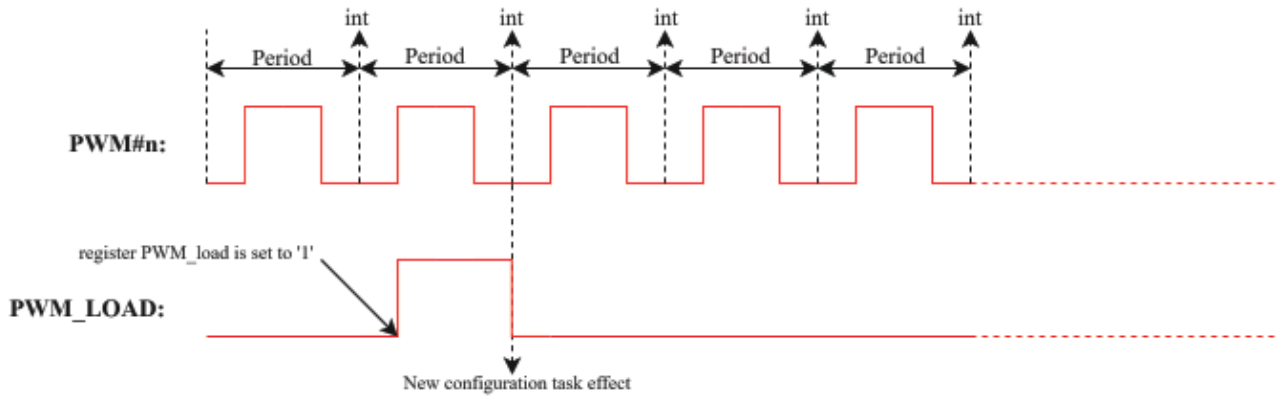
Figure 16-4 Continuous Mode (PWM_PHASE_MODE#n = 1'b0)



When PWM_PHASE_MODE#n is 1, during Continuous mode, the waveform could be changed via PWM_PHASE#n, PWM_TCMP#n and PWM_TMAX#n. New configuration for PWM_PHASE#n, PWM_TCMP#n and PWM_TMAX#n take effect when 'counter == PWM_TMAX#n' and 'PWM_LOAD == 1'b1'. After the new configuration takes effect, PWM_LOAD is reset to 1'b0. After each signal frame is finished, corresponding PWM cycle done interrupt flag bit (PWM_INT1[1:7]) is automatically set to 1'b1. If the interrupt is enabled by setting PWM_MASK1[1:7] as 1'b1, a frame interruption is generated. User needs to write 1'b1 to the flag bit to manually clear it.

NOTE: *When '0 < PWM_TCMP#n < PWM_TMAX#n', 'PWM_PHASE#n < PWM_TMAX#n', and '(PWM_PHASE#n + PWM_TCMP#n) >= PWM_TMAX#n', PWM Output Waveform is special, as detailed in* 16.6 PWM Load.

Figure 16-5 Continuous Mode (PWM_PHASE_MODE#n = 1'b1)



16.4.3 Counting Mode

Only PWM0 supports Counting mode. PWM_MODE [2:0] should be set as 4b'0001 to select PWM0 counting mode.

In this mode, PWM0 sends out specified number of signal frames which is defined as a pulse group. The number is configured via register PWM_PNUM.

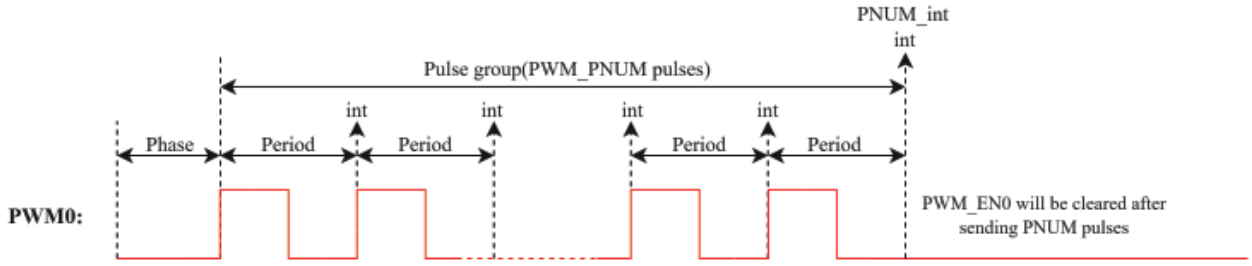
When a group of signals are transmitted, PWM0_EN is automatically disabled, PWM0 immediately toggles the transmission signal to low.

Write PWM_ MASK1[1] to 1 to enable interrupt. At the end of each cycle, set PWM_ INT1[1] to 1 to generate interrupt. Write PWM_ INT1[1] to 1 to clear the interrupt.

Write PWM_ MASK0[0] to 1 to enable the pnum interrupt. After transmitting a group of signals, set PWM_ INT0[0] to 1 to generate interrupt. Write PWM_ INT0[0] to 1 to clear the interrupt.

Counting mode also serves to stop IR mode gracefully.

Figure 16-6 Counting Mode



16.4.4 IR Mode

Only PWM0 supports IR mode. PWM_MODE[3:0] should be set as 4b'0011 to select PWM0 IR mode.

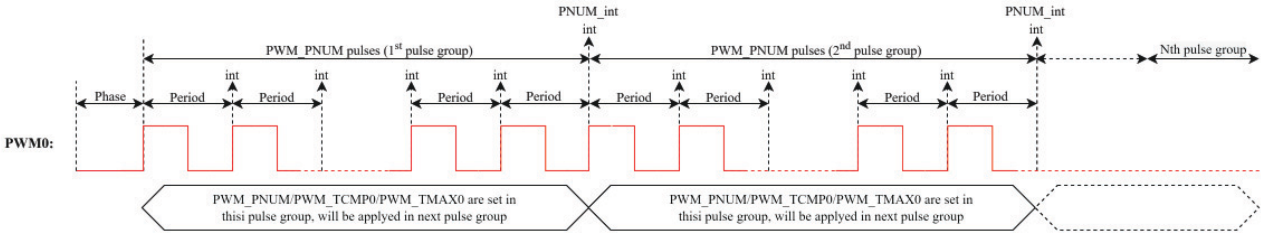
In this mode, specified number of frames is defined as one pulse group. In contrast to Counting mode where PWM0 stops after first pulse group is finished, PWM0 constantly sends pulse groups in IR mode.

During IR mode, PWM0 output waveform could also be changed freely via PWM_TCMP0, PWM_TMAX0 and PWM_PNUM0. New configuration for PWM_TCMP0, PWM_TMAX0 and PWM_PNUM0 take effect in the next pulse group.

To stop IR mode and complete current pulse group, user can switch PWM0 from IR mode to Counting mode so that PWM0 stops after current pulse group is finished. If PWM0 is disabled directly via PWM_EN0[0], PWM0 output turns Low immediately despite of current pulse group.

After each signal frame/pulse group is finished, PWM0 cycle done interrupt flag bit (PWM_INT1[1])/PWM0 pnum interrupt flag bit (PWM_INT0[0]) is automatically set to 1'b1. A frame interruption/Pnum interruption is generated.

Figure 16-7 IR Mode



16.4.5 IR FIFO Mode

IR FIFO mode is designed to allow IR transmission of long code patterns without the continued intervention of MCU, and it is designed as a selectable working mode on PWM0. The IR carrier frequency is divided down from the system clock and can be configured as any normal IR frequencies, e.g. 36 kHz, 38 kHz, 40 kHz, or 56 kHz.

Only PWM0 supports IR FIFO mode. PWM_MODE[3:0] should be set as 4b'0111 to select PWM0 IR FIFO mode.

An element ("FIFO CFG Data") is defined as basic unit of IR waveform, and written into FIFO. This element consists of 16 bits, including:

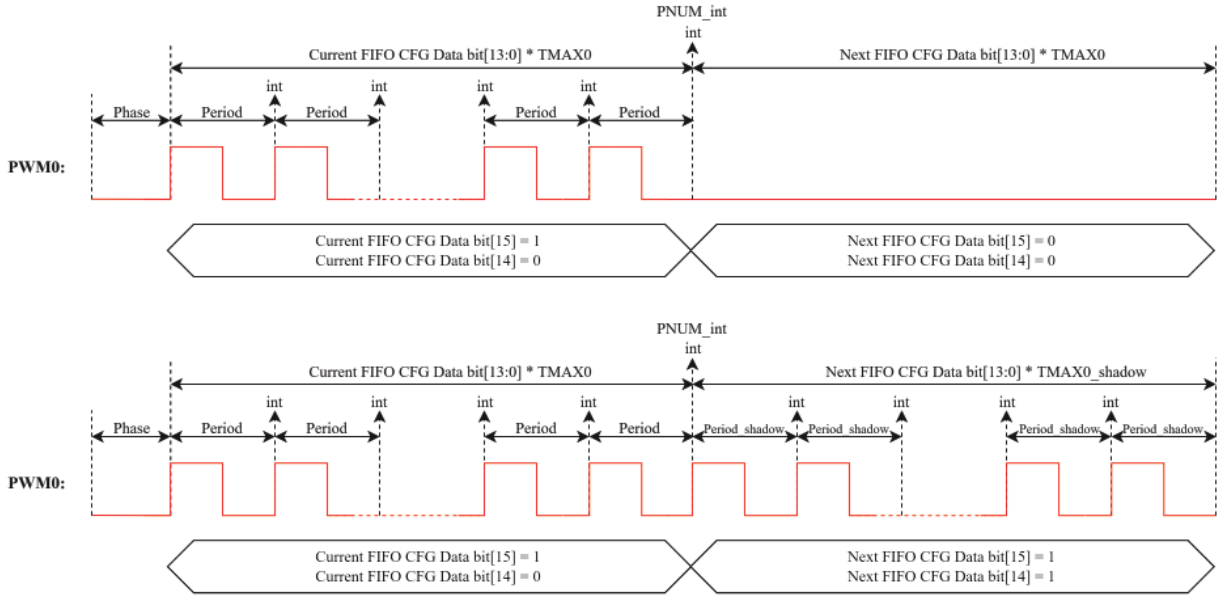
- bit[13:0] defines PWM cycles of current group.
- bit[14] determines the source of cmp and max value of the PWM signal in current group.
  - 1: use configuration of PWM_TCMP_FSK_L/PWM_TCMP_FSK_H and PWM_TMAX_FSK_L/ PWM_TMAX_FSK_H.
  - 0: use configuration of TCMP0 and TMAX0.
- bit[15] determines whether current PWM pulse group is used as carrier, i.e. whether PWM outputs pulse (1) or low level (0).

User should use PWM_RDAT_L0, PWM_RDAT_H0, PWM_RDAT_L1, PWM_RDAT_H1 to write the 16-bit "FIFO CFG Data" into FIFO by byte or half word or word.

- To write by byte, user should successively write 0x48, 0x49, 0x4a and 0x4b.
- To write by half word, user should successively write 0x48 and 0x4a.
- To write by word, user should write 0x48.

The FIFO depth is 8 bytes. User can read the register PWM_FIFO_STS in 0x55 to view FIFO empty/full status and check FIFO data number.

Figure 16-8 IR Format Examples



When "FIFO CFG Data" is configured in FIFO and PWM0 is enabled via PWM_EN0[0], the configured waveforms is output from PWM0 in sequence. As long as FIFO doesn't overflow, user can continue to add waveforms during IR waveforms sending process, and long IR code that exceeds the FIFO depth can be implemented this way. After all waveforms are sent, FIFO becomes empty, PWM0 is disabled automatically.

The FIFO_CLR register serves to clear data in FIFO. Writing 1 to this register clears all data in the FIFO. Note that the FIFO can only be cleared when not in active transmission.

16.4.6 IR DMA FIFO Mode

IR DMA FIFO mode is designed to allow IR transmission of long code patterns without occupation of MCU, and it is designed as a selectable working mode on PWM0. The IR carrier frequency is divided down from the system clock and can be configured as any normal IR frequencies, e.g. 36 kHz, 38 kHz, 40 kHz, or 56 kHz.

Only PWM0 supports IR DMA FIFO mode.PWM_MODE[3:0] should be set as 4b'1111 to select PWM0 IR DMA FIFO mode.

This mode is similar to IR FIFO mode, except that "FIFO CFG Data" is written into FIFO by DMA instead of MCU. User should write the configuration of "FIFO CFG Data" into RAM, and then enable DMA channel 5. DMA automatically writes the configuration into FIFO.

NOTE: *In this mode, when DMA channel 5 is enabled, PWM automatically outputs configured waveform, without the need to manually enable PWM0 via PWM_EN0 (i.e. PWM_EN0[0] is set as 1b'1 automatically).*

16.5 PWM Interrupt

There are 10 interrupt sources from PWM function.

After each signal frame, PWM#n (n = 0 ~ 6) generates a frame-done IRQ (Interrupt Request) signal.

In Counting mode and IR mode, PWM0 generates a Pnum IRQ signal after completing a pulse group.

In IR FIFO mode, PWM0 generates a FIFO mode count IRQ signal when the FIFO_NUM value is less than the FIFO_NUM_LVL, and generates a FIFO mode stop IRQ signal after FIFO becomes empty.

In IR DMA FIFO mode, PWM0 generates an IR waveform send done IRQ signal, after DMA has sent all configuration data, FIFO becomes empty and final waveform is sent.

To enable PWM interrupt, the total enabling bit "irq_pwm" should be set as 1b'1. To enable various PWM interrupt sources, PWM_MASK0 and PWM_MASK1 should be set as 1b'1 correspondingly.

Interrupt status can be cleared via register PWM_INT0 and PWM_INT1.

16.6 PWM Load

When PWM_PHASE_MODE#n is 1, during Continuous mode, waveform could be changed via PWM_PHASE#n, PWM_TCMP#n and PWM_TMAX#n. New configuration for PWM_PHASE#n, PWM_TCMP#n and PWM_TMAX#n take effect when 'counter == PWM_TMAX#n' and 'PWM_LOAD == 1'b1'. After the new configuration takes effect, PWM_LOAD is reset to 1'b0.

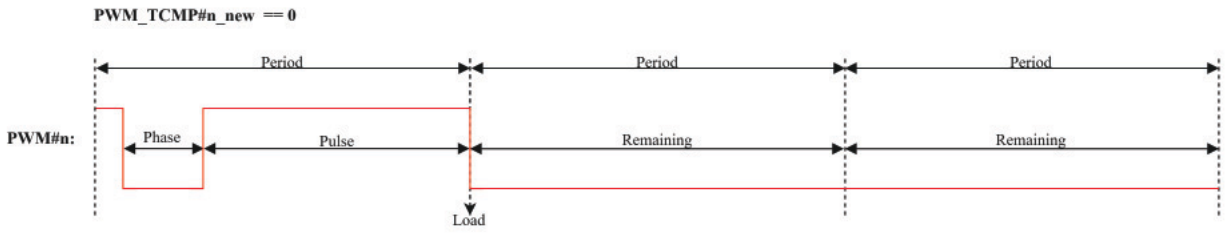
Assume the register value of current cycle are: PWM_PHASE#n_old, PWM_TCMP#n_old and PWM_TMAX#n_old;

The register value of PWM load are PWM_PHASE#n_new, PWM_TCMP#n_new and PWM_TMAX#n_new.

When '0 < PWM_TCMP#n_old < PWM_TMAX#n_old', 'PWM_PHASE#n_old < PWM_TMAX#n_old', and '(PWM_PHASE#n_old + PWM_TCMP#n_old) >= PWM_TMAX#n_old', PWM Output Waveform is special, the details are as follows.

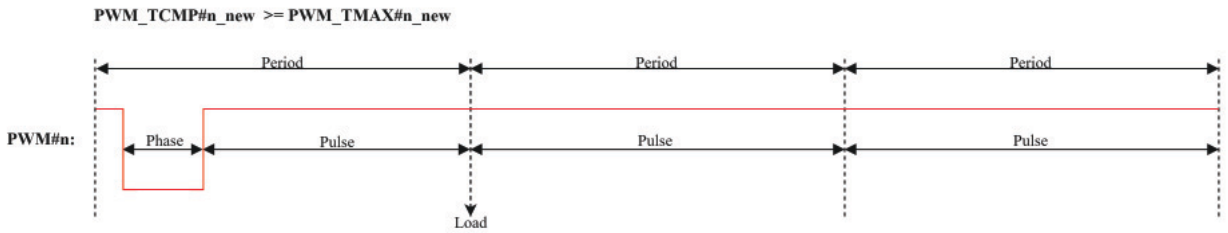
When 'PWM_TCMP#n_new == 0', there is no Phase state and Pulse state for every PWM cycle after the load.

Figure 16-9 PWM cycle after load 1



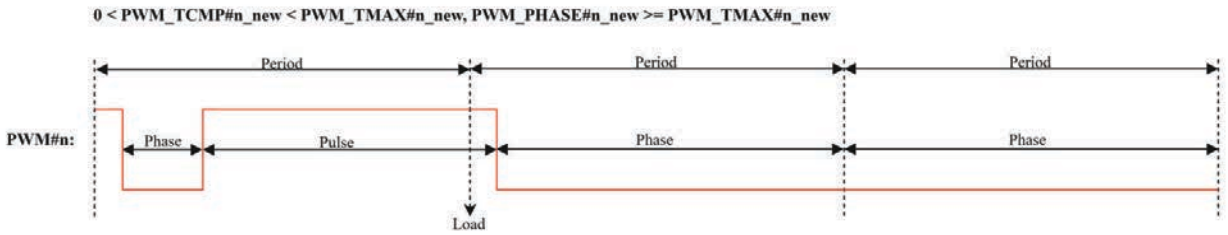
When 'PWM_TCMP#n_new >= PWM_TMAX#n_new', there is no Phase state and Remaining state for every PWM cycle after the load.

Figure 16-10 PWM cycle after load 2



When '0 < PWM_TCMP#n_new < PWM_TMAX#n_new' and 'PWM_PHASE#n_new >= PWM_TMAX#n_new', there is no Remaining state for each PWM cycle after the load. It is worth noting that the Pulse state of the cycle before the load overwrites the portion of the Phase state of the cycle after the load.

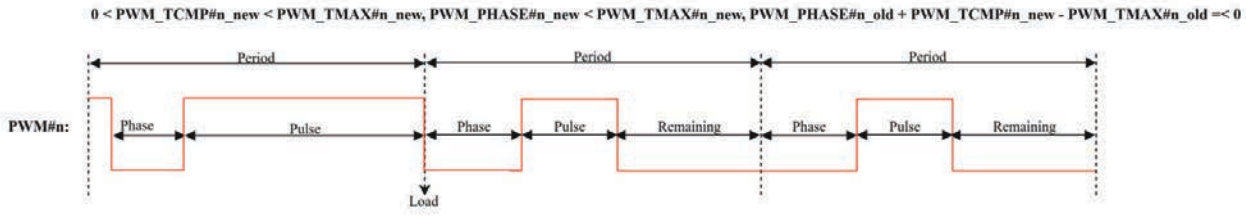
Figure 16-11 PWM cycle after load 3



When '0 < PWM_TCMP#n_new < PWM_TMAX#n_new' and 'PWM_PHASE#n_new < PWM_TMAX#n_new', thre are three cases after the load according to the relationship between (PWM_PHASE#n_old + PWM_TCMP#n_new - PWM_TMAX#n_old) and PWM_PHASE#n_new.

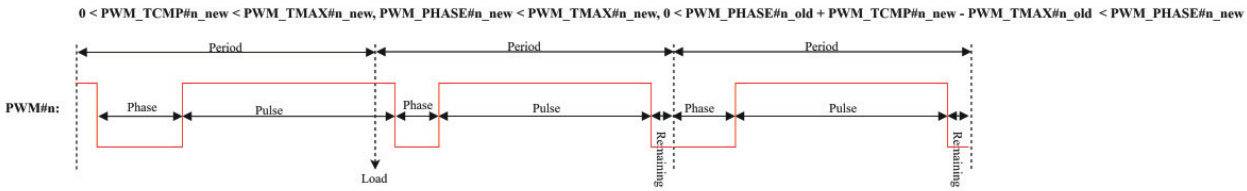
- Case1: when (PWM_PHASE#n_old + PWM_TCMP#n_new - PWM_TMAX#n_old) =< 0, PWM#n enters Phase state immediately after the load; when 'count == PWM_PHASE#n_new', the PWM#n enters Pulse state.

Figure 16-12 PWM cycle after load 4



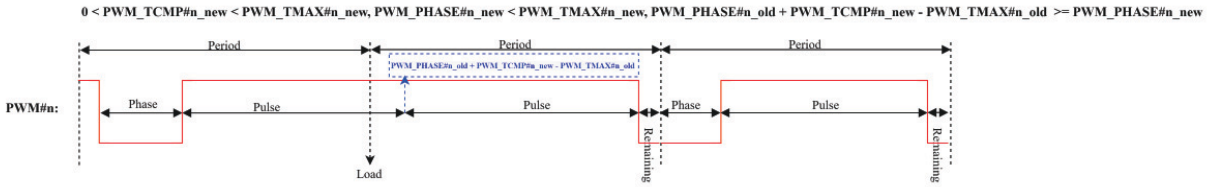
- Case2: 0< (PWM_PHASE#n_old + PWM_TCMP#n_new - PWM_TMAX#n_old) < PWM_PHASE#n_new, the PWM#n does not enter Phase state immediately after the load, and it continues to maintain the Pulse state of the previous cycle; when 'count == (PWM_PHASE#n_old + PWM_TCMP#n_new - PWM_TMAX#n_old)', the PWM#n enters Phase state in the current cycle; when 'count == PWM_ PHASE#n_new', the PWM#n enters Pulse state. The Phase state of PWM#n in the current cycle is partially overwritten by the Pulse state of the previous cycle.

Figure 16-13 PWM cycle after load 5



- Case3: (PWM_PHASE#n_old + PWM_TCMP#n_new - PWM_TMAX#n_old) >= PWM_PHASE#n_new, the PWM#n does not enter Phase state immediately after the load, and it continues to maintain the Pulse state of the previous cycle; when 'count == PWM_PHASE#n_new', the PWM#n directly enters the Pulse state of the current cycle, there is no Phase state in the current cycle of PWM#n, and the Phase state is completely overwritten by the Pulse state of the previous cycle.

Figure 16-14 PWM cycle after load 6

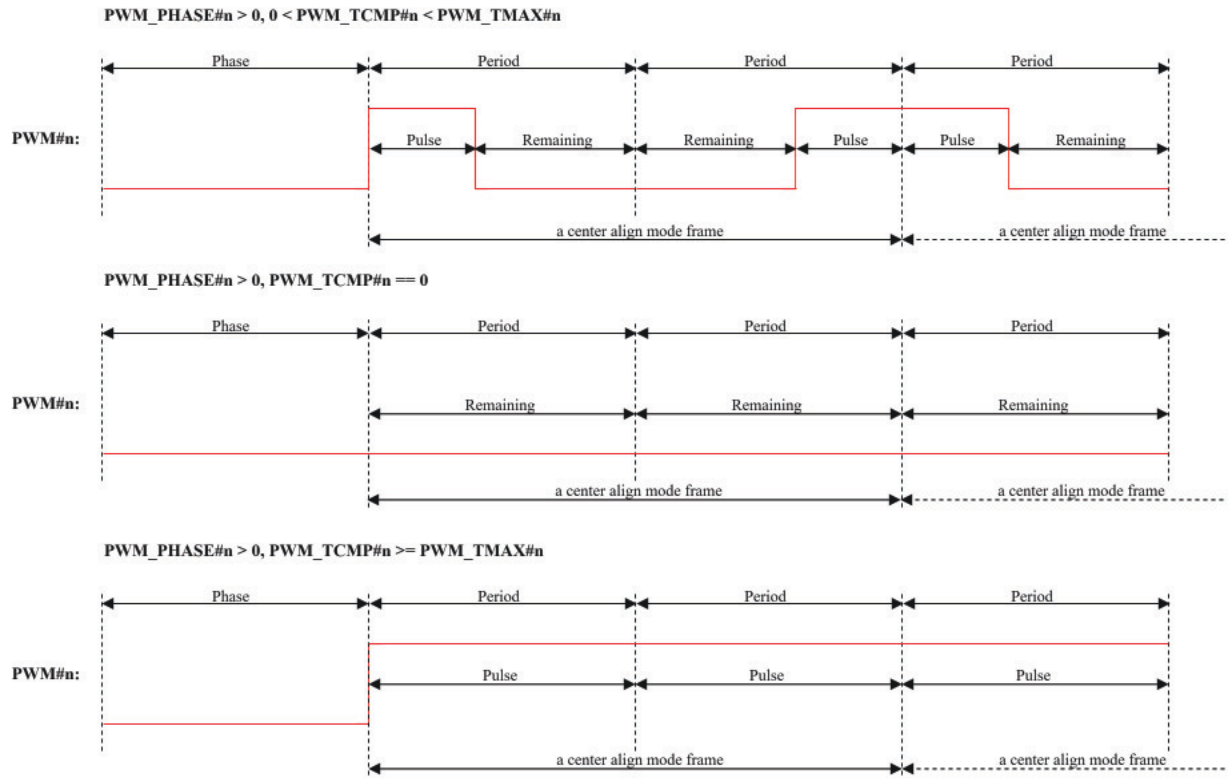


16.7 PWM Center Align Mode

The PWM#n (n = 0 ~ 6) support center align mode. Since the PWM#n is divided into two waveforms according to PWM_PHASE_MODE#n, the PWM center align mode can also be divided into two waveforms according to PWM_PHASE_MODE#n.

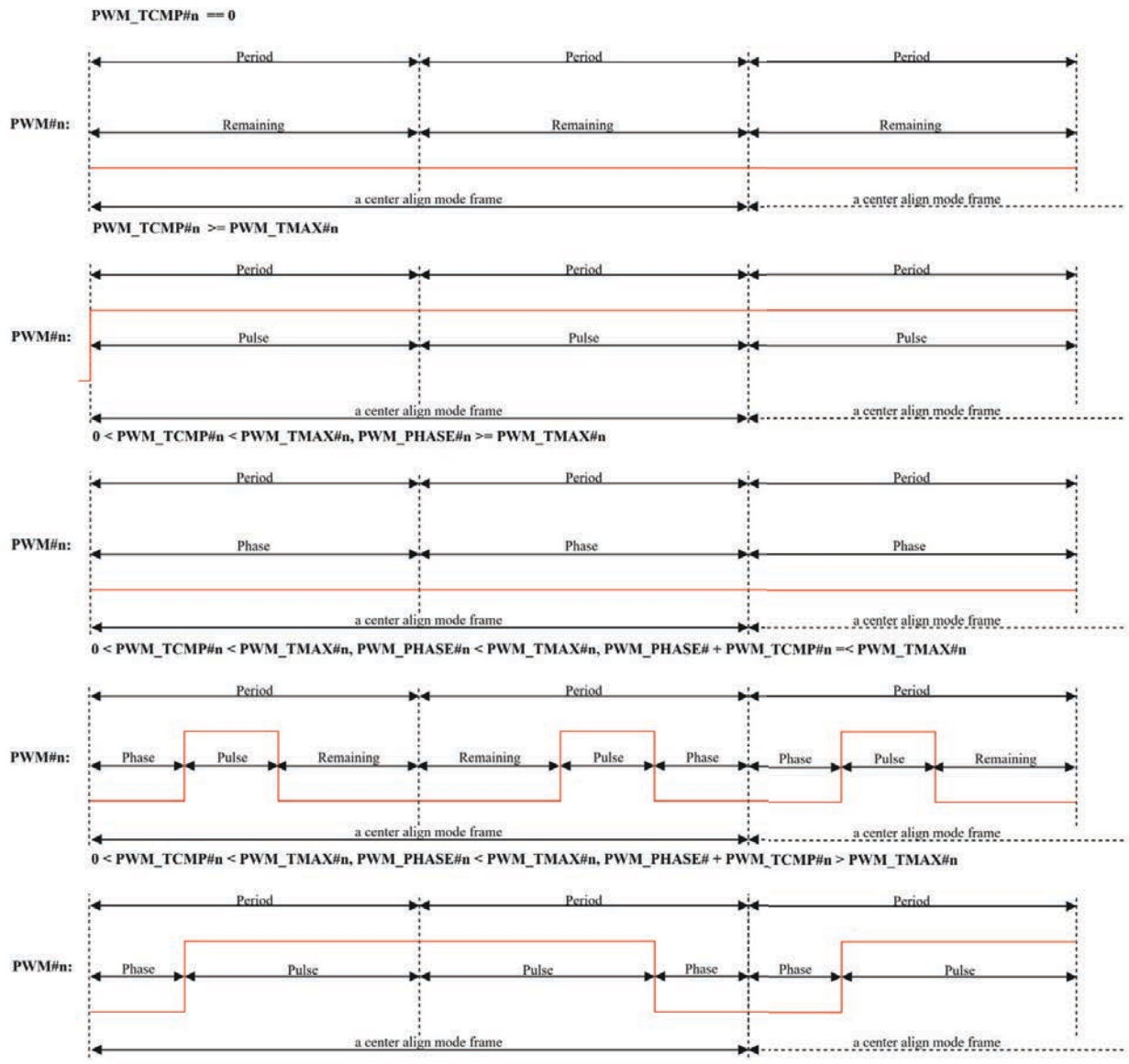
When PWM_PHASE_MODE#n == 1'b0, the waveform in center align mode is shown as below.

Figure 16-15 PWM Center Align Mode 1



When PWM_PHASE_MODE#n == 1'b1, the waveform in center align mode is shown as below.

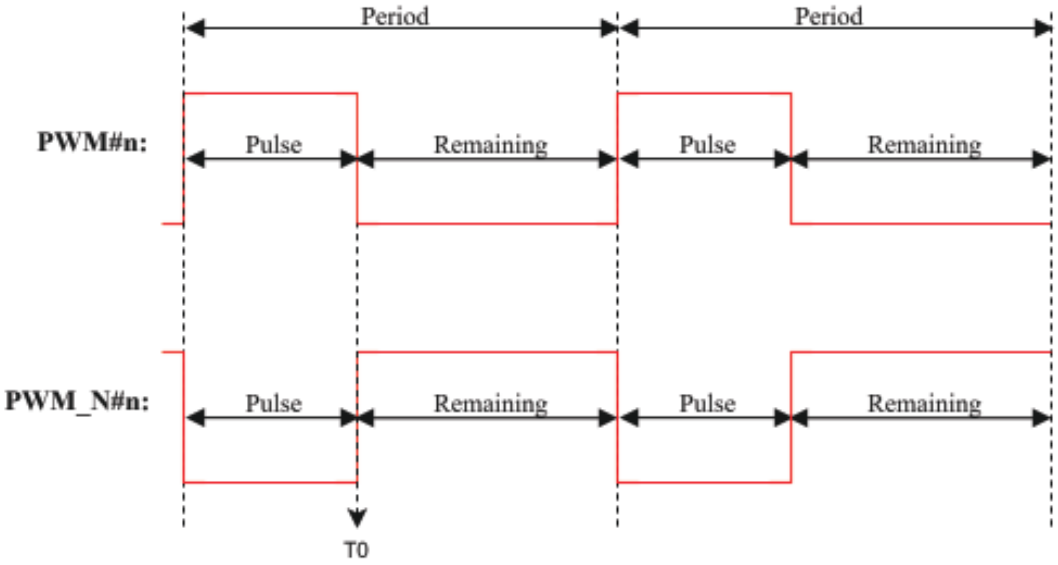
Figure 16-16 PWM Center Align Mode 2



16.8 PWM Dead Time

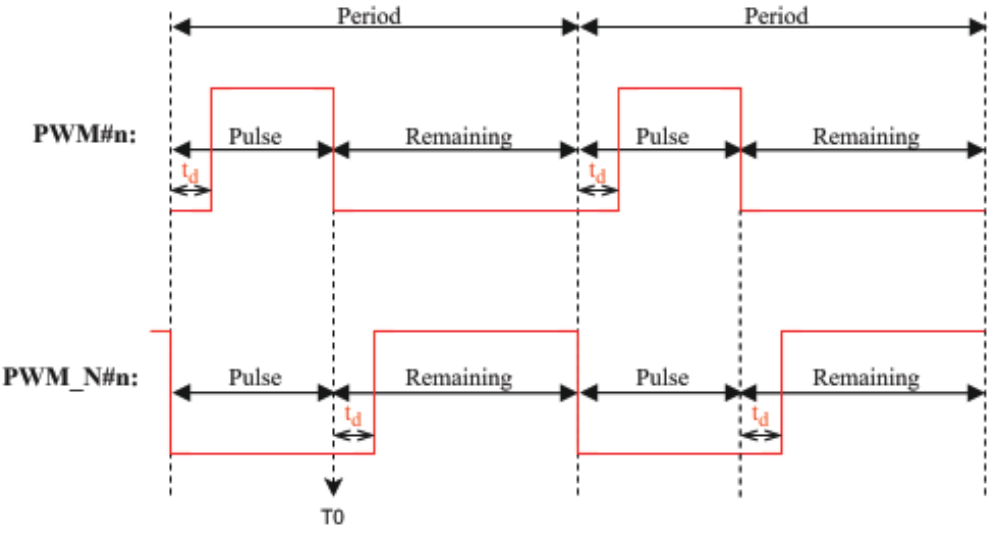
The working principle of Dead Time is: when configuring PWM_Dead_TIME#n to zero, the waveform of PWM#n and PWM_N#n are inverted as shown below:

Figure 16-17 PWM_Dead_TIME#n is zero



When configuring PWM_Dead_TIME#n to a non-zero value, the pulse rising edge of PWM#n and PWM_N#n is delayed for a period of time td, and the falling edge of pulse remains unchanged, so as to avoid the overlap between the pulses of PWM#n and PWM_N#n, as shown in the following figure:

Figure 16-18 PWM_Dead_TIME#n is non-zero



16.9 Enhanced Resolution with Dithering

Enhanced resolution with dithering only supports 'PWM_PHASE_MODE#n == 1'b1'.

Enhanced resolution with dithering supports center align mode.

16.9.1 Lookup Table

The lookup table is used to indicate the presence of jitter in the Pulse state for the current cycle. In the lookup table, 5LSB is register PWM_RESOL#n[4:0] and SLOT is the PWM cycle number. From the lookup table, we

can see that 32 PWM cycles are a group, that is, 32 SLOTs are a group, and when register PWM_RESOL#n[4:0] is set in the current group, it takes effect in the next group.

Figure 16-19 Lookup Table

SLOT / 5'LSB	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00000																																
00001	1																															
00010	1																1															
00011	1								1								1															
00100	1								1								1								1							
00101	1				1				1								1								1							
00110	1				1				1								1				1				1							
00111	1				1				1				1				1				1				1							
01000	1				1				1				1				1				1				1				1			
01001	1		1		1				1				1				1				1				1				1			
01010	1		1		1				1				1				1		1		1				1				1			
01011	1		1		1				1		1		1				1		1		1				1				1			
01100	1		1		1				1		1		1				1		1		1				1		1		1			
01101	1		1		1		1		1		1		1				1		1		1				1		1		1			
01110	1		1		1		1		1		1		1				1		1		1		1		1		1		1			
01111	1		1		1		1		1		1		1		1		1		1		1		1		1		1		1			
10000	1		1		1		1		1		1		1		1		1		1		1		1		1		1		1		1	
10001	1	1	1		1		1		1		1		1		1		1		1		1		1		1		1		1		1	
10010	1	1	1		1		1		1		1		1		1		1	1	1		1		1		1		1		1		1	
10011	1	1	1		1		1		1	1	1		1		1		1	1	1		1		1		1		1		1		1	
10100	1	1	1		1		1		1	1	1		1		1		1	1	1		1		1		1	1	1		1		1	
10101	1	1	1		1	1	1		1	1	1		1		1		1	1	1		1		1		1	1	1		1		1	
10110	1	1	1		1	1	1		1	1	1		1		1		1	1	1		1	1	1		1	1	1		1		1	
10111	1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1		1	
11000	1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1	
11001	1	1	1	1	1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1		1	1	1	
11010	1	1	1	1	1	1	1		1	1	1		1	1	1		1	1	1	1	1	1	1		1	1	1		1	1	1	
11011	1	1	1	1	1	1	1		1	1	1	1	1	1	1		1	1	1	1	1	1	1		1	1	1		1	1	1	
11100	1	1	1	1	1	1	1		1	1	1	1	1	1	1		1	1	1	1	1	1	1		1	1	1	1	1	1	1	
11101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1		1	1	1	1	1	1	1	
11110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
11111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

16.9.2 Principle of enhanced resolution with dithering

The default value of PWM_RESOL#n[4:0] is 5'b00000, that is, Enhanced resolution with dithering is not enabled by default;

The following takes example to illustrate the working principle of Enhanced resolution with dithering.

- Case1: When 'PWM_RESOL#n[4:0] == 5'b00000', from the lookup table, we can see that the dither corresponding to all SLOTs is 0, so there is no dithering in the Pulse state in every cycle, that is, in every cycle, when 'counter == PWM_TCMP#n', PWM#n enters Remaining state.
- Case2: When 'PWM_RESOL#n[4:0] == 5'b00001', from the lookup table, 'SLOT == 0', the corresponding dither is 1, and the other SLOT corresponding dither are 0, so there is jitter in the Pulse state in the 1st cycle, that is, PWM#n enters Remaining state in the 1st cycle when 'counter == PWM_TCMP#n+ 1'. The Pulse state in all other cycles is not jittered, that is, in the 2nd to 32nd cycle, PWM#n enters Remaining state when 'counter == PWM_TCMP#n'. If register PWM_RESOL#n[4:0] is not reset in the current group, the waveform of PWM#n output in the next group of 32 cycles is the same as the waveform of PWM#n output in the current group of 32 cycles. If register PWM_RESOL#n[4:0] is reset in the current group, the waveform output by PWM#n in the next set of 32 cycles is the same as the waveform output by PWM#n in the current set of 32 cycles according to the new PWM_RESOL#n[4:0].
- Case3: When 'PWM_RESOL#n[4:0] == 5'b00010', from the lookup table, 'SLOT == 0' and ' SLOT == 16', the corresponding dither is 1, and the dither corresponding to all other SLOTs is 0. Therefore, there is jitter in the Pulse state in the 1st and 17th cycles, that is, in the 1st and 17th cycles, 'counter == PWM_TCMP#n + 1 ' when PWM#n enters the Remaining state. There is no jitter in the Pulse state in any of the other cycles, that is, in the other cycles, PWM#n enters the Remaining state when 'counter == PWM_TCMP#n'.
- and so forth.

16.10 Deep Diming Synchronization

Deep Diming Synchronization only supports 'PWM_PHASE_MODE#n == 1'b1'.

Deep Diming Synchronization does not support center align mode.

16.10.1 Principle of Deep Diming Synchronization

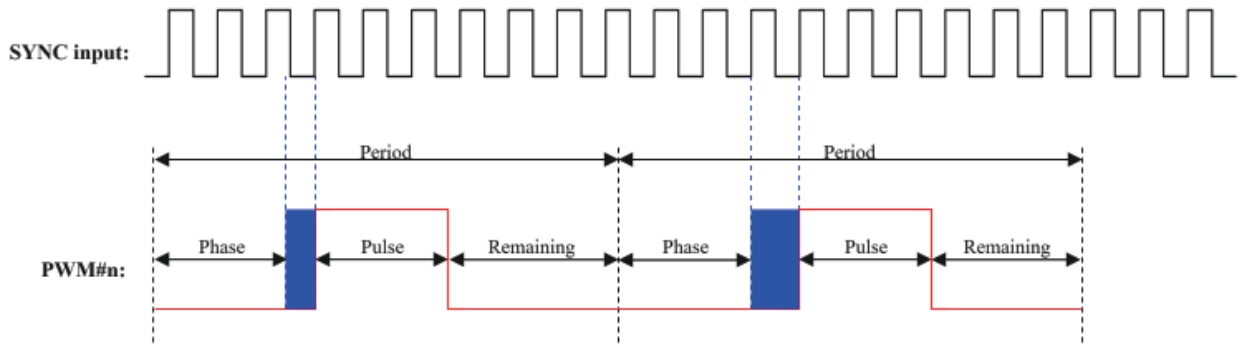
Deep Diming Synchronization means that in every cycle, the moment PWM#n enters Pulse state from Phase state needs to be aligned with the effective edge of the input signal outside the chip (PWM_SYNC_EDGE#n == 1'b0, the effective edge is the rising edge; PWM_SYNC_ EDGE#n == 1'b1, the effective edge is the falling edge).

All PWM#n (n = 0~6) from Phase state into Pulse state are aligned only with the edges of the input signals external to the same chip.

Deep Diming Synchronization does not affect the duration of the Pulse state per cycle.

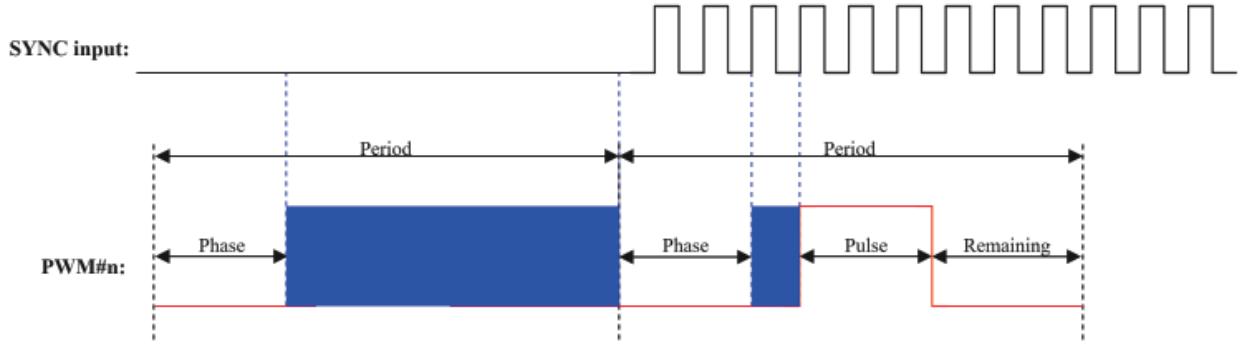
When 'PWM_PHASE_MODE#n == 1'b1' and 'PWM_SYNC_EN#n == 1'b1', Deep Diming Synchronization is enabled. When PWM#n is enabled, PWM#n's counter starts counting and PWM#n enters Phase state. When 'counter == PWM_PHASE#n', PWM#n does not enter Pulse state immediately, and needs to wait for the valid edge of the input signal outside the chip. When the valid edge of the input signal from outside the chip arrives, PWM#n enters the Pulse state and record the value of counter as CNT_SYNC#n at this moment. When 'counter == (PWM_TCMP#n + CNT_SYNC#n)', PWM#n enters into the Remaining state as shown below.

Figure 16-20 PWM#n Cycle for Deep Diming Synchronization 1



If after 'counter == PWM_PHASE#n' and before 'counter == PWM_TMAX#n', there is no waiting for a valid edge of the input signal from outside the chip, the current cycle PWM#n always outputs a low level, as shown in the following figure.

Figure 16-21 PWM#n Cycle for Deep Diming Synchronization 2



16.11 PWM Register Description

The PWM related registers are listed as following.The base address for below registers is 0x80140400.

Table 16-1 PWM Registers

Address Offset	Name	Type	Description	Default Value
0x00	PWM_EN	W	[6:1]: pwm1~6 enable	0x00
0x01	PWM_EN0	W	[0]: pwm0 enable	0x00
0x02	PWM_CLKDIV	RW	[7:0]: clkdiv	0x00
0x03	PWM_MODE	RW	[0]: crun_o [1]: catch_o [2]: fifo_mode_en [3]: txdma_en [4]: out2ana_en, 1: ir2ana = pwm0, 0: ir2ana = 0;	0x00
0x04	PWM_CC0	RW	[6:0]: inv, invert PWM output	0x00
0x05	PWM_CC1	RW	[6:0]: pos, invert PWM_INV output	0x00
0x06	PWM_CC2	RW	[6:0]: pola, PWM pola	0x00
0x07	MODE32K	RW	[6:0]: mode_32k	0x00
0x08	PWM_TCMP0_L	RW	tcmpb0[7:0]	0x00
0x09	PWM_TCMP0_H	RW	tcmpb0[15:8]	0x00
0x0a	PWM_TMAX0_L	RW	tmaxb0[7:0]	0x00
0x0b	PWM_TMAX0_H	RW	tmaxb0[15:8]	0x00
0x0c	PWM_TCMP1_L	RW	tcmpb1[7:0]	0x00
0x0d	PWM_TCMP1_H	RW	tcmpb1[15:8]	0x00
0x0e	PWM_TMAX1_L	RW	tmaxb1[7:0]	0x00
0x0f	PWM_TMAX1_H	RW	tmaxb1[15:8]	0x00
0x10	PWM_TCMP2_L	RW	tcmpb2[7:0]	0x00
0x11	PWM_TCMP2_H	RW	tcmpb2[15:8]	0x00
0x12	PWM_TMAX2_L	RW	tmaxb2[7:0]	0x00
0x13	PWM_TMAX2_H	RW	tmaxb2[15:8]	0x00
0x14	PWM_TCMP3_L	RW	tcmpb3[7:0]	0x00

Address Offset	Name	Type	Description	Default Value
0x15	PWM_TCMP3_H	RW	tcmpb3[15:8]	0x00
0x16	PWM_TMAX3_L	RW	tmaxb3[7:0]	0x00
0x17	PWM_TMAX3_H	RW	tmaxb3[15:8]	0x00
0x18	PWM_TCMP4_L	RW	tcmpb4[7:0]	0x00
0x19	PWM_TCMP4_H	RW	tcmpb4[15:8]	0x00
0x1a	PWM_TMAX4_L	RW	tmaxb4[7:0]	0x00
0x1b	PWM_TMAXB4_H	RW	tmaxb4[15:8]	0x00
0x1c	PWM_TCMP5_L	RW	tcmpb5[7:0]	0x00
0x1d	PWM_TCMP5_H	RW	tcmpb5[15:8]	0x00
0x1e	PWM_TMAX5_L	RW	tmaxb5[7:0]	0x00
0x1f	PWM_TMAX5_H	RW	tmaxb5[15:8]	0x00
0x20	PWM_TCMP6_L	RW	tcmpb6[7:0]	0x00
0x21	PWM_TCMP6_H	RW	tcmpb6[15:8]	0x00
0x22	PWM_TMAX6_L	RW	tmaxb6[7:0]	0x00
0x23	PWM_TMAX6_H	RW	tmaxb6[15:8]	0x00
0x24	PWM_PHASE0_L	RW	phase0[7:0]	0x00
0x25	PWM_PHASE0_H	RW	phase0[15:8]	0x00
0x26	PWM_PHASE1_L	RW	phase1[7:0]	0x00
0x27	PWM_PHASE1_H	RW	phase1[15:8]	0x00
0x28	PWM_PHASE2_L	RW	phase2[7:0]	0x00
0x29	PWM_PHASE2_H	RW	phase2[15:8]	0x00
0x2a	PWM_PHASE3_L	RW	phase3[7:0]	0x00
0x2b	PWM_PHASE3_H	RW	tmaxb3[15:8]	0x00
0x2c	PWM_PHASE4_L	RW	phase4[7:0]	0x00
0x2d	PWM_PHASE4_H	RW	phase4[15:8]	0x00
0x2e	PWM_PHASE5_L	RW	phase5[7:0]	0x00
0x2f	PWM_PHASE5_H	RW	phase5[15:8]	0x00

Address Offset	Name	Type	Description	Default Value
0x30	PWM_PHASE6_L	RW	phase6[7:0]	0x00
0x31	PWM_PHASE6_H	RW	phase6[15:8]	0x00
0x32	PWM_PNUM_L	RW	pnumb[7:0]	0x00
0x33	PWM_PNUM_H	RW	pnumb[13:8]	0x00
0x34	PWM_CENTER	RW	[6:0]: center_align_o	0x00
0x35	PWM_FIFO_CTRL	RW	[0]: auto_txclr_off [1]: txf_nempty_en	0x00
0x36	PWM_MASK0	RW	[0]: mask_pwm, [1]: mask_fifo	0x00
0x37	PWM_MASK1	RW	[0]: mask_lvl [7:1]: mask	0x00
0x38	PWM_INT0	W1C	[0]: int_pwm, pwm done interrupt in counting mode [1]: int_fifo_done, fifo done interrupt	0x00
0x39	PWM_INT1	W1C	[0]: int_lvl, Interrupts with number less than fifo_lvl in fifo [7:1]: int_flag, cycle done interrupt in continuous mode	0x00
0x3a	PWM_CNT0_L	VOLATILE	cmpcnt0_i[7:0]	0x00
0x3b	PWM_CNT0_H	VOLATILE	cmpcnt0_i[15:8]	0x00
0x3c	PWM_CNT1_L	VOLATILE	cmpcnt1_i[7:0]	0x00
0x3d	PWM_CNT1_H	VOLATILE	cmpcnt1_i[15:8]	0x10
0x3e	PWM_CNT2_L	VOLATILE	cmpcnt2_i[7:0]	0x00
0x3f	PWM_CNT2_H	VOLATILE	cmpcnt2_i[15:8]	0x00
0x40	PWM_CNT3_L	VOLATILE	cmpcnt3_i[7:0]	0x00
0x41	PWM_CNT3_H	VOLATILE	cmpcnt3_i[15:8]	0x00
0x42	PWM_CNT4_L	VOLATILE	cmpcnt4_i[7:0]	0x00
0x43	PWM_CNT4_H	VOLATILE	cmpcnt4_i[15:8]	0x00
0x44	PWM_CNT5_L	VOLATILE	cmpcnt5_i[7:0]	0x00
0x45	PWM_CNT5_H	VOLATILE	cmpcnt5_i[15:8]	0x00
0x46	PWM_CNT6_L	VOLATILE	cmpcnt6_i[7:0]	0x00

Address Offset	Name	Type	Description	Default Value
0x47	PWM_CNT6_H	VOLATILE	cmpcnt6_i[15:8]	0x00
0x48	PWM_RDAT0_L	RW	tx_rdat[7:0]	0x00
0x49	PWM_RDAT0_H	RW	tx_rdat[15:8]	0x00
0x4a	PWM_RDAT1_L	W	tx_rdat[7:0]	0x00
0x4b	PWM_RDAT1_H	W	tx_rdat[15:8]	0x00
0x4c	PWM_TCMP_FSK_L	RW	tcmpb_fsk[7:0]	0x00
0x4d	PWM_TCMP_FSK_H	RW	tcmpb_fsk[15:8]	0x00
0x4e	PWM_TMAX_FSK_L	RW	tmaxb_fsk[7:0]	0x00
0x4f	PWM_TMAX_FSK_H	RW	tmaxb_fsk[15:8]	0x00
0x50	PWM_PHASE_FSK_L	RW	phase_fsk[7:0]	0x00
0x51	PWM_PHASE_FSK_H	RW	phase_fsk[15:8]	0x00
0x52	PWM_NCNT_L	R	numcnt_i[7:0]	0x00
0x53	PWM_NCNT_H	R	numcnt_i[13:8]	0x00
0x54	PWM_FIFO_LVL	RW	[3:0]: fifo_lvl	0x00
0x55	PWM_FIFO_STS	R	[3:0]: tx_buf_cnt [4]: tx_empty [5]: tx_full	0x00
0x56	CLR_TX_FIFO	W	[0]: tx_clr PWM_LOAD [1]: reload_pul (load_sclk_o)	0x00
0x57	PWM_RESOL0	RW	[4:0]: resol0, extra resolution of PWM0	0x00
0x58	PWM_RESOL1	RW	[4:0]: resol1, extra resolution of PWM1	0x00
0x59	PWM_RESOL2	RW	[4:0]: resol2, extra resolution of PWM2	0x00
0x5a	PWM_RESOL3	RW	[4:0]: resol3, extra resolution of PWM3	0x00
0x5b	PWM_RESOL4	RW	[4:0]: resol4, extra resolution of PWM4	0x00
0x5c	PWM_RESOL5	RW	[4:0]: resol5, extra resolution of PWM5	0x00
0x5d	PWM_RESOL6	RW	[4:0]: resol6, extra resolution of PWM6	0x00

Address Offset	Name	Type	Description	Default Value
0x5e	PWM_SYNC_EN	RW	[6:0] pwm0~6 deep Diming Synchronization enable control: [0]: PWM_SYNC_EN0, [1]: PWM_SYNC_EN1, [2]: PWM_SYNC_EN2, [3]: PWM_SYNC_EN3, [4]: PWM_SYNC_EN4, [5]: PWM_SYNC_EN5, [6]: PWM_SYNC_EN6,	0x00
0x5f	PWM_SYNC_EDGE	RW	[6:0] pwm0~6 edge of pulse triggers synchronization: 1: negative edge of pulse triggers synchronization 0: positive edge of pulse triggers synchronization [0]: PWM_SYNC_EDGE0, [1]: PWM_SYNC_EDGE1, [2]: PWM_SYNC_EDGE2, [3]: PWM_SYNC_EDGE3, [4]: PWM_SYNC_EDGE4, [5]: PWM_SYNC_EDGE5, [6]: PWM_SYNC_EDGE6,	0x00
0x60	PWM_PEM_CTRL	RW	[0]: pem_event1_sel, 1: pem_event1_o = pem_event1[15:8], {5'h0, hit_lvl,fifo_done, pwmdone} 0: pem_event1_o = pem_event1[7:0], {1'b0, cycdone} [1]: pem_event1_en, [2]: pem_event0_en, pem_event0_o = pem_event0[7:0], {1'b0,period_start} [3]: pem_task1_en, clren = pem_task1_i[6:0] [4]: pem_task0_en, seten = pem_task0_i[6:0]	0x00

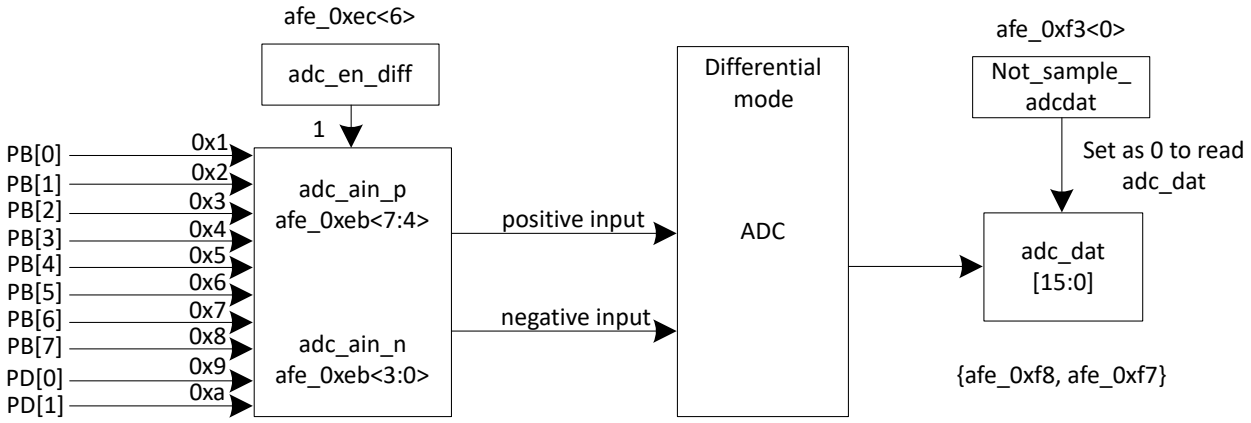
Address Offset	Name	Type	Description	Default Value
0x61	PWM_PHASE_MODE	RW	[6:0] pwm0~6 phase mode: 1: phase state exists during every pwm cycle 0: phase state exists only before all pwm cycles [0]: PWM_PHASE_MODE0, [1]: PWM_PHASE_MODE1, [2]: PWM_PHASE_MODE2, [3]: PWM_PHASE_MODE3, [4]: PWM_PHASE_MODE4, [5]: PWM_PHASE_MODE5, [6]: PWM_PHASE_MODE6,	0x00
0x62 ~ 0x63	PWM_DEAD_TIME0	RW	[15:0] dead_time0	0x00
0x64 ~ 0x65	PWM_DEAD_TIME1	RW	[15:0] dead_time1	0x00
0x66 ~ 0x67	PWM_DEAD_TIME2	RW	[15:0] dead_time2	0x00
0x68 ~ 0x69	PWM_DEAD_TIME3	RW	[15:0] dead_time3	0x00
0x6a ~ 0x6b	PWM_DEAD_TIME4	RW	[15:0] dead_time4	0x00
0x6c ~ 0x6d	PWM_DEAD_TIME5	RW	[15:0] dead_time5	0x00
0x6e ~ 0x6f	PWM_DEAD_TIME6	RW	[15:0] dead_time6	0x00

17 SAR ADC

The SoC integrates one SAR ADC module, which can be used to sample analog input signals such as battery voltage and temperature sensor.

The diagram of SAR ADC module is shown in figure below.

Figure 17-1 Diagram of ADC



NOTE:

- If the IO voltage of GPIO is configured as 1.8V, the sampling range of GPIO signal is 0 ~ 1.8 V; the maximum input voltage for ADC detection cannot be higher than 1.8V. If it is needed to detect a voltage higher than 1.8V, an external voltage divider circuit is required.
- If the IO voltage of GPIO is configured as 3.3V, the sampling range of GPIO signal is 0 ~ 3.3 V.

17.1 Power On/Down

The SAR ADC is disabled by default. To power on the ADC, the analog register adc_pd (afe_0xfc<5>) should be set as 1'b0.

17.2 ADC Clock

ADC clock is derived from external 24 MHz crystal source, with frequency dividing factor configurable via the analog register adc_clk_div (afe_0xf4<2:0>).

ADC clock frequency (marked as F_{ADC_clk}) = 24MHz/(adc_clk_div+1)

NOTE: The ADC clock is fixed at 4 MHz and should not be modified.

17.3 ADC Control in Auto Mode

17.3.1 Set Max State and Enable Channel

The SAR ADC supports Misc channel which consists of one "Set" state and one "Capture" state.

- The analog register r_max_scnt (afe_0xf2<5:4>) serves to set the max state index. As shown below, the r_max_scnt should be set as 0x02.
- The Misc channel can be enabled via r_en_misc (afe_0xf2<2>).

17.3.2 "Set" State

The length of "Set" state for the Misc channel is configurable via the analog register r_max_s (afe_0xf1<3:0>).

"Set" state duration (marked as T_{sd}) = r_max_s / 24MHz.

Each "Set" state serves to set ADC control signals for the Misc channel via corresponding analog registers, including:

- adc_en_diff: afe_0xec<6>. MUST set as 1'b1 to select differential input mode.
- adc_ain_p: afe_0xeb<7:4>. Select positive input in differential mode.
- adc_ain_n: afe_0xeb<3:0>. Select negative input in differential mode.
- adc_vref: afe_0xea<1:0>. Set reference voltage V_{REF}. ADC maximum input range is determined by the ADC reference voltage.
- adc_sel_ai_scale: afe_0xfa<7:6>. Set scaling factor for ADC analog input as 1 (default), or 1/8.

By setting this scaling factor, ADC maximum input range can be extended based on the V_{REF}.

For example, suppose the V_{REF} is set as 1.2V:

Since the scaling factor is 1 by default, the ADC maximum input range should be 0~1.2V (negative input is GND) / -1.2V~+1.2V (negative input is ADC GPIO pin).

If the scaling factor is set as 1/8, in theory ADC maximum input range should change to 0~9.6V (negative input is GND) / -9.6V~+9.6V (negative input is ADC GPIO pin). But limited by input voltage of the chip's PAD, the actual range is narrower.

- adc_res: afe_0xec<1:0>. Set resolution as 8/10/12/14 bits.

ADC data is always 16-bit format no matter what the resolution is set. For example, 14 bits resolution indicates ADC data consists of 14-bit valid data and 2-bit sign extension bit.

- adc_tsamp: afe_0xee<3:0>. Set sampling time which determines the speed to stabilize input signals.

Sampling time (marked as T_{samp}) = adc_tsamp / F_{ADC_clk}.

The lower sampling cycle, the shorter ADC convert time.

17.3.3 "Capture" State

For the Misc channel, at the beginning of its "Capture" state, a "run" signal is issued automatically to start an ADC sampling and conversion process; at the end of "Capture" state, ADC output data is captured.

- The length of "Capture" state is configurable via the analog register r_max_mc[9:0] (afe_0xf1<7:6>, afe_0xef<7:0>).

  "Capture" state duration for Misc channel (marked as T_{cd}) = r_max_mc / 24MHz.

- The "VLD" bit (afe_0xf6<0>) is set as 1'b1 at the end of "Capture" state to indicate the ADC data is valid, and this flag bit is cleared automatically.

- The 16-bit ADC output data can be read from the analog register adc_dat[15:0] (afe_0xf8<7:0>, afe_0xf7<7:0>) while the afe_0xf3<0> is set as 1'b0 (default). If the afe_0xf3<0> is set as 1'b1, the data in the afe_0xf8 and afe_0xf7 won't be updated.

NOTE: *The total duration "T_{td}", which is the sum of the length of "Set" state and "Capture" state, determines the sampling rate. Sampling frequency (marked as F_s) = 1 / T_{td}*

17.3.4 Usage Case with Detailed Register Setting

This case introduces the register setting details for Misc channel sampling.

In this case, afe_0xf2<2> should be set as 1'b1, so as to enable the Misc channel, while the max state index should be set as "2" by setting afe_0xf2<5:4> as 0x2.

The total duration (marked as T_{td}) = (1*r_max_s+1*r_max_mc) / 24MHz.

Table 17-1 Overall Register Setting

Function	Register Setting
Power on the ADC	afe_0xfc<5> = 1'b0
Set F_{ADC_clk} (ADC clock frequency) as 4MHz	afe_0xf4<2:0> = 5 F_{ADC_clk} = 24MHz/(5+1) = 4 MHz
Enable the Misc channel	afe_0xf2<2> = 1'b1
Set the max state index as "2"	afe_0xf2<5:4> = 2'b10
Set T_{sd} ("Set" state duration)	afe_0xf1<3:0> = 10 T_{sd} = r_max_s/24 MHz = 10/24 MHz = 0.417 μs
Set T_{cd} ("Capture" state duration)	afe_0xf1<7:6> = 1, afe_0xef<7:0> = 0xea T_{cd} = r_max_mc[9:0]/24 MHz = 490/24 MHz = 20.417 μs
T_{td} (total duration)	T_{td} = (1*r_max_s+1*r_max_mc) / 24 MHz = 500/24 MHz = 20.83 μs
F_s (Sampling frequency)	F_s = 1 / T_{td} = 24 MHz/500 = 48 kHz
Set differential input	afe_0xec<6> = 1
Set input channel	afe_0xeb = 0x56 Select PB[4] as positive input and PB[5] as negative input
Set reference voltage V_{REF}	afe_0xea<1:0> = 2 V_{REF} =1.2V
Set scaling factor for ADC analog input	afe_0xfa<7:6> = 0 scaling factor: 1 ADC maximum input range: -1.2V ~ +1.2V

Function	Register Setting
Set resolution	afe_0xec<1:0> = 3 resolution: 14 bits
Set T_{samp} (determines the speed to stabilize input before sampling)	afe_0xee<3:0> = 3 T_{samp} = adc_tsamp / F_{ADC_clk} = 12/4 MHz = 3 μs

17.4 Battery Voltage Sampling

The SoC use GPIO input for battery voltage sampling, by setting register afe_0xeb<7:4>, user can choose which GPIO port to use. Register afe_0xeb<3:0> should be set to 0xf.

NOTE:

- When IO voltage of GPIO is set to 1.8V, they cannot be used for battery voltage sampling. Users can use external voltage divider instead, the details refer to the Driver SDK Developer Handbook for this chip.
- When IO voltage of GPIO is set to 3.3V, the sampling range of VBAT is 2.0 ~ 4.3 V for high accuracy sampling. There is accuracy loss when VBAT is 1.8 ~ 2.0 V.

17.5 SAR ADC Register Description

The SAR ADC related registers are listed in the table below.

Table 17-2 SAR ADC Registers

Address	Default Value	Description
afe_0xea<1:0>	00	Select V_{REF} for M channel 0x0: 0.6V 0x1: 0.9V 0x2: 1.2V 0x3: rsvd
afe_0xea<7:2>	-	rsvd

Address	Default Value	Description
afe_0xeb<3:0>	0000	Select negative input for Misc channel: 0x0: No input 0x1: B[0] 0x2: B[1] ... 0x8: B[7] 0x9: C[4] 0xa: C[5] 0xb: rsvd 0xc: rsvd 0xd: rsvd 0xe: new tempsensor_n (Temperature sensor negative output) 0xf: Ground
afe_0xeb<7:4>	0000	Select positive input for Misc channel: 0x0: No input 0x1: B[0] 0x2: B[1] ... 0x8: B[7] 0x9: C[4] 0xa: C[5] 0xb: rsvd 0xc: rsvd 0xd: rsvd 0xe: new tempsensor_n (Temperature sensor negative output) 0xf: vbat
afe_0xec<1:0>	11	Set resolution for Misc channel 0x0: 8bits 0x1: 10bits 0x2: 12bits 0x3: 14bits
afe_0xec<5:2>	-	rsvd
afe_0xec<6>	0	Select input mode for Misc channel. 0: rsvd 1: differential mode
afe_0xec<7>	-	rsvd

Address	Default Value	Description
afe_0xee<3:0>	0000	Number of ADC clock cycles in sampling phase for Misc channel to stabilize the input before sampling: 0x0: 3 cycles 0x1: 6 cycles 0x2: 9 cycles 0x3: 12 cycles ... 0xf: 48 cycles
afe_0xef<7:0>	-	r_max_mc[9:0] serves to set length of "capture" state for Misc channel. r_max_s serves to set length of "set" state for Misc channel. Note: State length indicates number of 24M clock cycles occupied by the state.
afe_0xf0<7:0>	-	
afe_0xf1<3:0>	-	
afe_0xf1<5:4>	-	
afe_0xf1<7:6>	-	
afe_0xf2<0>	-	rsvd
afe_0xf2<1>	-	rsvd
afe_0xf2<2>	-	Enable Misc channel sampling. 1: enable
afe_0xf2<3>	0	0: enable write to core 1: disable write to core
afe_0xf2<5:4>	00	Set total length for sampling state machine (i.e. max state index)
afe_0xf2<7>	-	rsvd
afe_0xf3<0>	0	0: sample ADC data to afe_0xf8 and afe_0xf7 1: not sample ADC data to afe_0xf8 and afe_0xf7
afe_0xf3<1>	0	Dwa_en for analog
afe_0xf3<7:2>	-	rsvd
afe_0xf4<2:0>	011	ADC clock (derive from external 24M crystal) ADC clock frequency = 24M/(adc_clk_div+1)
afe_0xf4<7:3>-	-	rsvd
afe_0xf5<7:0>	-	rsvd
afe_0xf6<0>	-	[0]: vld, ADC data valid status bit (This bit is set as 1 at the end of capture state to indicate the ADC data is valid, and is cleared when set state starts.)

Address	Default Value	Description
afe_0xf6<7:1>	-	rsvd
afe_0xf7<7:0>	-	Read only [7:0]: Misc adc dat[7:0]
afe_0xf8<7:0>	-	Read only [7:0]: Misc adc_dat[15:8]
afe_0xf9<1:0>	-	rsvd
afe_0xf9<3:2>	0	Vbat divider select sel_vbat[1:0] Vbat 0x0 OFF 0x1 VBAT/4 0x2 VBAT/3 0x3 VBAT/2
afe_0xf9<5:4>	00	rsvd
afe_0xf9<7:6>	-	rsvd
afe_0xfa<1:0>	0	Comparator preamp bias current trimming itrim_preamp[1:0] Ibias 0x0 75% 0x1 100% 0x2 125% 0x3 150%
afe_0xfa<3:2>	0	Vref buffer bias current trimming itrim_vrefbuf[1:0] Ibias 0x0 75% 0x1 100% 0x2 125% 0x3 150%
afe_0xfa<5:4>	0	Vref buffer bias current trimming itrim_vcmbuf[1:0] Ibias 0x0 75% 0x1 100% 0x2 125% 0x3 150%

Address	Default Value	Description
afe_0xfa<7:6>	0	Analog input pre-scaling select sel_ai_scale[1:0]: scaling factor 0x0: 1 0x1: rsvd 0x2: 1/4 0x3: rsvd
afe_0xfc<4>	0	rsvd
afe_0xfc<5>	1	Power down ADC 1: Power down 0: Power up

18 Temperature Sensor

The SoC integrates a temperature sensor and it's used in combination with the SAR ADC to detect real-time temperature.

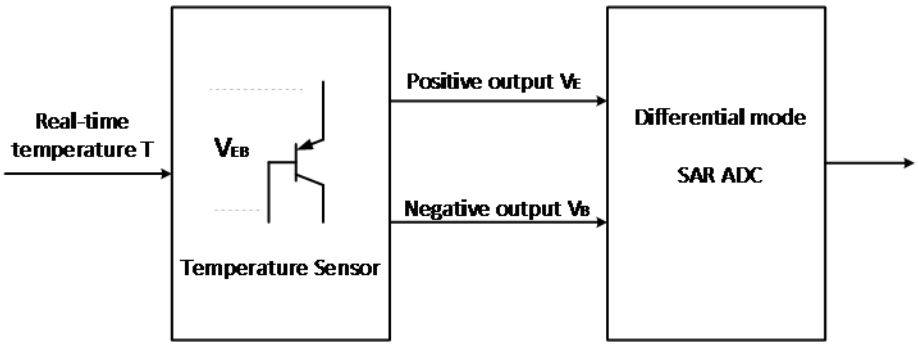
The temperature sensor is disabled by default. The analog register afe_0x06<2> should be set as 1b'0 to enable the temperature sensor.

Table 18-1 Analog Register for Temperature Sensor

Address	R/W	Description	Default Value
afe_0x06	R/W	[2]: pd_temp_sensor_3v, Power down of temp sensor: 1: Power down, 0: Power up	0x1

The temperature sensor embeds a pnp transistor. It takes the real-time temperature (T) as input, and outputs voltage drop (V_{EB}) signals of PNP transistor as positive and negative output respectively.

Figure 18-1 Block Diagram of Temperature Sensor



The voltage drop V_{EB} signals is determined by the real-time temperature T, as shown below:

$$V_{EB} = 884mV - 1.4286mV/℃ * (T - (-40℃))$$
$$= 884mV - 1.4286mV/℃ * (T + 40℃)$$

In this formula, "884mV" indicates the value of V_{EB} at the temperature of -40 .

To detect the temperature, the positive and negative output of the temperature sensor should be enabled as the input channels of the SAR ADC. The ADC converts the V_{EB} signals into digital signal.

The ADC should be configured as differential mode, and the positive and negative output of the temperature sensor should be configured as differential input of the ADC. The ADC should initiate one operation and obtain one output signal (ADCOUT); therefore,

$$V_{EB} = \frac{ADCOUT}{2^{N-1}} * V_{REF}.$$

In the formula, "N" and "V_{REF}" indicate the selected resolution and reference voltage of the SAR ADC.

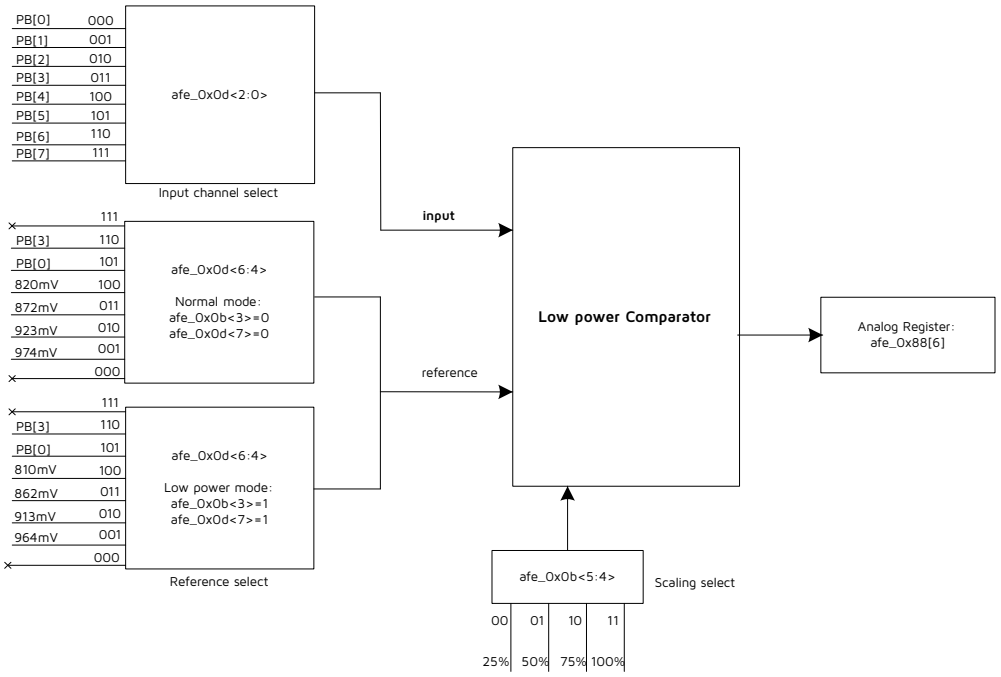
Then the real-time temperature T can be calculated according to the V_{EB}.

19 Low Power Comparator

The SoC embeds a low power comparator. This comparator takes two inputs: input derived from external PortB (PB[1]~PB[7]), and reference input derived from internal reference, PB[0], PB[3], or float.

By comparing the input voltage multiplied by selected scaling coefficient with reference input voltage, the low power comparator outputs high or low level accordingly.

Figure 19-1 Block Diagram of Low Power Comparator



NOTE: *The difference between the input level and the wake-up level needs to be greater than 100mV when using Low Power Comparator wake-up mode to enter sleep.*

19.1 Power On/Down

The low power comparator is powered down by default.

The analog register afe_0x07<3> serves to control power state of the low power comparator: By clearing this bit, this comparator is powered on; by setting this bit to 1'b1, this comparator is powered down.

To use the low power comparator, first set afe_0x07<3> as 1'b0, then the 32K RC clock source is enabled as the comparator clock.

19.2 Select Input Channel

Input channel is selectable from the PortB (PB[1]~PB[7]) via the analog register afe_0x0d<2:0>.

19.3 Select Mode and Input Channel for Reference

Generally, it's needed to clear both the afe_0x0b<3> and afe_0x0d<7> to select the normal mode. In normal mode, the internal reference is derived from Bandgap and has higher accuracy, but current bias is larger (10 μA); reference voltage input channel is selectable from internal reference of 974 mV, 923 mV, 872 mV and 820 mV, as well as PB[0], PB[3], and float.

To select the low power mode, both the afe_0x0b<3> and afe_0x0d<7> should be set as 1'b1. In low power mode, the internal reference is derived from UVLO and has lower accuracy, but current bias is decreased to 50 nA; reference voltage input channel is selectable from internal reference of 964 mV, 913 mV, 862 mV and 810 mV, as well as PB[0], PB[3], and float.

19.4 Select Scaling Coefficient

Equivalent reference voltage equals the selected reference input voltage divided by scaling coefficient.

The analog register afe_0x0b<5:4> serves to select one of the four scaling options: 25%, 50%, 75% and 100%.

19.5 Low Power Comparator Output

The low power comparator output is determined by the comparison result of the value of [input voltage *scaling] and reference voltage input. The comparison principle is shown as below:

- If the value of [input voltage *scaling] is larger than reference voltage input, the output is low ("0").
- If the value of [input voltage *scaling] is lower than reference voltage input, the output is high ("1").
- If the value of [input voltage *scaling] equals reference voltage input, or input channel is selected as float, the output is uncertain.

User can read the output of the low power comparator via the analog register afe_0x88[6].

The output of the low power comparator can be used as signal to wakeup system from low power modes.

19.6 Low Power Comparator Register Description

Table 19-1 Analog Register Related to Low Power Comparator

Address	Description	Default Value
afe_0x06<1>	Power down of low current comparator: 1: Power down 0: Power up	0x1
afe_0x0b<3>	Reference mode select: 1: ref from BG; 0: ref from UVLO.	0x1

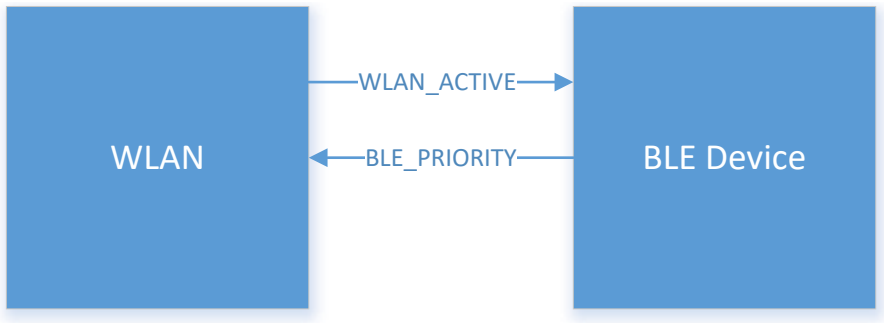
Address	Description	Default Value
afe_0x0b<5:4>	Reference voltage scaling: 11: 100% 10: 75% 01: 50% 00: 25%	0x1
afe_0x0c<3>	pd_diff, low power comparator diff mode disable: 1: single; 0: diff	0x1
afe_0x0d<2:0>	channel select of lc comparator: 000: B[0] 001: B[1] 010: B[2] 011: B[3] 100: B[4] 101: B[5] 110: B[6] 111: B[7]	000
afe_0x0d<3>	lc_comp_vbus_inen, inner detect point enable: 1: enable; 0: disable	000
afe_0x0d<6:4>	lc_comp_refsel<2:0>, channel select of lc comparator: reference from bg / reference from uvlo 0x000 -> float / 0x000 -> float 0x001 -> 974mV / 0x001 -> 1088mV 0x010 -> 923mV / 0x010 -> 1036mV 0x011 -> 872mV / 0x011 -> 983mV 0x100 -> 820mV / 0x100 -> 931mV 0x101 -> B[0] / 0x101 -> B[0] 0x110 -> B[3] / 0x110 -> B[3]	000
afe_0x0d<7>	lc_comp_pd_10u power down of 10u current to voltage reference 1: power down; 0: active	000
afe_0x4b<3>	comparator wakeup enable	0x0
afe_0x4d<0>	pd_lc_comp auto 1: auto power down low power comparator	0x0

20 PTA Interface

The SoC supports a Packet Traffic Arbitration (PTA) interface to facilitate co-existence with 802.11 WLAN. The SoC supports a 2/3/4-wire BLE PTA interface interface. Regarding the PTA's usage, the 2-wire BLE PTA can use any GPIO which has function of ble_activity plus any other GPIO; the 3-wire BLE PTA must use GPIO pins which are defined as ble_activity, ble_status and wlan_deny by the user, the 4-wire BLE PTA must use GPIO pins which are defined as ble_activity, ble_status and wlan_deny by the user plus any other GPIO. The detailed GPIO configuration refers to Table 11-1 GPIO Pad Function Mux.

20.1 BLE Two-Wire Signaling

Figure 20-1 BLE Two-Wire Signaling



WLAN_ACTIVE:

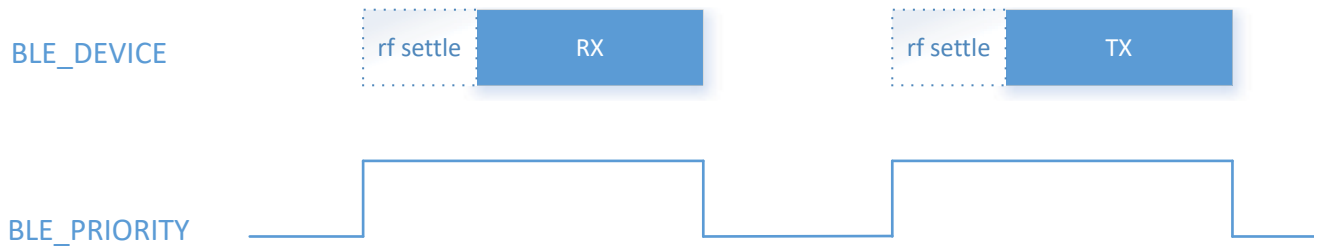
The WLAN_AVTIVE signal is asserted by WLAN controller when 802.11b/g packets are actively being transmitted or received. The BLE device avoids transmitting low-priority packets that are likely to cause interference with the 802.11b/g activity.

BLE_PRIORITY:

The BLE_PRIORITY signal should be asserted by the BLE device during high-priority transmit or receive activity. When this signal is asserted, WLAN device defers (or aborts) some or all of its transmissions.

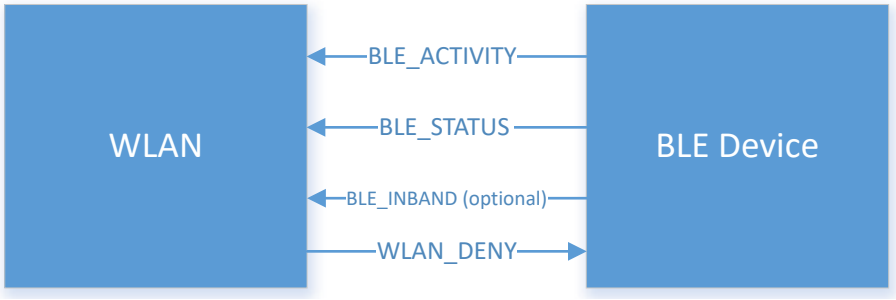
The WLAN_ACTIVE signal is judged by the software.

Figure 20-2 Example of BLE Two-Wire PTA Timing Diagram



20.2 BLE Three-Wire or Four-Wire Signaling

Figure 20-3 BLE Three-Wire or Four-Wire Signaling



BLE_ACTIVITY:

The BLE device should assert BLE_ACTIVITY for the duration of a "transaction". This usually corresponds to a transmit-receive or receive-transmit pair. This signal is asserted the time t1 before RF settle operation of first BLE RX/TX packet.

BLE_STATUS:

At the same time as asserting BLE_ACTIVE, the BLE device should assert BLE_STATUS if the transaction is considered to be high priority. After the time t2 the signal should be changed to indicate whether or not the BLE device is transmitting (asserted) or receiving (de-asserted). This signal must be updated prior to transmission or reception to indicate any change of direction.

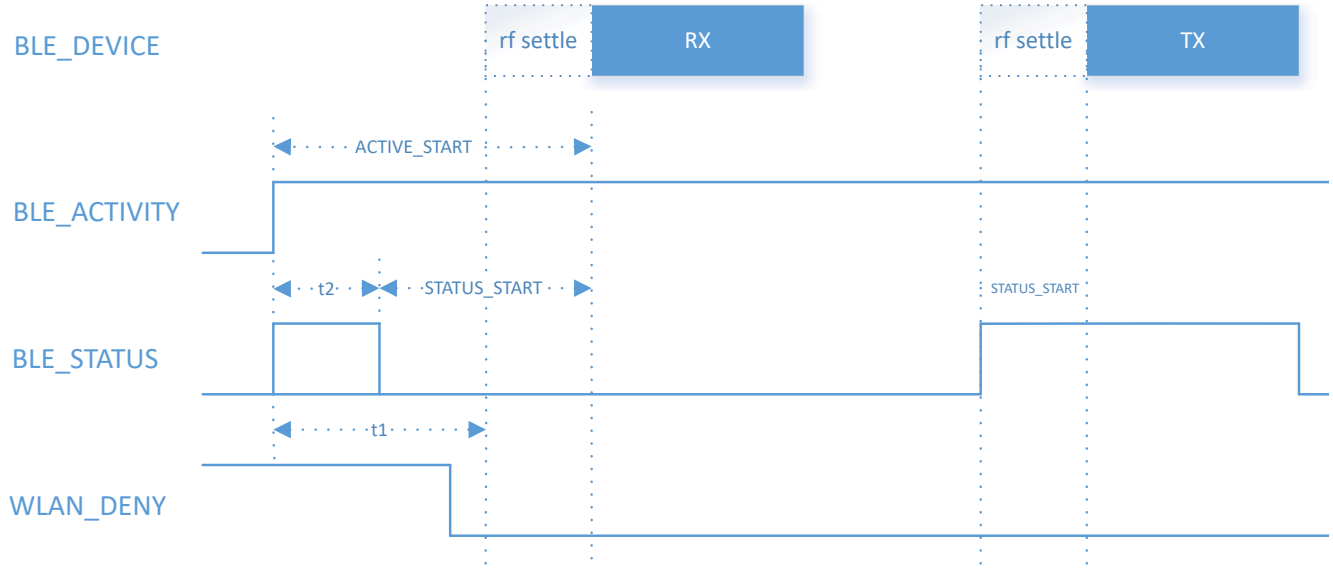
BLE_INBAND (optional):

This signal is optional and is only of benefit if there is sufficient isolation between the radios to support simultaneous operation on non-overlapping frequencies. The BLE device asserts BLE_INBAND (asserted by software) if any of the channels used in the transaction overlap the 802.11b/g frequencies.

WLAN_DENY:

The WLAN controller drives WLAN_DENY to indicate whether the requested BLE transaction is allowed or denied (which should be effective within the time t1 after asserting BLE_ACTIVE) to determine the activity direction. If the signal is asserted, the BLE device does not start the transaction.

Figure 20-4 Example of BLE Four-Wire PTA Timing Diagram



The two registers below are used to configure t1/t2:

Table 20-1 Register Configuration for t1/t2

Address	Name	R/W	Description	Default Value
0xf12	r_t_coex_t1	RW	[7:0]: Corresponds to t1 in Figure 20-4 above. Specifies the time after assertion of BLE_ACTIVITY signal at which the WLAN_DENY should be stable and is sampled by BLE device to determine whether to launch transaction The value of the register should be t1 - 1 (Unit: μs)	0x31
0xf13	r_t_coex_t2	RW	[7:0]: Corresponds to t2 in Figure 20-4 above. Specifies the time after assertion of the BLE_ACTIVITY signal at which the BLE_STATUS signal is changed from transaction priority to packet direction The value of the register should be t2 - 1 (Unit: μs)	0x13

21 Public Key Engine (PKE)

The SoC embeds Public Key Engine (PKE) Standard Performance acceleration module and this section describes its function and use.

21.1 Calculation Model Overview

The Public Key Engine (PKE) contains a low-power version of the public key cryptography acceleration engine, which can support a variety of asymmetric cryptographic algorithms. It should be noted that to fully implement SM2, ECDSA and ECDH functions, a random number generator module and a Hash module are required. In this version, the following features are available:

- Support modular operations: modular addition, modular subtraction, modular multiplication, modular exponentiation, modular inverse
- Support elliptic curve point operations: point addition, point doubling, point multiplication, and verify whether the point is on the curve
- Support large number operations: large number multiplication

With software drivers, it can support multiple public key algorithms, including:

- RSA (supports CRT): operand length 512 ~ 4096 bits, 32-bit step
- ECC (supports ECDH and ECDSA, prime field): 192, 224, 256 and 521 bits
- Ed25519/X25519
- SM2

21.2 Function Description

21.2.1 Module Description

PKE is designed to accelerate large number operations involved in RSA and Elliptic Curve Cryptography (ECC) operations in public key cryptography. Recently PKE can directly complete modular exponentiation in RSA and point multiplication in ECC. The CPU can query the operation of the PKE by polling or interrupting. The PKE includes one program memory unit (ROM), one instruction arithmetic unit (IEU), one 32-bit arithmetic unit (ALU), two pseudo-double-ended data RAMs, one register combination with interface module.

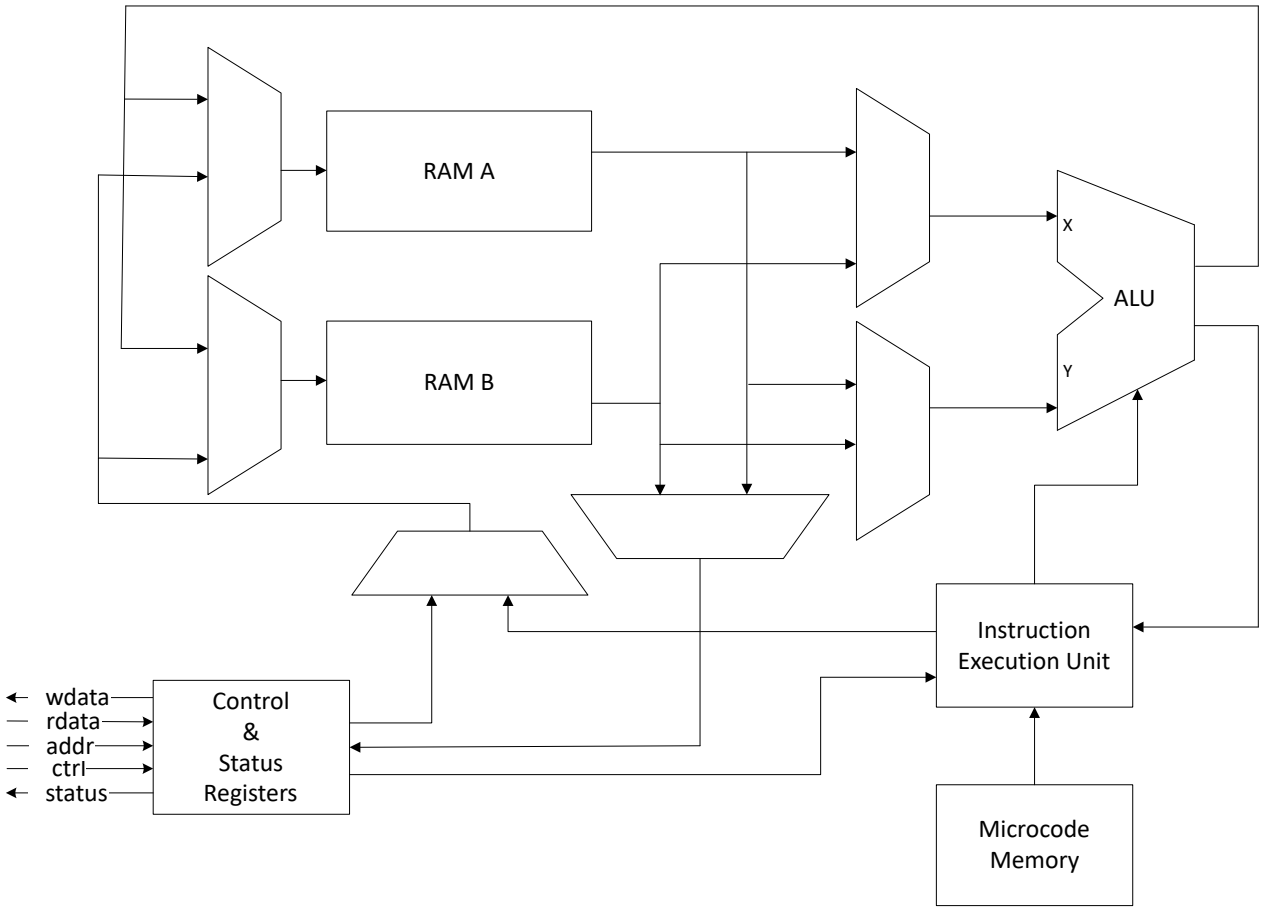
According to different register configurations, the PKE can complete the following operations of different precisions:

- RSA: length 512 ~ 4096 bits, 32-bit step
- ECC (prime field): length 192, 224, 256, and 521 bits
- Ed25519/X25519
- SM2

In addition, the calculation of the PKE is finished in the form of Microcode and the Microcode is stored in the program storage unit. Therefore, different kind of public key cryptographic calculations can be implemented by pouring different microcode into the program storage unit. For instance, a high security public key algorithm instruction can be injected into a program storage unit in the PKE module in a SoC with high security requirements. Certainly these arithmetic instructions can be written to the ROM with a large program memory

unit capacity. The CPU makes real-time calls according to different usage scenarios. The full microcode size is approximately 2 KB.

Figure 21-1 Block Diagram of PKE Module



21.2.2 Software Interface (Programming Model)

The interfaces of the PKE are all mapped into the bus address space. The block of address mapping space mainly contains all the operands that the CPU can access. These operands contain modulus, power exponents, partial intermediate variables, and so on. In addition to this, the address map also contains control and status registers. The CPU can configure and monitor the PKE module through these control and status registers.

In the operations supported by PKE, the operands are also 192 bits at minimum. Therefore, it encounters the problem of big-endian and little-endian when putting data into data RAM in the CPU or DMA. In the PKE module, words are arranged following an order of little-endian.

In PKE, the smallest operand is 32 bits (1 word), because the current ALU bit width input is 32 bits. If the operand is not word aligned, the high bit needs to be filled as 0.

After the PKE receives the start command, it starts the operation. During the operation, the host computer can query the current running state through the status register, or interrupt the current operation through the control register. In addition, the result of partial intermediate operations can be obtained by accessing the data RAM address.

The host computer can obtain the result of target operation finish by PKE through polling or interrupting. Data RAM supports word aligned and does not support byte alignment.

Table 21-1 Dual Port RAM Address Map

First Address of Operand	ECC			RSA			
	256 Bits	512 Bits	1024 Bits	512 Bits	1024 Bits	2048 Bits	4096 Bits
A0	0x0400	0x0400	0x0400	0x0400	0x0400	0x0400	0x400
A1	0x0424	0x0444	0x0484	0x0444	0x0484	0x0504	0x604
A2	0x0448	0x0488	0x0508	0x0488	0x0508	0x0608	0x808
A3	0x046C	0x04CC	0x058C	0x04CC	0x058C	0x070C	0xA0C
A4	0x0490	0x0510	0x0610	0x0510	0x0610	0x0810	0xC10
A5	0x04B4	0x0554	0x0694	-	-	-	-
A6	0x04D8	0x0598	0x0718	-	-	-	-
A7	0x04FC	0x05DC	0x079C	-	-	-	-
A8	0x0520	0x0620	0x0820	-	-	-	-
A9	0x0544	0x0664	0x08A4	-	-	-	-
B0	0x1000	0x1000	0x1000	0x1000	0x1000	0x1000	0x1000
B1	0x1024	0x1044	0x1084	0x1044	0x1084	0x1104	0x1204
B2	0x1048	0x1088	0x1108	0x1088	0x1108	0x1208	0x1408
B3	0x106C	0x10CC	0x118C	0x10CC	0x118C	0x130C	0x160C
B4	0x1090	0x1110	0x1210	0x1110	0x1210	0x1410	0x1810
B5	0x10B4	0x1154	0x1294	-	-	-	-
B6	0x10D8	0x1198	0x1318	-	-	-	-
B7	0x10FC	0x11DC	0x139C	-	-	-	-
B8	0x1120	0x1220	0x1420	-	-	-	-
B9	0x1144	0x1264	0x14A4	-	-	-	-

The above table shows the address assignment of two RAMs in ECC mode and RSA mode. The operand registers are distributed in two blocks of data RAM, using the prefixes A and B to distinguish the two blocks of RAM. The addresses listed in the table are all CPU addressable addresses, RAM A has an address offset of 0x400, and RAM B has an address offset of 0x1000. The actual space used by RAM is larger than the space listed in the table and some intermediate variable storage is not open to the CPU.

Data is stored in the mode of little-endian in RAM.

21.3 PKE Register Description

The base address for the following PKE related registers is 0x80110000.

Table 21-2 PKE Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	PKE_CTRL	W1S	[0]: Start Trigger PKE to start operation 0: No effect 1: PKE starts operation in the next clock cycle. The operations performed by PKE are decided by PKE_CFG and PKE_MC_PTR.	0x00
0x04	PKE_CFG0	WR	[7:0] partial_radix_lo {partial_radix_hi, partial_radix_lo} determines the bit width that the operation really needs to use in the operation. The value of this field indicates the number of bits, and the bit width of the operand is partial_radix. For example, if BASE_RADIX = 2 and PARTIAL_RADIX = 192 (0xC0), then the bit width of the operand is 192 bits. If you need to perform secp192r1 operations, you need to configure BASE_RADIX and PARTIAL_RADIX as shown in this example. If you want to use other bit-width operands, follow the above formula to configure BASE_RADIX and PARTIAL_RADIX. If the operands are all word-aligned, then the values of [15:11] are all 0x0. If you want to perform secp521r1 operation, then configure BASE_RADIX as 4, and PARTIAL_RADIX as 521 (0x209).	0x00
0x05	PKE_CFG1	RW	[4:0] partial_radix_hi	0x01
0x06	PKE_CFG2	RW	[2:0] base_radix This field indicates the bit width base of operations. At the same time, the base also indicates the space required for storing the operand in RAM. For RSA modular operations, the value of this field should be 4, 5 or 6 For ECC point operations, the value of this field should be 2, 3 or 4 2: 256 bits 3: 512 bits 4: 1024 bits 5: 2048 bits 6: 4096 bits Other: Reserved	0x02

Address Offset	Name	Type	Description	Default Value
0x08	PKE_MC_PTR0	RW	[7:0] addr_lo {addr_hi, addr_lo} PKE microcode execution entry This register can be rewritten only when PKE is not working; any write operation when PKE is working is ignored. During the operation of PKE, this field updates in real time; it always points to the address of the next instruction to be executed. It should be noted that the instructions are all word aligned. Therefore, the lowest 2 bits of this field are both 0.	0x00
0x09	PKE_MC_PTR1	RW	[3:0] addr_hi	0x00
0x0c	PKE_RISR	W0C	[0] core_ris CPU mode interrupt indicator 0: PKE generates no interrupts in CPU mode. 1: PKE has completed operations in CPU mode, interrupt generated.	0x00
0x10	PKE_IMCR	RW	[0] core_irqen Enable CPU mode interrupt 0: Disable PKE from generating interrupts in CPU mode 1: Enable PKE to generate interrupts in CPU mode	0x00
0x14	PKE_MISR	RO	[0] core_mi CPU mode interrupt output 0: PKE does not generate interrupts in CPU mode 1: PKE has generated interrupts in CPU mode	0x00
0x24	PKE_RT_CODE	RO	[3:0] stop_log This field is used to indicate the reason for PKE stop. If PKE is stopped because the operation is completed, then the value of this field is 0. If the value of this field is non-zero, the operation of PKE has not been completed and some exceptions have been encountered, which require external processing and the results are not available. [0]: Normal stop [1]: Termination request received (CTRL.STOP is high) [2]: No valid modular inverse result [3]: Point is not on the curve (CTRL.CMD: PVER) [4]: Invalid Microcode others: Reserved	0x00

Address Offset	Name	Type	Description	Default Value
0x50	PKE_EXE_CFG	RW	[0] iaff_r0 Enable R0 input's affine coordinate system form, this bit is only valid for ECC operations In ECC operations, R0 is the position of A0, A1 and B2 In RSA operations, R0 is the position of A0 0: The input point is a point in the Jacobian coordinate system; when it involves modular multiplication, if the bit is low, the point in its scope is converted to the Jacobian coordinate system before the operation 1: The input point is a point on the affine coordinate system	0x15
			[1] imon_r0 Enable R0 input's Montgomery form In ECC operations, R0 is the position of A0, A1 and B2 In RSA operations, R0 is the position of A0 0: The input data is in ordinary form; when it comes to modular multiplication, if the bit is low, the number in its scope is converted to Montgomery form before the operation 1: The input data is in Montgomery form	
			[2] iaff_r1 Enable R1 input's affine coordinate system form, this bit is only valid for ECC operations. In ECC operations, R1 is the position of B0, B1 and A2 In RSA operations, R1 is the B0 position 0: The input point is a point on the Jacobian coordinate system; when it involves modular multiplication, if the bit is low, the point in its scope is converted to the Jacobian coordinate system before the operation 1: The input point is a point on the affine coordinate system	

Address Offset	Name	Type	Description	Default Value
0x50	PKE_EXE_CFG	RW	[3] imon_r1 Enable R1 input's Montgomery form In ECC operations, R1 is the position of B0, B1 and A2 In RSA operations, R1 is the B0 position 0: The input data is in normal form; when it comes to modular multiplication, if the bit is low, the number in its scope is converted to Montgomery form before the operation 1: The input data is in Montgomery form	
			[4] oaff Enable output's affine coordinate system form, this bit is only valid for ECC operations 0: The output point is a point on the Jacobian coordinate system 1: The output point is a point on the affine coordinate system	
			[5] omon Enable output's Montgomery form 0: The output result is in normal form 1: The output result is in Montgomery form	
0xfc	PKE_RBG_VERSION0	R	[3:0] mir: Sub version number. [7:4] mar: Main version number	0x10
0xfe	PKE_RBG_VERSION2	R	[7:0] project_lo {project_hi, project_lo} Project number	0x06
0xff	PKE_RBG_VERSION3	R	[7:0] project_hi	0xef

22 True Random Number Generator (TRNG)

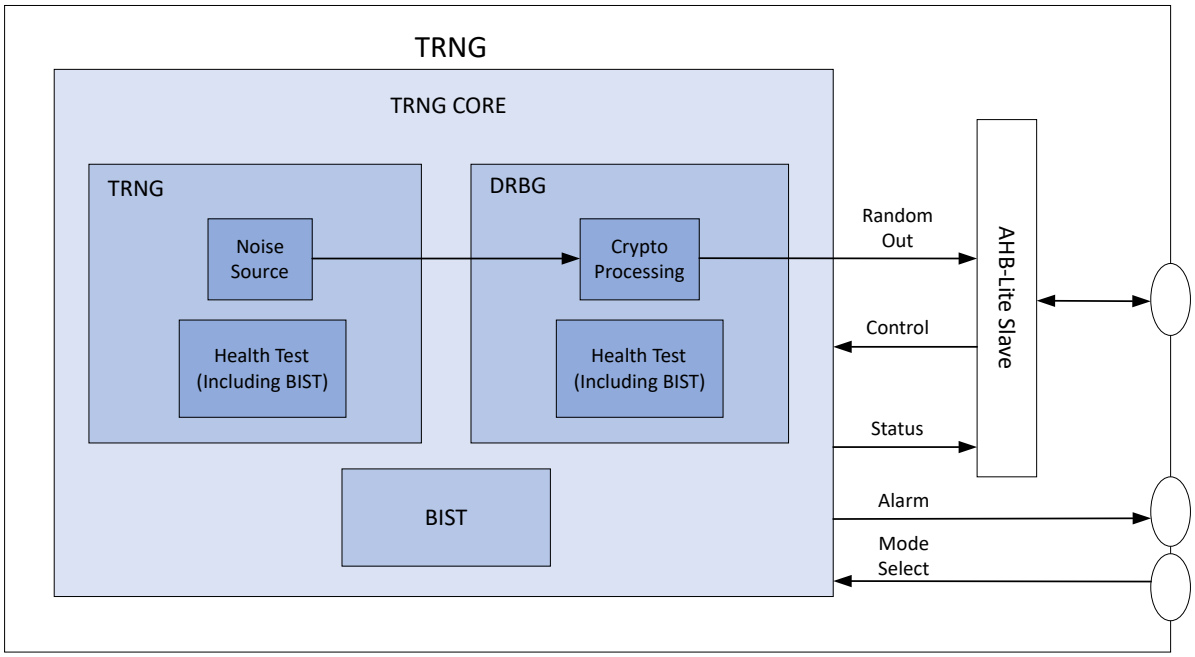
22.1 Module Overview

The True Random Number Generator (TRNG) module contains both a true random number generator and a pseudo-random number generator, and features include:

- Meets NIST SP800-90 a/b/c and GM 0008-2012 design requirements.
- Post-processing algorithm supports CTR-SM4.
- Output random numbers pass NIST SP800-22 and GM 0005-2012 tests.
- Supports online health monitoring of random numbers.
- Supports interrupts.
- Supports shutting down the TRNG module to save power consumption.

Its top-level block diagram is shown below.

Figure 22-1 Block Diagram of TRNG



22.2 Function Description

22.2.1 Function Overview

After power-on reset, TRNG is on by default; the entropy source is Ring Oscillator (RO), which is all on by default; the sampling clock frequency is the lowest; the default sampling clock is controlled by the register ROSR_TRNG_CTRL4.trng_sclk_sel, and it is recommended to use the internal RO CLK for sampling. After reset, if the power-on self-test is not skipped, that is, the register ROSR_TRNG_CTRL4.trng_skip_startup is 0, the TRNG enters the self-test, and starts generating random numbers after completion. When the system detects an interrupt or the query register TRNG_SR.DRDY is equal to 1, it can read 8*32 bits random numbers from the

TRNG data output register TRNG_DR, that is, read the TRNG_DR register 8 times. Before changing the TRNG configuration, it is needed to set TRNG enable register TRNG_CR.RNGEN to 0.

The TERO RNG is an independent part, and the control, status and data registers are all independent registers. After power-on reset, the TERO RNG is on by default, all 4 TERO entropy sources are turned on by default, and the TERO data output register TERO_DR0 outputs random numbers. When the system detects that the TERO status register TERO_SR.DR is equal to 1, the 32-bit random number can be read from TERO_DR.

22.2.2 Interrupt Description

There are three sources of interrupts for the TRNG module:

- CPU reads TRNG_DR without data
- Data valid
- Errors in online detection

The above interrupts can be set by RBG_CR. By default, the data valid interrupt is enabled.

When the TRNG_CR.RNGEN is low, the interrupt signal is not cleared. Therefore, before enabling TRNG_CR.RNGEN, it is necessary to ensure that there is no previous interrupt signal, otherwise it affects the next interrupt.

22.2.2.1 CPU Reads TRNG_DR without Data

To avoid the CPU reading invalid data, the CPU should query the TRNG for valid random numbers by reading TRNG_SR.DRDY before reading the data to TRNG_DR.

The CPU can clear the interrupt by writing 1 to the register TRNG_SR.ERERR. If the write is successful, the interrupt is cleared. When the above situation occurs again, the interrupt is valid again.

22.2.2.2 Data Valid

The TRNG provides two ways to output data. Up to eight 32-bit random numbers can be read by interrupt when interrupt enable is active; or data readiness can be determined by querying register TRNG_SR.DRDY. If the CPU's rate of handling random numbers is slower than the TRNG random number generation rate, it is usually not recommended to use interrupts to obtain random numbers.

22.2.2.3 Errors in Online Detection

In normal operation mode, two phases are performed inside the TRNG module: the initialization phase and the working phase. In the initialization phase, the TRBG module and the DRBG module performs BIST respectively. At the same time, the TRBG module also pre-generates some random numbers to test the randomness of these random numbers, and during the process, if the test fails, the TRNG module generates an interrupt to notify the CPU.

In normal operation mode, the online detection module can be turned on to monitor the quality of the random numbers generated by the TRBG module. If the online detection fails, it indicates that there is a problem with the randomness of the random numbers generated by the TRBG.

Optionally, an online monitoring module can be added after the DRBG to monitor the quality of the random numbers generated by the DRBG module.

The CPU can clear this interrupt by writing 1 to register TRNG_SR.HTF. After clearing, the TRNG still does not work, it is needed to write 0 to register TRNG_CR.RNGEN to stop it, reconfigure it and start it again. The TRNG can also be reset globally to make it work again.

22.2.3 Operation Procedure

22.2.3.1 Normal Operation

After the CPU is working normally, the TRNG module can be turned off first, that is, setting register TRNG_CR.RNGEN to 0. After that, it can be configured, and after the configuration is completed, it can be written 1 to register TRNG_CR.RNGEN to make it work normally. The CPU can configure it by configuring optional configuration registers such as TRNG_CR, TRNG_MSEL, and so on. When register TRNG_CR.RNGEN is written 1, modifying the value of the above registers has no effect on TRNG. Therefore, for configuration, it is needed to set register TRNG_CR.RNGEN last to turn on the TRNG module after the other registers are configured. During operation, TRNG_MSEL can be modified to switch between TRBG and DRBG to meet different usage environments.

22.2.3.2 Entropy Source

The random number generator module in TRNG contains two different random number generators, RO RNG and TERO RNG, which are completely independent of each other, where RO RNG is the main random number generator and contains the BIST, post-processing and on-line detection modules and is connected to the random clock, while TERO occupies only four independent registers and does not contain post-processing and on-line detection modules.

The RO RNG has four mutually independent RO entropy sources, and each entropy source can choose to adopt the self-contained RO CLK as the sampling clock, or can choose hclk as the sampling clock, the choice is determined by register ROSR_TRNG_CTRL4.trng_sclk_sel, with high as hclk and low as the internal RO CLK. Meanwhile, the enable signals of all ROs are open. Some of these ROs can be turned on or off for testing by the control register.

The TERO RNG is independent of the RO RNG and can be used alone or as a backup entropy source. The corresponding control registers are TERO_CR, TERO_THOLD, TERO_SR and TERO_DR.

Under the default configuration, the value of TERO_THOLD register is 0x000001F4, that is, the number of oscillations of TERO is controlled between 0 and 500, and all four TEROs are turned on, and the corresponding control bits are TERO_CR.TEROEN. When the state of TERO_SR is 1, it means that the TERO random number is ready, and it can read TERO_DR once to obtain a 32-bit random number. In addition, set TERO_CR.MS to 1, you can read the number of oscillations of TERO through TERO_DR. It should be noted that each register can only output the number of oscillations of two TEROs, therfore viewing the number of oscillations of different TEROs needs to select through TERO_CR.OSEL.

22.3 TRNG Register Description

The TRNG related registers are listed in the following table. The base address for the following registers is 0x80103000.

Table 22-1 TRNG Related Register

Address Offset	Name	Type	Description	Reset Value
0x00	TRNG_CR0	RW	[0]:RNGEN, random number generator enable; [4]:ROS3EN, RO entropy source 3 enable; [5]:ROS2EN, RO entropy source 2 enable; [6]:ROS1EN, RO entropy source 1 enable; [7]:ROS0EN, RO entropy source 0 enable;	0xf1
0x02	TRNG_CR2	RW	[0]:DIEN, data interrupt enable; [1]:ERIEN, null read interrupt enable;	0x01
0x03	TRNG_CR3	RW	[0]:IRQEN, global interrupt enable;	0x01
0x04	TRNG_MSEL0	RW	[0]:MSEL, mode select. 0: TRNG output register output true random number 1: TRNG output register output pseudo-random number	0x00
0x08	TRNG_SR0	W1C	[0]:HTF, health test failure; [1]:DRDY, data ready; [2]:ERERR, null read error;	0x00
0x0c	TRNG_DR0	RO	[7:0]:RDATA, data output	0x00
0x0d	TRNG_DR1	RO	[7:0]:RDATA, data output	0x00
0x0e	TRNG_DR2	RO	[7:0]:RDATA, data output	0x00
0x0f	TRNG_DR3	RO	[7:0]:RDATA, data output	0x00
0x10	TRNG_VERSION0	RO	[3:0]: MIR, Sub version number [7:4]: MAR, Main version number	0x10
0x12	TRNG_VERSION2	RO	[7:0]:PROJECT0	0x0d
0x13	TRNG_VERSION3	RO	[7:0]:PROJECT1	0x00
0x40	TRNG_RESEED0	W1S	[0]:RSED, Reseed operation triggered. 0: Reseed operation is triggered according to the built-in counter. 1: Trigger DRBG Reseed operation manually.	0x00

Address Offset	Name	Type	Description	Reset Value
0x60	TRNG_HT_CR0	RW	[0]:DRPTEN ,DRBG output repeatability test [1]:DRCTEN ,DRBG repeat count test enable [4]:TRATEN ,TRBG adaptive scale test enable [5]:TRCTEN ,TRBG repeat count test enable	0x33
0x70	TRNG_HT_SR0	RO	[0]:DRPTBF, DRBG output repeatability test BIST test failed; [1]:DRCTBF, DRBG repeatability count test BIST test failed;	0x00
0x71	TRNG_HT_SR1	RO	[0]:TAPTNBBF, TRBG non-binary adaptive scale test BIST test failed; [1]:TAPTBBF, TRBG Binary Adaptive Scaling Test BIST Test Failed; [2]:TRCTBF, TRBG Repeat Count Test BIST Test Failed;	0x00
0x72	TRNG_HT_SR2	RO	[0]:DRPTF, DRBG output repeatability test failed; [1]:DRCTF, DRBG repeatability count test failed;	0x00
0x73	TRNG_HT_SR3	RO	[0]:TAPTF, TRBG adaptive scaling test failed; [1]:TRCTF, TRBG repeat count test failed;	0x00
0x80	RO_SRC_EN1_0	RW	[7:0]: RO enable of RO entropy source 2. Each bit controls one RO. In total, there are 16 ROs in RO source 2.	0xff
0x81	RO_SRC_EN1_1	RW	[7:0]: RO enable of RO entropy source 2. Each bit controls one RO. In total, there are 16 ROs in RO source 2.	0xff
0x82	RO_SRC_EN1_2	RW	[7:0]: RO enable of RO entropy source 1. Each bit controls one RO. In total, there are 16 ROs in RO source 1.	0xff
0x83	RO_SRC_EN1_3	RW	[7:0]: RO enable of RO entropy source 1. Each bit controls one RO. In total, there are 16 ROs in RO source 1.	0xff
0x84	RO_SRC_EN2_0	RW	[7:0]: RO enable of RO entropy source 4. Each bit controls one RO. In total, there are 16 ROs in RO source 4.	0xff
0x85	RO_SRC_EN2_1	RW	[7:0]: RO enable of RO entropy source 4. Each bit controls one RO. In total, there are 16 ROs in RO source 4.	0xff
0x86	RO_SRC_EN2_2	RW	[7:0]: RO enable of RO entropy source 3. Each bit controls one RO. In total, there are 16 ROs in RO source 3.	0xff
0x87	RO_SRC_EN2_3	RW	[7:0]: RO enable of RO entropy source 3. Each bit controls one RO. In total, there are 16 ROs in RO source 3.	0xff

Address Offset	Name	Type	Description	Reset Value
0x88	SCLK_FREQ_0	RW	[1:0]:FSEL, sampling clock frequency configuration. The slower the sampling clock frequency, the slower the random number output of the true random number part, and the better the randomness. 00: Sampling clock frequency is 1/4 of the input clock. 01: Sampling clock frequency is 1/8 of the input clock. 10: Sampling clock frequency is 1/16 of the input clock 11: Sampling clock frequency is 1/32 of the input clock	0x03
0xb0	TERO_CR_0	RW	[0]:EN,TERO RNG enable. [1]:MS, select TERO RNG outputs 0: Random number 1: number of oscillations of TERO [2]:OSEL, valid only when MS is 1 0: Oscillation count of TERO 1&2 is selected as output. 1: Select the output to be the oscillation count of TERO 3&4.	0x01
0xb1	TERO_CR_1	RW	[3:0]: TEROEN, each bit enables 1 TERO entropy source	0x0f
0xb3	TERO_CR_3	RW	[7:0]:COTV, Cutoff Threshold, When the number of system clocks held by the TERO counter reaches this threshold, it is assumed that the TERO has been stopped.	0x07
0xb4	TERO_THOLD_0	RW	[7:0]:UBTERO_B0, Upper limit of TERO oscillations, maximum acceptable number of TERO oscillations, TERO automatically adjusts the length above this value.	0xf4
0xb5	TERO_THOLD_1	RW	[7:0]:UBTERO_B1, Upper limit of TERO oscillations, maximum acceptable number of TERO oscillations, TERO automatically adjusts the length above this value.	0x01
0xb6	TERO_THOLD_2	RW	[7:0]:LBTERO_B0, Lower limit of the number of TERO oscillations, Minimum acceptable number of TERO oscillations below which TERO automatically adjusts the length	0x00
0xb7	TERO_THOLD_3	RW	[7:0]:LBTERO_B1, Lower limit of the number of TERO oscillations, Minimum acceptable number of TERO oscillations below which TERO automatically adjusts the length	0x00
0xc0	TERO_CNT_0	RO	[7:0]:NEGCNT_B0, TERO entropy source 1 oscillation falling edge count detection	0x00

Address Offset	Name	Type	Description	Reset Value
0xc1	TERO_CNT_1	RO	[7:0]:NEGCNT_B1, TERO entropy source 1 oscillation falling edge count detection	0x00
0xc2	TERO_CNT_2	RO	[7:0]:POSCNT_B0, TERO entropy source 1 oscillation rising edge count detection	0x00
0xc3	TERO_CNT_3	RO	[7:0]:POSCNT_B1, TERO entropy source 1 oscillation rising edge count detection	0x00
0xd0	TERO_SR_0	RO	[0]:DR, data ready 0: data is being prepared 1: data is ready	0x00
0xd4	TERO_DR_0	RO	[7:0]:RND, data output Depending on the selection of TERO_CR.MS, this register outputs a random number or the number of TERO oscillations	0x00
0xd5	TERO_DR_1	RO	[7:0]:RND, data output Depending on the selection of TERO_CR.MS, this register outputs a random number or the number of TERO oscillations	0x00
0xd6	TERO_DR_2	RO	[7:0]:RND, data output Depending on the selection of TERO_CR.MS, this register outputs a random number or the number of TERO oscillations	0x00
0xd7	TERO_DR_3	RO	[7:0]:RND, data output Depending on the selection of TERO_CR.MS, this register outputs a random number or the number of TERO oscillations	0x00
0xe0	TERO_RCR_0	RO	[7:0]:TCONF, actual configured value of TERO entropy source 1	0x00
0xe1	TERO_RCR_1	RO	[7:0]:TCONF, actual configured value of TERO entropy source 1	0x00
0xe2	TERO_RCR_2	RO	[7:0]:TCONF, actual configured value of TERO entropy source 1	0x00
0xe3	TERO_RCR_3	RO	[7:0]:TCONF, actual configured value of TERO entropy source 1	0x00

The additional TRNG related registers are listed in the following table. The base address for the following registers is 0x80100800.

Table 22-2 Additional TRNG Related Registers

Address Offset	Name	Type	Description	Reset Value
0x00	ROSR_CTRL0	RO	[7:0]:trng_fifo_b0, TRNG DMA mode read data entry.	0x00

Address Offset	Name	Type	Description	Reset Value
0x01	ROSR_CTRL1	RO	[7:0]:trng_fifo_b1, TRNG DMA mode read data entry.	0x00
0x02	ROSR_CTRL2	RO	[7:0]:trng_fifo_b2, TRNG DMA mode read data entry.	0x00
0x03	ROSR_CTRL3	RO	[7:0]:trng_fifo_b3, TRNG DMA mode read data entry.	0x00
0x04	ROSR_CTRL4	RW	[0]:trng_rxdma_en [1]:trng_skip_startup, skip power-up detection process, valid high 0: no effect 1: Skip power-up self-test [2]:trng_sclk_sel, select sample clock source 0: Asynchronous RO sampling clock generated internally by TRNG 1: hclk [3]:trng_irq_en [4]:trng_irq_alarm_en	0x02
0x05	ROSR_TRL5	RO	[0]:trng_irq_status [1]:trng_alarm_status [2]:trng_rdy_status	0x00

23 Symmetric Key Engine (SKE)

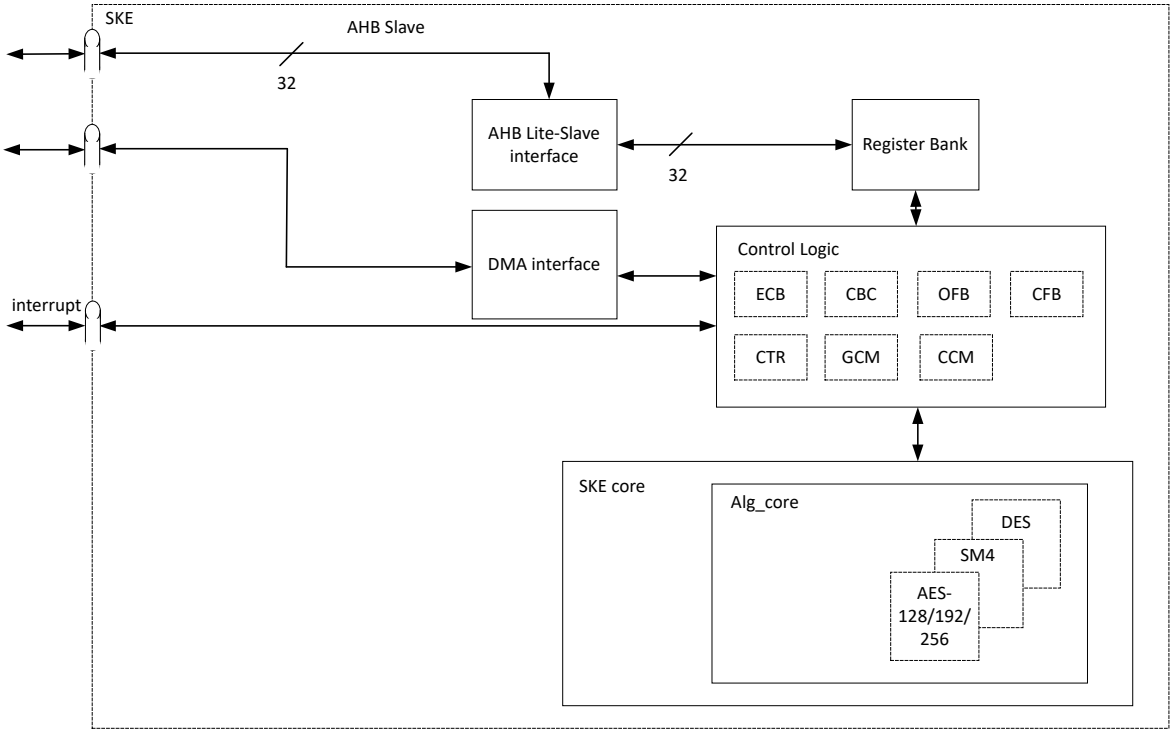
23.1 Calculation Model Overview

The SKE module contains a low-power symmetric encryption algorithm engine, and its features include:

- Supports algorithm of AES-128, AES-192, AES-256, DES, SM4
- Supports hardware of ECB, CBC, CTR, CFB, OFB, GCM, CCM
- Supports hardware of DMA mode
- Supports polling mode or interrupt mode for the host computer to query the status

The top-level block diagram is shown as below.

Figure 23-1 Block Diagram of SKE Module



23.2 Function Description

23.2.1 Function Overview

After power-on reset, the SKE is in an idle state. At this time, the SKE waits for external writing of configuration and data. Before calling SKE to perform the operation, the host computer needs to write the data and configuration required for the operation into the corresponding registers, and then trigger SKE to perform the operation. During the operation of SKE, the host computer can query the status by polling to determine whether the operation is finished or not.

The SKE supports CPU mode and DMA mode for data handling.

In the CPU mode, all data are carried to SKE by the host computer via the AHB bus, and after the operation is finished, the host computer reads the result from SKE via the AHB bus and stores it in the system memory unit. In this mode, the SKE can receive one packet of data at a time for arithmetic. If the length of a block of data is greater than the length of that algorithmic grouping, the host computer needs to slice the message into integer algorithmic groupings and feed each grouping in turn into the SKE module for computation.

In DMA mode, all data is placed at the DMA source address. Before calling DMA, it is needed to update the relevant configuration in CPU mode, then configure the relevant information of DMA, enable the DMA, then start DMA operation and wait for the end.

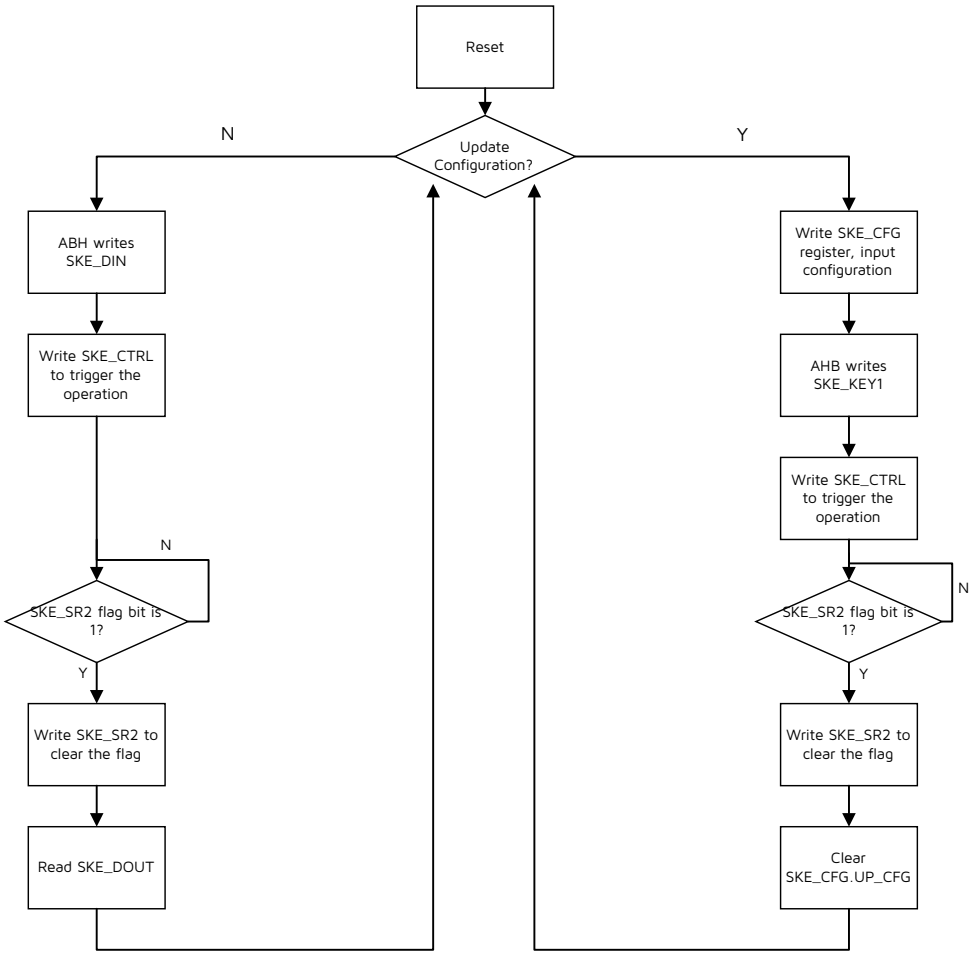
23.2.2 Status Description

The SKE_SR1.BUSY is the busy flag bit of the SKE. In CPU mode, when SKE is triggered to start an operation, this flag bit is set to 1 until it jumps to 0 at the end of the operation.

The SKE_SR2.CORE_DONE is the end-of-operation flag bit of the SKE CPU mode. When the SKE completes the CPU mode arithmetic, this bit jumps to 1, signaling the end of the arithmetic. The host computer can poll this bit until it jumps to 1 after triggering an SKE operation.

23.2.3 Usage Flow

Figure 23-2 Usage Flow of SKE Module



The recommended usage flow of SKE module is shown as above.

- After power-on reset, SKE is idle and waiting to be invoked.
- Judge whether the configuration in SKE needs to be updated, if yes, then carry out the configuration update process, otherwise carry out the data encryption and decryption process; in the case of not updating the configuration, the SKE uses the last saved configuration for the operation.
- When updating the configuration
  - First write the required configuration to SKE_CFG, the SKE_CFG.UP_CFG should be guaranteed to be 1;
  - Then write key 1 via AHB, it is not necessary to input the key every time. If the key has been input before, there is no need to input the same key again, the SKE uses the previously input key for the operation.
  - Write 1 to SKE_CTRL to trigger SKE to perform operation.
  - Poll the flag bit corresponding to SKE_SR2 until the flag bit is 1.
  - Write operation to SKE_SR2 to clear the flag bit when it detects a 1 in SKE_SR2.
  - Write 0 to SKE_CFG.UP_CFG; thus, the configuration update process ends.
- When data encryption and decryption
  - Write 1 to SKE_CTRL to trigger SKE to perform operation.
  - Poll the flag bit corresponding to SKE_SR2 until the flag bit is 1.
  - Write to SKE_SR2 to clear the flag bit when it detects a 1 in SKE_SR2.
  - Read SKE_DOUT to get the result of the operation.
- To this point, SKE has completed a complete call, you can loop the above steps for multiple calls.

23.3 SKE Register Description

The SKE related registers are listed in the following table. The base address for the following PKE related registers is 0x80104000.

Table 23-1 PKE Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	SKE_CTRL_B0	W1S	[0]: Start, trigger PKE to start operation 0: No effect 1: When SKE_CFG.UP_CFG is 1, the SKE is initiated to update the configuration without data operation; when SKE_CFG.UP_CFG is 0, the SKE is initiated to carry out data operation, and the value of SKE_DIN is used for data input.	0x00

Address Offset	Name	Type	Description	Default Value
0x04	SKE_CFG_B0	RW	[3:0]: ALG, algorithm selection 1: AES 2: SM4 3: DES 4: AES-192 5: AES-256 Other: reserved values	0x00
0x05	SKE_CFG_B1	RW	[3]: DEC, encryption/decryption selection 0: Encrypt 1: decrypt [4]: UP_CFG, update configuration 0: not update the configuration, SKE uses the last input configuration for the operation. 1: update configuration, SKE uses the new input configuration for the operation. Configuration contains the values of ALG, DEC and SKE_KEY1 registers in SKE_CFG.	0x00
0x06	SKE_CFG_B2	RW	[0]: dmaen, DMA enable [1]: irqen, interrupt enable	0x00

Address Offset	Name	Type	Description	Default Value
0x07	SKE_CFG_B3	RW	[1:0]: data_type, data format Data exchange for SKE_DIN, SKE_DOUT data 0: No exchange 1: Halfword exchange 2: Byte exchange 3: Bit exchange [7:4]: mode, mode selection 1: ECB 3: CBC 4: CFB 5: OFB 6: CTR 9: GCM 10: CCM Others: ECB	0x00
0x08	SKE_SR1_B0	RW	[0]: busy, busy flag bit 1: SKE is in running state, cannot receive instruction or data input; 0: SKE is in idle state, can receive instructions or data inputs.	0x00
0x0c	SKE_SR2_B0	W0C	[0]: core_done, CPU interrupt indicator bit 0: SKE_LP CPU mode no interrupt generation 1: SKE_LP CPU mode operation completed, interrupt generated	0x00
0x10	SKE_KEY1_B0	RW	[7:0]: key1[7:0], Key 1 Input In ECB mode, this register value is used as the encryption and decryption key The key needs to be entered when SKE_CTRL.UP_CFG is 1 to be effective	0x00
0x11	SKE_KEY1_B1	RW	[7:0]: key1[15:8], Key 1 Input	0x00
0x12	SKE_KEY1_B2	RW	[7:0]: key1[23:16], Key 1 Input	0x00
0x13	SKE_KEY1_B3	RW	[7:0]: key1[31:24], Key 1 Input	0x00
0x14	SKE_KEY1_B4	RW	[7:0]: key1[39:32], Key 1 Input	0x00

Address Offset	Name	Type	Description	Default Value
0x15	SKE_KEY1_B5	RW	[7:0]: key1[47:40], Key 1 Input	0x00
0x16	SKE_KEY1_B6	RW	[7:0]: key1[55:48], Key 1 Input	0x00
0x17	SKE_KEY1_B7	RW	[7:0]: key1[63:56], Key 1 Input	0x00
0x18	SKE_KEY1_B8	RW	[7:0]: key1[71:64], Key 1 Input	0x00
0x19	SKE_KEY1_B9	RW	[7:0]: key1[79:72], Key 1 Input	0x00
0x1a	SKE_KEY1_B10	RW	[7:0]: key1[87:80], Key 1 Input	0x00
0x1b	SKE_KEY1_B11	RW	[7:0]: key1[95:88], Key 1 Input	0x00
0x1c	SKE_KEY1_B12	RW	[7:0]: key1[103:96], Key 1 Input	0x00
0x1d	SKE_KEY1_B13	RW	[7:0]: key1[111:104], Key 1 Input	0x00
0x1e	SKE_KEY1_B14	RW	[7:0]: key1[119:112], Key 1 Input	0x00
0x1f	SKE_KEY1_B15	RW	[7:0]: key1[127:120], Key 1 Input	0x00
0x20	SKE_KEY1_B16	RW	[7:0]: key1[135:128], Key 1 Input	0x00
0x21	SKE_KEY1_B17	RW	[7:0]: key1[143:136], Key 1 Input	0x00
0x22	SKE_KEY1_B18	RW	[7:0]: key1[151:144], Key 1 Input	0x00
0x23	SKE_KEY1_B19	RW	[7:0]: key1[159:152], Key 1 Input	0x00
0x24	SKE_KEY1_B20	RW	[7:0]: key1[167:160], Key 1 Input	0x00
0x25	SKE_KEY1_B21	RW	[7:0]: key1[175:168], Key 1 Input	0x00
0x26	SKE_KEY1_B22	RW	[7:0]: key1[183:176], Key 1 Input	0x00
0x27	SKE_KEY1_B23	RW	[7:0]: key1[191:184], Key 1 Input	0x00
0x28	SKE_KEY1_B24	RW	[7:0]: key1[199:192], Key 1 Input	0x00
0x29	SKE_KEY1_B25	RW	[7:0]: key1[207:200], Key 1 Input	0x00
0x2a	SKE_KEY1_B26	RW	[7:0]: key1[215:208], Key 1 Input	0x00
0x2b	SKE_KEY1_B27	RW	[7:0]: key1[223:216], Key 1 Input	0x00
0x2c	SKE_KEY1_B28	RW	[7:0]: key1[231:224], Key 1 Input	0x00
0x2d	SKE_KEY1_B29	RW	[7:0]: key1[239:232], Key 1 Input	0x00
0x2e	SKE_KEY1_B30	RW	[7:0]: key1[247:240], Key 1 Input	0x00
0x2f	SKE_KEY1_B31	RW	[7:0]: key1[255:248], Key 1 Input	0x00

Address Offset	Name	Type	Description	Default Value
0x60	SKE_AAD_B0	RW	[7:0]: a_len[7:0], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x61	SKE_AAD_B1	RW	[7:0]: a_len[15:8], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x62	SKE_AAD_B2	RW	[7:0]: a_len[23:16], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x63	SKE_AAD_B3	RW	[7:0]: a_len[31:24], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x64	SKE_AAD_B4	RW	[7:0]: a_len[39:32], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x65	SKE_AAD_B5	RW	[7:0]: a_len[47:40], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x66	SKE_AAD_B6	RW	[7:0]: a_len[55:48], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x67	SKE_AAD_B7	RW	[7:0]: a_len[63:56], AAD bit length This register is valid in GCM, CCM mode.	0x00
0x68	SKE_CLEN_B0	RW	[7:0]: c_len[7:0], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x69	SKE_CLEN_B1	RW	[7:0]: c_len[15:8], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x6a	SKE_CLEN_B2	RW	[7:0]: c_len[23:16], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x6b	SKE_CLEN_B3	RW	[7:0]: c_len[31:24], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x6c	SKE_CLEN_B4	RW	[7:0]: c_len[39:32], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x6d	SKE_CLEN_B5	RW	[7:0]: c_len[47:40], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x6e	SKE_CLEN_B6	RW	[7:0]: c_len[55:48], ciphertext bit length This register is valid in GCM, CCM mode.	0x00

Address Offset	Name	Type	Description	Default Value
0x6f	SKE_CLEN_B7	RW	[7:0]: c_len[63:56], ciphertext bit length This register is valid in GCM, CCM mode.	0x00
0x70	SKE_IV_B0	RW	[7:0]: iv[7:0], initial value	0x00
0x71	SKE_IV_B1	RW	[7:0]: iv[15:8], initial value	0x00
0x72	SKE_IV_B2	RW	[7:0]: iv[23:16], initial value	0x00
0x73	SKE_IV_B3	RW	[7:0]: iv[31:24], initial value	0x00
0x74	SKE_IV_B4	RW	[7:0]: iv[39:32], initial value	0x00
0x75	SKE_IV_B5	RW	[7:0]: iv[47:40], initial value	0x00
0x76	SKE_IV_B6	RW	[7:0]: iv[55:48], initial value	0x00
0x77	SKE_IV_B7	RW	[7:0]: iv[63:56], initial value	0x00
0x78	SKE_IV_B8	RW	[7:0]: iv[71:64], initial value	0x00
0x79	SKE_IV_B9	RW	[7:0]: iv[79:72], initial value	0x00
0x7a	SKE_IV_B10	RW	[7:0]: iv[87:80], initial value	0x00
0x7b	SKE_IV_B11	RW	[7:0]: iv[95:88], initial value	0x00
0x7c	SKE_IV_B12	RW	[7:0]: iv[103:96], initial value	0x00
0x7d	SKE_IV_B13	RW	[7:0]: iv[111:104], initial value	0x00
0x7e	SKE_IV_B14	RW	[7:0]: iv[119:112], initial value	0x00
0x7f	SKE_IV_B15	RW	[7:0]: iv[127:120], initial value	0x00
0x80	SKE_DIN_CR_B0	RO	[7:0]: rsvd	0x00
0x81	SKE_DIN_CR_B1	RO	[7:0]: rsvd	0x00
0x82	SKE_DIN_CR_B2	RW	[0]: last, data end flag bit 0: the current data does not contain the end of the data 1: the current data contains the end of the data; this flag is valid for all GCM modes.	0x00
0x90	SKE_DIN_B0	WO	[7:0]: data_in[7:0], data input	0x00
0x91	SKE_DIN_B1	WO	[7:0]: data_in[15:8], data input	0x00
0x92	SKE_DIN_B2	WO	[7:0]: data_in[23:16], data input	0x00
0x93	SKE_DIN_B3	WO	[7:0]: data_in[31:24], data input	0x00

Address Offset	Name	Type	Description	Default Value
0x94	SKE_DIN_B4	WO	[7:0]: data_in[39:32], data input	0x00
0x95	SKE_DIN_B5	WO	[7:0]: data_in[47:40], data input	0x00
0x96	SKE_DIN_B6	WO	[7:0]: data_in[55:48], data input	0x00
0x97	SKE_DIN_B7	WO	[7:0]: data_in[63:56], data input	0x00
0x98	SKE_DIN_B8	WO	[7:0]: data_in[71:64], data input	0x00
0x99	SKE_DIN_B9	WO	[7:0]: data_in[79:72], data input	0x00
0x9a	SKE_DIN_B10	WO	[7:0]: data_in[87:80], data input	0x00
0x9b	SKE_DIN_B11	WO	[7:0]: data_in[95:88], data input	0x00
0x9c	SKE_DIN_B12	WO	[7:0]: data_in[103:96], data input	0x00
0x9d	SKE_DIN_B13	WO	[7:0]: data_in[111:104], data input	0x00
0x9e	SKE_DIN_B14	WO	[7:0]: data_in[119:112], data input	0x00
0x9f	SKE_DIN_B15	WO	[7:0]: data_in[127:120], data input	0x00
0xb0	SKE_DOUT_B0	RO	[7:0]: data_out[7:0], data output When CORE_DONE of SR2 register is 1, the user can read the value of this register as the result of the operation.	0x00
0xb1	SKE_DOUT_B1	RO	[7:0]: data_out[15:8], data output	0x00
0xb2	SKE_DOUT_B2	RO	[7:0]: data_out[23:16], data output	0x00
0xb3	SKE_DOUT_B3	RO	[7:0]: data_out[31:24], data output	0x00
0xb4	SKE_DOUT_B4	RO	[7:0]: data_out[39:32], data output	0x00
0xb5	SKE_DOUT_B5	RO	[7:0]: data_out[47:40], data output	0x00
0xb6	SKE_DOUT_B6	RO	[7:0]: data_out[55:48], data output	0x00
0xb7	SKE_DOUT_B7	RO	[7:0]: data_out[63:56], data output	0x00
0xb8	SKE_DOUT_B8	RO	[7:0]: data_out[71:64], data output	0x00
0xb9	SKE_DOUT_B9	RO	[7:0]: data_out[79:72], data output	0x00
0xba	SKE_DOUT_B10	RO	[7:0]: data_out[87:80], data output	0x00
0xbb	SKE_DOUT_B11	RO	[7:0]: data_out[95:88], data output	0x00
0xbc	SKE_DOUT_B12	RO	[7:0]: data_out[103:96], data output	0x00
0xbd	SKE_DOUT_B13	RO	[7:0]: data_out[111:104], data output	0x00

Address Offset	Name	Type	Description	Default Value
0xbe	SKE_DOUT_B14	RO	[7:0]: data_out[119:112], data output	0x00
0xbf	SKE_DOUT_B15	RO	[7:0]: data_out[127:120], data output	0x00
0xc0	SKE_SADDR_B0	RW	[7:0]: dma_saddr[7:0], DMA source address	0x00
0xc1	SKE_SADDR_B1	RW	[7:0]: dma_saddr[15:8], DMA source address	0x00
0xc2	SKE_SADDR_B2	RW	[7:0]: dma_saddr[23:16], DMA source address	0x00
0xc3	SKE_SADDR_B3	RW	[7:0]: dma_saddr[31:24], DMA source address	0x00
0xc4	SKE_DADDR_B0	RW	[7:0]: dma_daddr[7:0], DMA destination address	0x00
0xc5	SKE_DADDR_B1	RW	[7:0]: dma_daddr[15:8], DMA destination address	0x00
0xc6	SKE_DADDR_B2	RW	[7:0]: dma_daddr[23:16], DMA destination address	0x00
0xc7	SKE_DADDR_B3	RW	[7:0]: dma_daddr[31:24], DMA destination address	0x00
0xc8	SKE_RLEN_B0	RW	[7:0]: dma_rlen[7:0], DMA read byte length	0x00
0xc9	SKE_RLEN_B1	RW	[7:0]: dma_rlen[15:8], DMA read byte length	0x00
0xca	SKE_RLEN_B2	RW	[7:0]: dma_rlen[23:16], DMA read byte length	0x00
0xcb	SKE_RLEN_B3	RW	[7:0]: dma_rlen[24:31], DMA read byte length	0x00
0xcc	SKE_WLEN_B0	RW	[7:0]: dma_wlen[7:0], DMA write byte length	0x00
0xcd	SKE_WLEN_B1	RW	[7:0]: dma_wlen[15:8], DMA write byte length	0x00
0xce	SKE_WLEN_B2	RW	[7:0]: dma_wlen[23:16], DMA write byte length	0x00
0xcf	SKE_WLEN_B3	RW	[7:0]: dma_wlen[24:31], DMA write byte length	0x00
0xfc	SKE_VERSION_B0	RO	[3:0]: mir, minor version number [7:4]: mar, major version number	0x11
0xfe	SKE_VERSION_B2	RO	[7:0]: project_l, project code	0x1d
0xff	SKE_VERSION_B3	RO	[7:0]: project_h, project code	0x00

The additional SKE related registers are listed in the following table. The base address for the following registers is 0x80100800.

Table 23-2 Additional SKE Related Registers

Address Offset	Name	Type	Description	Reset Value
0x20	ROSR_CTRL20	RW	[7:0]: ske_fifo_b0, SKE DMA mode read/write data entry	0x00

Address Offset	Name	Type	Description	Reset Value
0x21	ROSR_CTRL21	RO	[7:0]: ske_fifo_b1, SKE DMA mode read/write data entry	0x00
0x22	ROSR_CTRL22	RO	[7:0]: ske_fifo_b2, SKE DMA mode read/write data entry	0x00
0x23	ROSR_CTRL23	RO	[7:0]: ske_fifo_b3, SKE DMA mode read/write data entry	0x00
0x24	ROSR_CTRL24	RW	[0]: ske_txdma_en [1]: ske_rxdma_en	0x00
0x25	ROSR_CTRL25	RW	[3:0]: ske_txf_thres [7:4]: ske_rxf_thres	0x11

24 Low-Power Hash Accelerator (HASH_LP)

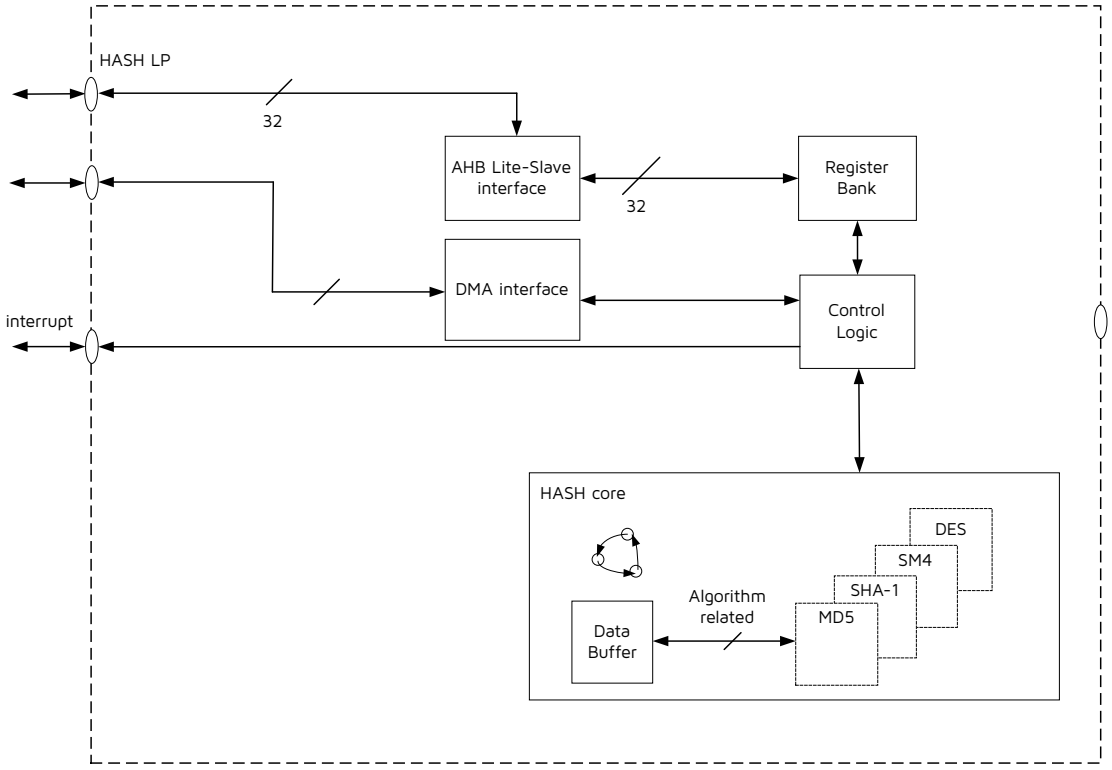
24.1 HASH_LP Overview

The HASH_LP contains a low-power hash algorithm engine, and its features include:

- Supports algorithm of MD5, SM3, SHA1, SHA224, SHA256, SHA384, SHA512
- Supports hardware of DMA mode
- Supports hardware byte padding
- Supports polling mode or interrupt mode for the host computer to query the status

The top-level block diagram is shown as below.

Figure 24-1 Block Diagram of HASH_LP Module



24.2 Function Description

24.2.1 Function Overview

After power-on reset, the HASH_LP is in the initial state. At this time, HASH_LP waits for the user to write configuration and data. Before calling HASH_LP, the user needs to write the data to be calculated into the corresponding address space, and then trigger HASH_LP to calculate. Therefore, the length of the message processed by each call to HASH_LP is less than or equal to the length of the block corresponding to the algorithm. For example, for SHA256 operations, the block length of each call to HASH_LP is less than or equal to 512 bits. For long message hash operations, the upper layer needs to make multiple calls to HASH_LP to

complete the overall hash operation. When processing to the end of the message, the HASH_LP supports hardware byte padding and software padding. The HASH_LP supports reading the message from the relevant address with DMA to perform the operation; and writing the resultant hash value to the relevant address with DMA.

24.2.2 Configuration Description

The HASH_LP_CFG.MSEL specifies which algorithm to use for the operation. The algorithm type determines the block size of the current operation, and the hash length.

The HASH_LP_CFG.IRQEN is the interrupt enable switch. When this bit is 1, the HASH_LP generates an interrupt.

The HASH_LP_CFG.DMAEN is the DMA mode enable switch. When this bit is 1, the HASH_LP uses DMA to carry data.

The HASH_LP_CFG.LAST is the end-of-message flag bit. When LAST=1, HASH_LP starts the hardware fill function.

The HASH_LP fills the message based on the algorithm type, and the current amount of data. The length information of HASH_LP_PCR_LEN is padded to the end of the message besides 1 and 0 added to the end of the message.

The HASH_LP_PCR_LEN specifies the byte length of the hash operation message. The HASH_LP adds this value to the end of the message as the last part of the hardware padding. The value consists of 4 registers (addresses 0x20 - 0x2C), write the status of the length value to the 0x20 register.

The HASH_LP_MDIN is the entry point when handling the message.

The HASH_LP_DMA_RLEN is the address space for read data length value in DMA mode.

The HASH_LP_DMA_WLEN is the address space for write data length value in DMA mode.

24.2.3 Hardware Padding

The length of the message input to HASH_LP needs to be byte-aligned. If hardware padding is required, the user needs to configure HASH_LP_CFG.LAST to 1 before calling HASH_LP, and then HASH_LP byte-paddles the last block of the message.

In addition, when performing hardware padding, there is a special case where the HASH_LP algorithm needs to be started manually twice when the input message length is an integer multiple of the algorithm's block length. Configure HASH_LP_CFG.LAST to 0 before the first start, and configure HASH_LP_CFG.LAST to 1 before the second start.

24.2.4 Status Register

During HASH_LP operation, users can query HASH_LP_SR_1, HASH_LP_SR_2 to know the operation status of the HASH_LP.

(1) HASH_LP_SR_1

BUSY is the busy indication bit. When the HASH_LP module is performing an operation, this bit is set to 1 until the operation is finished.

(2) HASH_LP_SR_2

The HASH_LP_SR_2.DONE is set to 1 after HASH_LP executes normal operation, when the HASH_LP_SR_2.DONE is set to 1, users can write 1 to this bit to clear it, and the HASH_LP also clears this status bit automatically after receiving GO instruction.

The HASH_LP_SR_2.ERR_CFG is used to indicate the error configuration, if the current MSEL is an error value, it triggers the value to be 1, and the user can clear the bit by write 1 operation.

(3) Interrupt

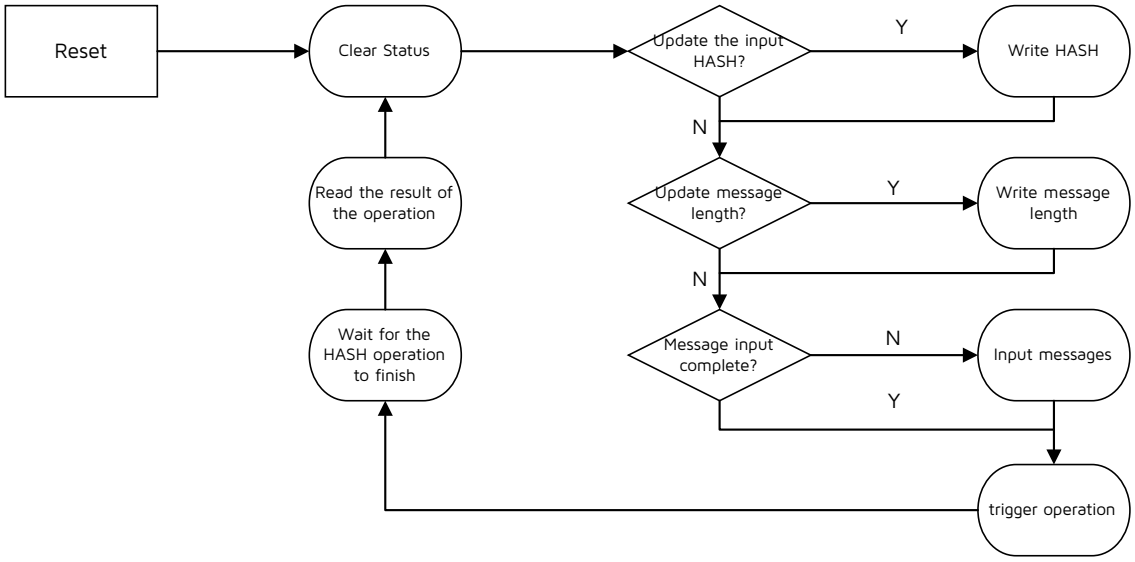
The HASH_LP provides interrupt function. The HASH_LP_CFG.IRQEN is the global interrupt enable switch. When the HASH_LP_CFG.IRQEN is set to 0, the HASH_LP does not generate interrupt. When the HASH_LP_CFG.IRQEN is 1, the HASH_LP generates interrupt. The HASH_LP generates an interrupt in the following cases:

After the HASH_LP completes the GO instruction, the HASH_LP_SR_2.DONE is set to 1 and the o_irq is pulled high to generate an interrupt, the user can clear the interrupt by writing 1 to HASH_LP_SR_2.DONE. If the user does not clear this flag bit, the HASH_LP automatically clears this flag bit when it receives the next GO instruction to ensure that this status bit corresponds to the status of the last round of operation.

24.2.5 Usage Flow

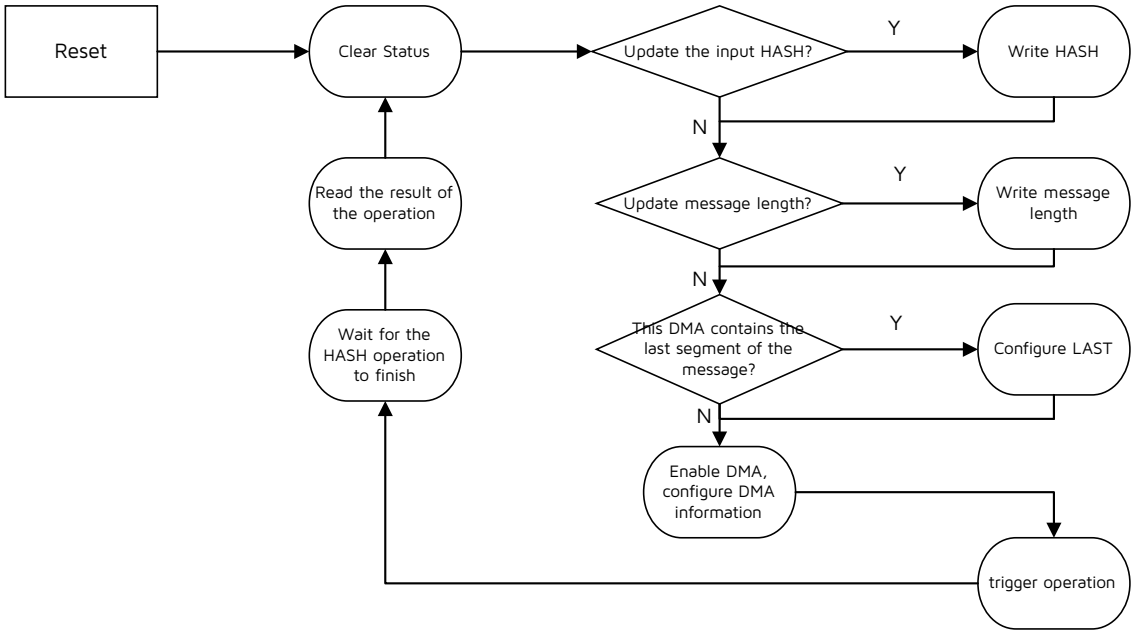
24.2.5.1 CPU Mode

Figure 24-2 Usage Flow of HASH_LP in CPU Mode



24.2.5.2 DMA Mode

Figure 24-3 Usage Flow of HASH_LP in DMA Mode



The DMA enablement is divided into two use cases: hardware padding and no hardware padding.

(1) Without message end, that is, without hardware padding, DMA only activates the reading channel and does not enable the writing channel, and the intermediate result needs to be retrieved by the CPU through bus access to HASH_LP;

(2) With the message end, that is, requires hardware padding, DMA start reading channel and writing channel, the result is stored in the SOC memory.

24.2.6 Application Example

24.2.6.1 CPU Mode SHA-256

(1) Clear the HASH_LP_SR_2 Register

0x00000001 => *(+0x000C)

(2) Write the configuration to HASH_LP_CFG register

0x01010002 => *(+0x0004)

(3) Write the message length to HASH_LP_PCR_LEN register

0x00000003 => *(+0x0020)

0x00000000 => *(+0x0024)

(4) Write data into HASH_LP_MDIN register

0x00636261 => *(+0x0100)

(5) Write the initial value of the algorithm to the HASH_LP_IN register

0x67e6096a => *(+0x0070)

0x85ae67bb => *(+0x0074)

0x72f36e3c => *(+0x0078)

0x3af54fa5 => *(+0x007C)

0x7f520e51 => *(+0x0080)

0x8c68059b => *(+0x0084)

0xabd9831f => *(+0x0088)

0x19cde05b => *(+0x008C)

(6) Start operation

0x00000001 => *(+0x0000)

(7) Read the status register to determine whether the operation is finished or not.

while ((*(+0x000C) & 0x00000001) == 0);

(8) Respond to the completion signal

0x00000001 => *(+0x000C)

(9) Read HASH_LP_OUT register

*(+0x0030) => 0XBF1678BA

*(+0x0034) => 0XEACF018F

*(+0x0038) => 0XDE404141

*(+0x003C) => 0X2322AE5D

*(+0x0040) => 0XA36103B0

*(+0x0044) => 0X9C7A1796

*(+0x0048) => 0X61FF10B4

*(+0x004C) => 0XAD1500F2

24.2.6.2 DMA Mode SHA-256

(1) Input the configuration and data

0x00000001 => *(+0x000C) //clear the status register SR2

0x01030002 => *(+0x0004) //LAST=1 DMAEN=1 MSEL=SHA-256

0x67e6096a => *(+0x0070)

0x85ae67bb => *(+0x0074)

0x72f36e3c => *(+0x0078)

0x3af54fa5 => *(+0x007C)

0x7f520e51 => *(+0x0080)

0x8c68059b => *(+0x0084)

0xabd9831f => *(+0x0088)

0x19cde05b => *(+0x008C)

0x00000003 => *(+0x0020) //Total message length is 24 bits

0x00000000 => *(+0x0024) //Total message length is 24 bits

0x00000000 => *(+0x0028) //Total message length is 24 bits

0x00000000 => *(+0x002C) //Total message length is 24 bits

(2) Configure DMA

0x00000003 => *(+0x0188) //Read data length of 03 byte

0x00000020 => *(+0x018C) //Write data length of 32 byte

(3) Write data to DMA addressable memory

*(SADDR) => 0x00636261 // message 'ABC'

where SADDR refers to the DMA source address

(4) Enable HASH_LP

0x00000001 => *(+0x0000) //enable HASH_LP

(5) Query the operation status

while ((*(+0x000C) & 0x00000001) == 0);

(6) Read the result at the destination address

*(DADDR+0x0000) => 0XBF1678BA

*(DADDR+0x0004) => 0XEACF018F

*(DADDR+0x0008) => 0XDE404141

*(DADDR+0x000C) => 0X2322AE5D

*(DADDR+0x0010) => 0XA36103B0

*(DADDR+0x0014) => 0X9C7A1796

*(DADDR+0x0018) => 0X61FF10B4

*(DADDR+0x001C) => 0XAD1500F2

where DADDR refers to the DMA destination address.

24.3 HASH_LP Register Description

The HASH_LP related registers are listed in the following table. The base address for the following HASH_LP related registers is 0x80102000.

Table 24-1 HASH_LP Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	HASH_LP_CTRL_B0	W1S	[0]:GO, start the hash operation 1: HASH_LP uses the initial hash value defined by the algorithm to perform the operation 0: No effect	0x00
0x04	HASH_LP_CFG_B0	RW	[3:0]: msel, algorithm selection 4'b0000: SM3 4'b0001: MD5 4'b0010: SHA-256 4'b0011: SHA-384 4'b0100: SHA-512 4'b0101: SHA-1 4'b0110: SHA-224 4'b0111: SHA-512/224 4'b1000: SHA-512/256 others: RSVD	0x00
0x06	HASH_LP_CFG_B2	RW	[0]: IRQEN, global interrupt enable 0: disable interrupt 1: enable interrupt [1]:DMAEN, DMA mode enable 0: disable DMA mode 1: enable DMA mode	0x01
0x07	HASH_LP_CFG_B3	RW	[0]: LAST, message end flag 0: This processed message does not contain the end 1: This processed message contains the end	0x00
0x08	HASH_LP_SR1_B0	RO	[0]: BUSY, busy indicator bit 0: HASH_LP is in busy state 1: HASH_LP is in idle state	0x00

Address Offset	Name	Type	Description	Default Value
0x0c	HASH_LP_SR2_B0	W0C	[0]: DONE, HASH_LP interrupt indicator bit 0: HASH_LP no interrupt generation 1: HASH_LP has completed the operation, interrupt generated Users can write 0 to this bit to clear the interrupt. HASH_LP also clears the interrupt when it receives a GO command. [1]: ERR_CFG When HASH_LP_CFG.MSEL is configured as an invalid value, this flag is set to 1	0x00
0x20	HASH_LP_PCR_LEN_B0	RW	[7:0]: message length This register indicates the byte length of the message to be hashed. HASH_LP uses LEN as the message length. If the length of the message to be hashed is 1024 bytes, LEN should be set to 1024.	0x00
0x21	HASH_LP_PCR_LEN_B1	RW	[7:0]: message length	0x00
0x22	HASH_LP_PCR_LEN_B2	RW	[7:0]: message length	0x00
0x23	HASH_LP_PCR_LEN_B3	RW	[7:0]: message length	0x00
0x24	HASH_LP_PCR_LEN_B4	RW	[7:0]: message length	0x00
0x25	HASH_LP_PCR_LEN_B5	RW	[7:0]: message length	0x00
0x26	HASH_LP_PCR_LEN_B6	RW	[7:0]: message length	0x00
0x27	HASH_LP_PCR_LEN_B7	RW	[7:0]: message length	0x00
0x28	HASH_LP_PCR_LEN_B8	RW	[7:0]: message length	0x00
0x29	HASH_LP_PCR_LEN_B9	RW	[7:0]: message length	0x00
0x2a	HASH_LP_PCR_LEN_B10	RW	[7:0]: message length	0x00
0x2b	HASH_LP_PCR_LEN_B11	RW	[7:0]: message length	0x00
0x2c	HASH_LP_PCR_LEN_B12	RW	[7:0]: message length	0x00
0x2d	HASH_LP_PCR_LEN_B13	RW	[7:0]: message length	0x00
0x2e	HASH_LP_PCR_LEN_B14	RW	[7:0]: message length	0x00
0x2f	HASH_LP_PCR_LEN_B15	RW	[7:0]: message length	0x00

Address Offset	Name	Type	Description	Default Value
0x30	HASH_LP_OUT_B0	RO	[7:0]: hash value output	0x00
0x31	HASH_LP_OUT_B1	RO	[7:0]: hash value output	0x00
0x32	HASH_LP_OUT_B2	RO	[7:0]: hash value output	0x00
0x33	HASH_LP_OUT_B3	RO	[7:0]: hash value output	0x00
0x34	HASH_LP_OUT_B4	RO	[7:0]: hash value output	0x00
0x35	HASH_LP_OUT_B5	RO	[7:0]: hash value output	0x00
0x36	HASH_LP_OUT_B6	RO	[7:0]: hash value output	0x00
0x37	HASH_LP_OUT_B7	RO	[7:0]: hash value output	0x00
0x38	HASH_LP_OUT_B8	RO	[7:0]: hash value output	0x00
0x39	HASH_LP_OUT_B9	RO	[7:0]: hash value output	0x00
0x3a	HASH_LP_OUT_B10	RO	[7:0]: hash value output	0x00
0x3b	HASH_LP_OUT_B11	RO	[7:0]: hash value output	0x00
0x3c	HASH_LP_OUT_B12	RO	[7:0]: hash value output	0x00
0x3d	HASH_LP_OUT_B13	RO	[7:0]: hash value output	0x00
0x3e	HASH_LP_OUT_B14	RO	[7:0]: hash value output	0x00
0x3f	HASH_LP_OUT_B15	RO	[7:0]: hash value output	0x00
0x40	HASH_LP_OUT_B16	RO	[7:0]: hash value output	0x00
0x41	HASH_LP_OUT_B17	RO	[7:0]: hash value output	0x00
0x42	HASH_LP_OUT_B18	RO	[7:0]: hash value output	0x00
0x43	HASH_LP_OUT_B19	RO	[7:0]: hash value output	0x00
0x44	HASH_LP_OUT_B20	RO	[7:0]: hash value output	0x00
0x45	HASH_LP_OUT_B21	RO	[7:0]: hash value output	0x00
0x46	HASH_LP_OUT_B22	RO	[7:0]: hash value output	0x00
0x47	HASH_LP_OUT_B23	RO	[7:0]: hash value output	0x00
0x48	HASH_LP_OUT_B24	RO	[7:0]: hash value output	0x00
0x49	HASH_LP_OUT_B25	RO	[7:0]: hash value output	0x00
0x4a	HASH_LP_OUT_B26	RO	[7:0]: hash value output	0x00

Address Offset	Name	Type	Description	Default Value
0x4b	HASH_LP_OUT_B27	RO	[7:0]: hash value output	0x00
0x4c	HASH_LP_OUT_B28	RO	[7:0]: hash value output	0x00
0x4d	HASH_LP_OUT_B29	RO	[7:0]: hash value output	0x00
0x4e	HASH_LP_OUT_B30	RO	[7:0]: hash value output	0x00
0x4f	HASH_LP_OUT_B31	RO	[7:0]: hash value output	0x00
0x50	HASH_LP_OUT_B32	RO	[7:0]: hash value output	0x00
0x51	HASH_LP_OUT_B33	RO	[7:0]: hash value output	0x00
0x52	HASH_LP_OUT_B34	RO	[7:0]: hash value output	0x00
0x53	HASH_LP_OUT_B35	RO	[7:0]: hash value output	0x00
0x54	HASH_LP_OUT_B36	RO	[7:0]: hash value output	0x00
0x55	HASH_LP_OUT_B37	RO	[7:0]: hash value output	0x00
0x56	HASH_LP_OUT_B38	RO	[7:0]: hash value output	0x00
0x57	HASH_LP_OUT_B39	RO	[7:0]: hash value output	0x00
0x58	HASH_LP_OUT_B40	RO	[7:0]: hash value output	0x00
0x59	HASH_LP_OUT_B41	RO	[7:0]: hash value output	0x00
0x5a	HASH_LP_OUT_B42	RO	[7:0]: hash value output	0x00
0x5b	HASH_LP_OUT_B43	RO	[7:0]: hash value output	0x00
0x5c	HASH_LP_OUT_B44	RO	[7:0]: hash value output	0x00
0x5d	HASH_LP_OUT_B45	RO	[7:0]: hash value output	0x00
0x5e	HASH_LP_OUT_B46	RO	[7:0]: hash value output	0x00
0x5f	HASH_LP_OUT_B47	RO	[7:0]: hash value output	0x00
0x60	HASH_LP_OUT_B48	RO	[7:0]: hash value output	0x00
0x61	HASH_LP_OUT_B49	RO	[7:0]: hash value output	0x00
0x62	HASH_LP_OUT_B50	RO	[7:0]: hash value output	0x00
0x63	HASH_LP_OUT_B51	RO	[7:0]: hash value output	0x00
0x64	HASH_LP_OUT_B52	RO	[7:0]: hash value output	0x00
0x65	HASH_LP_OUT_B53	RO	[7:0]: hash value output	0x00

Address Offset	Name	Type	Description	Default Value
0x66	HASH_LP_OUT_B54	RO	[7:0]: hash value output	0x00
0x67	HASH_LP_OUT_B55	RO	[7:0]: hash value output	0x00
0x68	HASH_LP_OUT_B56	RO	[7:0]: hash value output	0x00
0x69	HASH_LP_OUT_B57	RO	[7:0]: hash value output	0x00
0x6a	HASH_LP_OUT_B58	RO	[7:0]: hash value output	0x00
0x6b	HASH_LP_OUT_B59	RO	[7:0]: hash value output	0x00
0x6c	HASH_LP_OUT_B60	RO	[7:0]: hash value output	0x00
0x6d	HASH_LP_OUT_B61	RO	[7:0]: hash value output	0x00
0x6e	HASH_LP_OUT_B62	RO	[7:0]: hash value output	0x00
0x6f	HASH_LP_OUT_B63	RO	[7:0]: hash value output	0x00
0x70	HASH_LP_IN_B0	WO	[7:0]: hash value input	0x00
0x71	HASH_LP_IN_B1	WO	[7:0]: hash value input	0x00
0x72	HASH_LP_IN_B2	WO	[7:0]: hash value input	0x00
0x73	HASH_LP_IN_B3	WO	[7:0]: hash value input	0x00
0x74	HASH_LP_IN_B4	WO	[7:0]: hash value input	0x00
0x75	HASH_LP_IN_B5	WO	[7:0]: hash value input	0x00
0x76	HASH_LP_IN_B6	WO	[7:0]: hash value input	0x00
0x77	HASH_LP_IN_B7	WO	[7:0]: hash value input	0x00
0x78	HASH_LP_IN_B8	WO	[7:0]: hash value input	0x00
0x79	HASH_LP_IN_B9	WO	[7:0]: hash value input	0x00
0x7a	HASH_LP_IN_B10	WO	[7:0]: hash value input	0x00
0x7b	HASH_LP_IN_B11	WO	[7:0]: hash value input	0x00
0x7c	HASH_LP_IN_B12	WO	[7:0]: hash value input	0x00
0x7d	HASH_LP_IN_B13	WO	[7:0]: hash value input	0x00
0x7e	HASH_LP_IN_B14	WO	[7:0]: hash value input	0x00
0x7f	HASH_LP_IN_B15	WO	[7:0]: hash value input	0x00
0x80	HASH_LP_IN_B16	WO	[7:0]: hash value input	0x00

Address Offset	Name	Type	Description	Default Value
0x81	HASH_LP_IN_B17	WO	[7:0]: hash value input	0x00
0x82	HASH_LP_IN_B18	WO	[7:0]: hash value input	0x00
0x83	HASH_LP_IN_B19	WO	[7:0]: hash value input	0x00
0x84	HASH_LP_IN_B20	WO	[7:0]: hash value input	0x00
0x85	HASH_LP_IN_B21	WO	[7:0]: hash value input	0x00
0x86	HASH_LP_IN_B22	WO	[7:0]: hash value input	0x00
0x87	HASH_LP_IN_B23	WO	[7:0]: hash value input	0x00
0x88	HASH_LP_IN_B24	WO	[7:0]: hash value input	0x00
0x89	HASH_LP_IN_B25	WO	[7:0]: hash value input	0x00
0x8a	HASH_LP_IN_B26	WO	[7:0]: hash value input	0x00
0x8b	HASH_LP_IN_B27	WO	[7:0]: hash value input	0x00
0x8c	HASH_LP_IN_B28	WO	[7:0]: hash value input	0x00
0x8d	HASH_LP_IN_B29	WO	[7:0]: hash value input	0x00
0x8e	HASH_LP_IN_B30	WO	[7:0]: hash value input	0x00
0x8f	HASH_LP_IN_B31	WO	[7:0]: hash value input	0x00
0x90	HASH_LP_IN_B32	WO	[7:0]: hash value input	0x00
0x91	HASH_LP_IN_B33	WO	[7:0]: hash value input	0x00
0x92	HASH_LP_IN_B34	WO	[7:0]: hash value input	0x00
0x93	HASH_LP_IN_B35	WO	[7:0]: hash value input	0x00
0x94	HASH_LP_IN_B36	WO	[7:0]: hash value input	0x00
0x95	HASH_LP_IN_B37	WO	[7:0]: hash value input	0x00
0x96	HASH_LP_IN_B38	WO	[7:0]: hash value input	0x00
0x97	HASH_LP_IN_B39	WO	[7:0]: hash value input	0x00
0x98	HASH_LP_IN_B40	WO	[7:0]: hash value input	0x00
0x99	HASH_LP_IN_B41	WO	[7:0]: hash value input	0x00
0x9a	HASH_LP_IN_B42	WO	[7:0]: hash value input	0x00
0x9b	HASH_LP_IN_B43	WO	[7:0]: hash value input	0x00

Address Offset	Name	Type	Description	Default Value
0x9c	HASH_LP_IN_B44	WO	[7:0]: hash value input	0x00
0x9d	HASH_LP_IN_B45	WO	[7:0]: hash value input	0x00
0x9e	HASH_LP_IN_B46	WO	[7:0]: hash value input	0x00
0x9f	HASH_LP_IN_B47	WO	[7:0]: hash value input	0x00
0xa0	HASH_LP_IN_B48	WO	[7:0]: hash value input	0x00
0xa1	HASH_LP_IN_B49	WO	[7:0]: hash value input	0x00
0xa2	HASH_LP_IN_B50	WO	[7:0]: hash value input	0x00
0xa3	HASH_LP_IN_B51	WO	[7:0]: hash value input	0x00
0xa4	HASH_LP_IN_B52	WO	[7:0]: hash value input	0x00
0xa5	HASH_LP_IN_B53	WO	[7:0]: hash value input	0x00
0xa6	HASH_LP_IN_B54	WO	[7:0]: hash value input	0x00
0xa7	HASH_LP_IN_B55	WO	[7:0]: hash value input	0x00
0xa8	HASH_LP_IN_B56	WO	[7:0]: hash value input	0x00
0xa9	HASH_LP_IN_B57	WO	[7:0]: hash value input	0x00
0xaa	HASH_LP_IN_B58	WO	[7:0]: hash value input	0x00
0xab	HASH_LP_IN_B59	WO	[7:0]: hash value input	0x00
0xac	HASH_LP_IN_B60	WO	[7:0]: hash value input	0x00
0xad	HASH_LP_IN_B61	WO	[7:0]: hash value input	0x00
0xae	HASH_LP_IN_B62	WO	[7:0]: hash value input	0x00
0xaf	HASH_LP_IN_B63	WO	[7:0]: hash value input	0x00
0xb0	HASH_LP_VERSION_B0	RO	[3:0]: mir, minor version number [7:4]: mar, major version number	0x23
0xb1	HASH_LP_VERSION_B1	RO	[7:0]: rsvd	0x00
0xb2	HASH_LP_VERSION_B2	RO	[7:0]: project_l, project code	0x1e
0xb3	HASH_LP_VERSION_B3	RO	[7:0]: project_h, project code	0x00
0x100	HASH_LP_MDIN_B0	RW	[7:0]:message input	0x00
0x101	HASH_LP_MDIN_B1	RW	[7:0]:message input	0x00
0x102	HASH_LP_MDIN_B2	RW	[7:0]:message input	0x00

Address Offset	Name	Type	Description	Default Value
0x103	HASH_LP_MDIN_B3	RW	[7:0]:message input	0x00
0x104	HASH_LP_MDIN_B4	RW	[7:0]:message input	0x00
0x105	HASH_LP_MDIN_B5	RW	[7:0]:message input	0x00
0x106	HASH_LP_MDIN_B6	RW	[7:0]:message input	0x00
0x107	HASH_LP_MDIN_B7	RW	[7:0]:message input	0x00
0x108	HASH_LP_MDIN_B8	RW	[7:0]:message input	0x00
0x109	HASH_LP_MDIN_B9	RW	[7:0]:message input	0x00
0x10a	HASH_LP_MDIN_B10	RW	[7:0]:message input	0x00
0x10b	HASH_LP_MDIN_B11	RW	[7:0]:message input	0x00
0x10c	HASH_LP_MDIN_B12	RW	[7:0]:message input	0x00
0x10d	HASH_LP_MDIN_B13	RW	[7:0]:message input	0x00
0x10e	HASH_LP_MDIN_B14	RW	[7:0]:message input	0x00
0x10f	HASH_LP_MDIN_B15	RW	[7:0]:message input	0x00
0x110	HASH_LP_MDIN_B16	RW	[7:0]:message input	0x00
0x111	HASH_LP_MDIN_B17	RW	[7:0]:message input	0x00
0x112	HASH_LP_MDIN_B18	RW	[7:0]:message input	0x00
0x113	HASH_LP_MDIN_B19	RW	[7:0]:message input	0x00
0x114	HASH_LP_MDIN_B20	RW	[7:0]:message input	0x00
0x115	HASH_LP_MDIN_B21	RW	[7:0]:message input	0x00
0x116	HASH_LP_MDIN_B22	RW	[7:0]:message input	0x00
0x117	HASH_LP_MDIN_B23	RW	[7:0]:message input	0x00
0x118	HASH_LP_MDIN_B24	RW	[7:0]:message input	0x00
0x119	HASH_LP_MDIN_B25	RW	[7:0]:message input	0x00
0x11a	HASH_LP_MDIN_B26	RW	[7:0]:message input	0x00
0x11b	HASH_LP_MDIN_B27	RW	[7:0]:message input	0x00
0x11c	HASH_LP_MDIN_B28	RW	[7:0]:message input	0x00
0x11d	HASH_LP_MDIN_B29	RW	[7:0]:message input	0x00

Address Offset	Name	Type	Description	Default Value
0x11e	HASH_LP_MDIN_B30	RW	[7:0]:message input	0x00
0x11f	HASH_LP_MDIN_B31	RW	[7:0]:message input	0x00
0x120	HASH_LP_MDIN_B32	RW	[7:0]:message input	0x00
0x121	HASH_LP_MDIN_B33	RW	[7:0]:message input	0x00
0x122	HASH_LP_MDIN_B34	RW	[7:0]:message input	0x00
0x123	HASH_LP_MDIN_B35	RW	[7:0]:message input	0x00
0x124	HASH_LP_MDIN_B36	RW	[7:0]:message input	0x00
0x125	HASH_LP_MDIN_B37	RW	[7:0]:message input	0x00
0x126	HASH_LP_MDIN_B38	RW	[7:0]:message input	0x00
0x127	HASH_LP_MDIN_B39	RW	[7:0]:message input	0x00
0x128	HASH_LP_MDIN_B40	RW	[7:0]:message input	0x00
0x129	HASH_LP_MDIN_B41	RW	[7:0]:message input	0x00
0x12a	HASH_LP_MDIN_B42	RW	[7:0]:message input	0x00
0x12b	HASH_LP_MDIN_B43	RW	[7:0]:message input	0x00
0x12c	HASH_LP_MDIN_B44	RW	[7:0]:message input	0x00
0x12d	HASH_LP_MDIN_B45	RW	[7:0]:message input	0x00
0x12e	HASH_LP_MDIN_B46	RW	[7:0]:message input	0x00
0x12f	HASH_LP_MDIN_B47	RW	[7:0]:message input	0x00
0x130	HASH_LP_MDIN_B48	RW	[7:0]:message input	0x00
0x131	HASH_LP_MDIN_B49	RW	[7:0]:message input	0x00
0x132	HASH_LP_MDIN_B50	RW	[7:0]:message input	0x00
0x133	HASH_LP_MDIN_B51	RW	[7:0]:message input	0x00
0x134	HASH_LP_MDIN_B52	RW	[7:0]:message input	0x00
0x135	HASH_LP_MDIN_B53	RW	[7:0]:message input	0x00
0x136	HASH_LP_MDIN_B54	RW	[7:0]:message input	0x00
0x137	HASH_LP_MDIN_B55	RW	[7:0]:message input	0x00
0x138	HASH_LP_MDIN_B56	RW	[7:0]:message input	0x00

Address Offset	Name	Type	Description	Default Value
0x139	HASH_LP_MDIN_B57	RW	[7:0]:message input	0x00
0x13a	HASH_LP_MDIN_B58	RW	[7:0]:message input	0x00
0x13b	HASH_LP_MDIN_B59	RW	[7:0]:message input	0x00
0x13c	HASH_LP_MDIN_B60	RW	[7:0]:message input	0x00
0x13d	HASH_LP_MDIN_B61	RW	[7:0]:message input	0x00
0x13e	HASH_LP_MDIN_B62	RW	[7:0]:message input	0x00
0x13f	HASH_LP_MDIN_B63	RW	[7:0]:message input	0x00
0x140	HASH_LP_MDIN_B64	RW	[7:0]:message input	0x00
0x141	HASH_LP_MDIN_B65	RW	[7:0]:message input	0x00
0x142	HASH_LP_MDIN_B66	RW	[7:0]:message input	0x00
0x143	HASH_LP_MDIN_B67	RW	[7:0]:message input	0x00
0x144	HASH_LP_MDIN_B68	RW	[7:0]:message input	0x00
0x145	HASH_LP_MDIN_B69	RW	[7:0]:message input	0x00
0x146	HASH_LP_MDIN_B70	RW	[7:0]:message input	0x00
0x147	HASH_LP_MDIN_B71	RW	[7:0]:message input	0x00
0x148	HASH_LP_MDIN_B72	RW	[7:0]:message input	0x00
0x149	HASH_LP_MDIN_B73	RW	[7:0]:message input	0x00
0x14a	HASH_LP_MDIN_B74	RW	[7:0]:message input	0x00
0x14b	HASH_LP_MDIN_B75	RW	[7:0]:message input	0x00
0x14c	HASH_LP_MDIN_B76	RW	[7:0]:message input	0x00
0x14d	HASH_LP_MDIN_B77	RW	[7:0]:message input	0x00
0x14e	HASH_LP_MDIN_B78	RW	[7:0]:message input	0x00
0x14f	HASH_LP_MDIN_B79	RW	[7:0]:message input	0x00
0x150	HASH_LP_MDIN_B80	RW	[7:0]:message input	0x00
0x151	HASH_LP_MDIN_B81	RW	[7:0]:message input	0x00
0x152	HASH_LP_MDIN_B82	RW	[7:0]:message input	0x00
0x153	HASH_LP_MDIN_B83	RW	[7:0]:message input	0x00

Address Offset	Name	Type	Description	Default Value
0x154	HASH_LP_MDIN_B84	RW	[7:0]:message input	0x00
0x155	HASH_LP_MDIN_B85	RW	[7:0]:message input	0x00
0x156	HASH_LP_MDIN_B86	RW	[7:0]:message input	0x00
0x157	HASH_LP_MDIN_B87	RW	[7:0]:message input	0x00
0x158	HASH_LP_MDIN_B88	RW	[7:0]:message input	0x00
0x159	HASH_LP_MDIN_B89	RW	[7:0]:message input	0x00
0x15a	HASH_LP_MDIN_B90	RW	[7:0]:message input	0x00
0x15b	HASH_LP_MDIN_B91	RW	[7:0]:message input	0x00
0x15c	HASH_LP_MDIN_B92	RW	[7:0]:message input	0x00
0x15d	HASH_LP_MDIN_B93	RW	[7:0]:message input	0x00
0x15e	HASH_LP_MDIN_B94	RW	[7:0]:message input	0x00
0x15f	HASH_LP_MDIN_B95	RW	[7:0]:message input	0x00
0x160	HASH_LP_MDIN_B96	RW	[7:0]:message input	0x00
0x161	HASH_LP_MDIN_B97	RW	[7:0]:message input	0x00
0x162	HASH_LP_MDIN_B98	RW	[7:0]:message input	0x00
0x163	HASH_LP_MDIN_B99	RW	[7:0]:message input	0x00
0x164	HASH_LP_MDIN_B100	RW	[7:0]:message input	0x00
0x165	HASH_LP_MDIN_B101	RW	[7:0]:message input	0x00
0x166	HASH_LP_MDIN_B102	RW	[7:0]:message input	0x00
0x167	HASH_LP_MDIN_B103	RW	[7:0]:message input	0x00
0x168	HASH_LP_MDIN_B104	RW	[7:0]:message input	0x00
0x169	HASH_LP_MDIN_B105	RW	[7:0]:message input	0x00
0x16a	HASH_LP_MDIN_B106	RW	[7:0]:message input	0x00
0x16b	HASH_LP_MDIN_B107	RW	[7:0]:message input	0x00
0x16c	HASH_LP_MDIN_B108	RW	[7:0]:message input	0x00
0x16d	HASH_LP_MDIN_B109	RW	[7:0]:message input	0x00
0x16e	HASH_LP_MDIN_B110	RW	[7:0]:message input	0x00

Address Offset	Name	Type	Description	Default Value
0x16f	HASH_LP_MDIN_B111	RW	[7:0]:message input	0x00
0x170	HASH_LP_MDIN_B112	RW	[7:0]:message input	0x00
0x171	HASH_LP_MDIN_B113	RW	[7:0]:message input	0x00
0x172	HASH_LP_MDIN_B114	RW	[7:0]:message input	0x00
0x173	HASH_LP_MDIN_B115	RW	[7:0]:message input	0x00
0x174	HASH_LP_MDIN_B116	RW	[7:0]:message input	0x00
0x175	HASH_LP_MDIN_B117	RW	[7:0]:message input	0x00
0x176	HASH_LP_MDIN_B118	RW	[7:0]:message input	0x00
0x177	HASH_LP_MDIN_B119	RW	[7:0]:message input	0x00
0x178	HASH_LP_MDIN_B120	RW	[7:0]:message input	0x00
0x179	HASH_LP_MDIN_B121	RW	[7:0]:message input	0x00
0x17a	HASH_LP_MDIN_B122	RW	[7:0]:message input	0x00
0x17b	HASH_LP_MDIN_B123	RW	[7:0]:message input	0x00
0x17c	HASH_LP_MDIN_B124	RW	[7:0]:message input	0x00
0x17d	HASH_LP_MDIN_B125	RW	[7:0]:message input	0x00
0x17e	HASH_LP_MDIN_B126	RW	[7:0]:message input	0x00
0x17f	HASH_LP_MDIN_B127	RW	[7:0]:message input	0x00
0x188	HASH_LP_DMA_RLEN_B0	RW	[7:0]:DMA read message length	0x00
0x189	HASH_LP_DMA_RLEN_B1	RW	[7:0]:DMA read message length	0x00
0x18a	HASH_LP_DMA_RLEN_B2	RW	[7:0]:DMA read message length	0x00
0x18b	HASH_LP_DMA_RLEN_B3	RW	[7:0]:DMA read message length	0x00
0x18c	HASH_LP_DMA_WLEN_B0	RW	[7:0]:DMA write length	0x00
0x18d	HASH_LP_DMA_WLEN_B1	RW	[7:0]:DMA write length	0x00
0x18e	HASH_LP_DMA_WLEN_B2	RW	[7:0]:DMA write length	0x00
0x18f	HASH_LP_DMA_WLEN_B3	RW	[7:0]:DMA write length	0x00

The additional HASH_LP related registers are listed in the following table. The base address for the following registers is 0x80100800.

Table 24-2 Additional HASH_LP Related Registers

Address Offset	Name	Type	Description	Reset Value
0x30	ROSR_CTRL30	RW	[7:0]: hash_fifo_b0, HASH DMA mode read/write data entry	0x00
0x31	ROSR_CTRL31	RO	[7:0]: hash_fifo_b1, HASH_LP DMA mode read/write data entry	0x00
0x32	ROSR_CTRL32	RO	[7:0]: hash_fifo_b2, HASH_LP DMA mode read/write data entry	0x00
0x33	ROSR_CTRL33	RO	[7:0]: hash_fifo_b3, HASH_LP DMA mode read/write data entry	0x00
0x34	ROSR_CTRL34	RW	[0]: hash_txdma_en [1]: hash_rxdma_en	0x00
0x35	ROSR_CTRL35	RW	[3:0]: hash_txf_thres [7:4]: hash_rxf_thres	0x11

25 Chacha20-poly1305 Accelerator (CHACHA20)

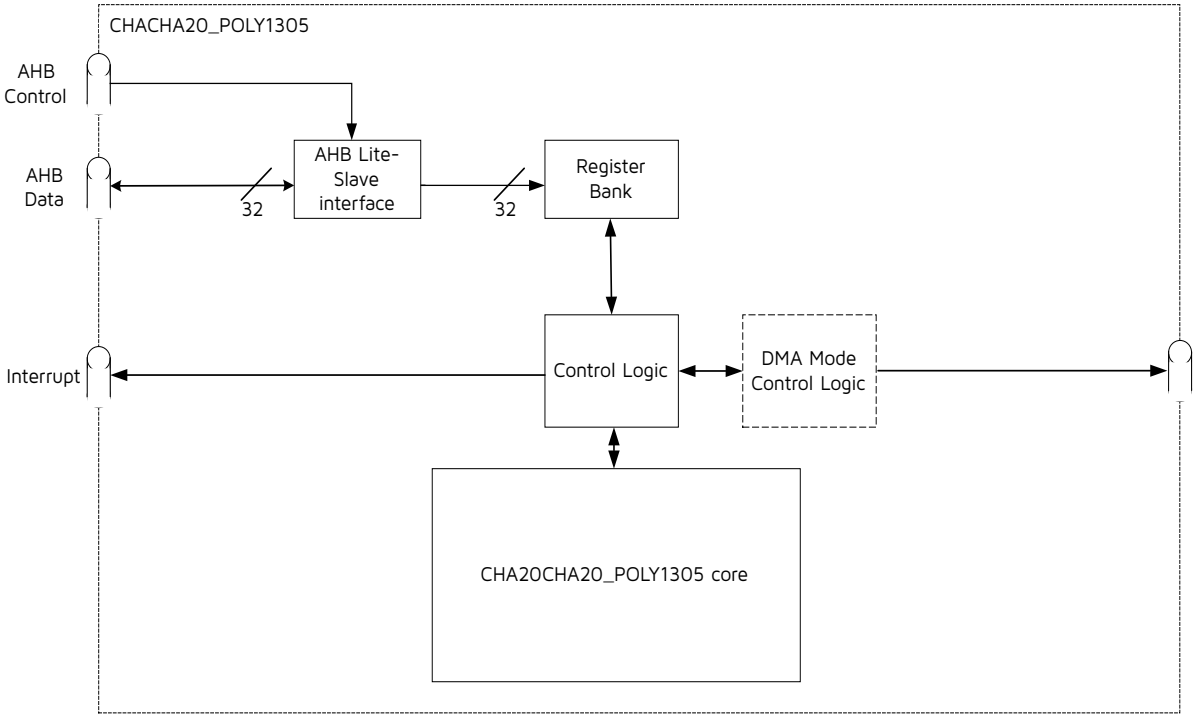
25.1 CHACHA20 Overview

The CHACHA20 contains a Chacha20-poly1305 algorithm engine, and its features include:

- Compliants to 'ChaCha20 and Poly1305 for IETF Protocols'
- Supports CPU and DMA mode
- Supports encryption and decryption
- Supports data interleave

The top-level block diagram is shown as below.

Figure 25-1 Block Diagram of CHACHA20_POLY1305

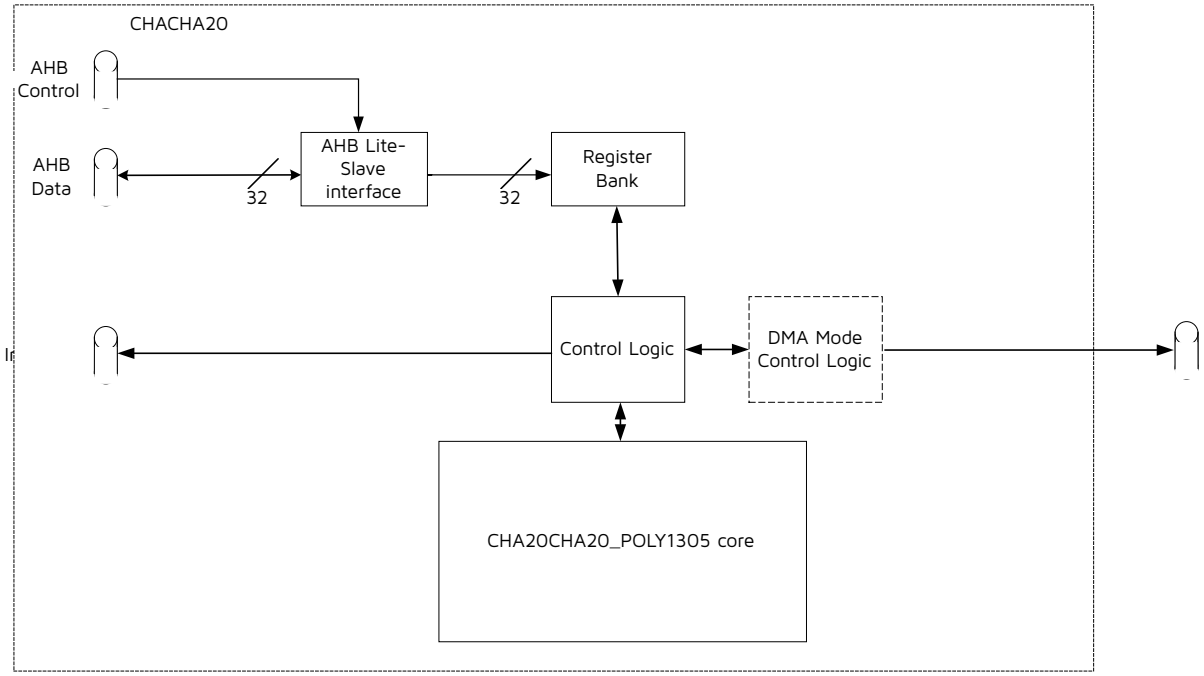


25.2 Function Description

25.2.1 CHACHA20 Introduction

The top-level block diagram of CHACHA20 is shown as below.

Figure 25-2 Block Diagram of CHACHA20



CHACHA20 is a module containing chacha20_poly1305 core and its control logic. Users can easily configure and transfer message to chacha20_poly1305 core through AHB bus and DMA means.

The configuration of core mode, length, nonce, keys, counters and other information is mainly completed through AHB bus interface. Message input and output can be realized by AHB bus interface and DMA.

Before calculation, users need to configure the options before transferring message. In CPU mode, to indicate the end of the message, Users need to configure LAST register to 1 before sending last message, and after sending last message, LAST register need to be configured to 0. In DMA mode, Users don't need to configure this register, The DMA control logic completes it.

User can determine whether the calculation is done by querying STATUS register or the interrupt signal, and check whether the calculation is wrong by querying ERR_CODE register.

This implementation compliants to 'ChaCha20 and Poly1305 for IETF Protocols'.

In DMA mode, DMA Mode Control Logic module reads and writes message according to the configured sec_stage information. For example, when stage is 0, that is, in One stage, the DMA control module inputs AAD and PAYLOAD, meanwhile output ciphertext in encryption or plaintext in decryption. Another output TAG can only be read through AHB interface, not through DMA. For a detailed sec_stage usage, please refer to SEC_STAGE register description.

25.2.2 Configuration

The configuration includes dma_en, dec, sec_stage, key, iv, const, length, counter and tag. All config can be write into the core when it is not busy, which can read from STATUS register. If STATUS register is 1, which is mean core is busy, it does not receive config value.When core is busy, config information can't be write into the core.

25.2.2.1 Section Stage, Dec and DMA_EN

The section_stage, dec and DMA_en must input before other config information. Section stage is a flag for data interleave. Section_stage have 4 values, including 0, 1, 2 and 3. 0 means current computation processes all aad and payload of a transation. If data of a transation is divided into several sections, section_stage must be config according to section order

For example, data is divided into 3 sections. First section includes all aad and partial payload, section_stage must be 1, which means init stage of a transation. Second section includes partial payload and section_stage must be 2, which means middle stage of a transation. Third section includes the residual payload and section_stage must be 3.The difference between middle stage and last stage is wheather the data includes the end of the transation.

Dec is a flag of encryption or decryption. 0 means encryption and 1 means decryption

When dma_en is 0, which means CPU mode, the assert of CTRL register starts the core operation directly, and the core is in the stage for waiting data. When dma_en is 1, which means DMA mode, additional DMA options need to be config before start. The assert of CTRL register starts DMA control logic and trigger data transfer between core and DMA.

25.2.2.2 Key

Chacha20 need 32 bytes key (8 words). 8 words key are stored in a chacha20 state. It can be configured through the AHB bus. Following example shows how to write a word into a chacha20 state.

If a key stream is 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d,0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e,0x1f. Value 0x00 is in address 0. The key word position and its relationship with state index shows in the following figure.

Figure 25-3 CHACHA20 Key Position and State Index

state index	0	1	2	3
value	constant	constant	constant	constant
state index	4	5	6	7
key_value	0x03020100	0x07060504	0x0b0a0908	0x0f0e0d0c
state index	8	9	10	11
key_value	0x13121110	0x17161514	0x1b1a1918	0x1f1e1d1c
state index	12	13	14	15
value	counter	nonce	nonce	nonce

The 1st least significant word key writes to state[4], mapping to register address 0x30.

The 2nd word key writes to state[5], mapping to register address 0x34.

The 3rd word key writes to state[6], mapping to register address 0x38.

The 4th word key writes to state[7], mapping to register address 0x3C.

The 5th word key writes to state[8], mapping to register address 0x40.

The 6th word key writes to state[9], mapping to register address 0x44.

The 7th word key writes to state[10], mapping to register address 0x48.

The most significant word key writes to state[11], mapping to register address 0x4C.

25.2.2.3 IV and Constant

Initial nonce of chacha20 state contains iv and constant from outside. IV is 64 bits and constant is 32 bits. If iv stream are 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, and value 0x40 is in low address. Constant stream are 0x50, 0x52, 0x53, 0x54, and value 0x50 is in low address. The iv and constant word position and there relationship with state index shown in the following figure.

Figure 25-4 CHACHA20 Nonce Position and State Index

state index	0	1	2	3
value	constant	constant	constant	constant
state index	4	5	6	7
key_value	key	key	key	key
state index	8	9	10	11
key_value	key	key	key	key
state index	12	13	14	15
value	cnt	0x53525150	0x43424140	0x47464544

The least significant word iv writes to state[14], mapping to register address 0x50.

The most significant word iv writes to state[15], mapping to register address 0x54.

The one_word const writes to state[13], mapping to register address 0x58.

25.2.2.4 Counter

There is a counter in chacha20 state. If current section_stage is 0 or 1, CHACHA20 sets counter to 1 automatically. If current section_stage is 2 or 3, it is required to set counter from outside.

If counter stream is 0x02, 0x00, 0x00, 0x00 and value 0x02 is in low address, the counter in state shows in following figure.

Figure 25-5 CHACHA20 Counter Position and State Index

state index	0	1	2	3
value	constant	constant	constant	constant
state index	4	5	6	7
key_value	0x03020100	0x07060504	0x0b0a0908	0x0f0e0d0c
state index	8	9	10	11
key_value	0x13121110	0x17161514	0x1b1a1918	0x1f1e1d1c
state index	12	13	14	15
value	0x00000002	const	iv	iv

The one_word cnt writes to state[12], mapping to register address 0x74.

25.2.2.5 Length Information

There is a 134 bits register of length information in CHACHA20, which includes lengths of aad and payload. The 6 most significant bits indicate the current stage byte length of last word. When section stage is 0 and 3, the last 128 bit represents the total length of message. And when section stage is 1 and 2, the last 128 bit represents the current stage length of message. Following figure shows relationship between register section with length information.

The byte length of aad is 0x00000000_00100002.

The byte length of last aad word in current stage is 2 bytes.

The byte length of payload is 0x00000000_00200003.

The byte length of last payload word in current stage is 3 bytes.

Figure 25-6 CHACHA20 Length Information and Register Address

index		4		3	2	1 0
length section	[133:131]	[130:128]	[127:96]	[95:64]	[63:32]	[31:0]
	aad last word	payload last word	payload len MSB	payload len LSB	aad len MSB	aad len LSB
length register value	0x02	0x03	0x00000000	0x00200003	0x00000000	0x00100002
register address		0x70	0x6c	0x68	0x64	0x60

The aad len least significant word writes to index_0, mapping to register address 0x60.

The aad len most significant word writes to index_1, mapping to register address 0x64.

The payload len least significant word writes to index_2, mapping to register address 0x68.

The payload len most significant word writes to index_3, mapping to register address 0x6C.

The last word len of aad and payload in current stage writes to index_4, mapping to register address 0x70.

25.2.2.6 Tag

There is a tag in poly1305 state. If current section_stage is 0 or 1, the core sets tag to 0 automatically. If current section_stage is 2 or 3, it is required to set tag from outside.

If tag stream is 0x1f, 0x1e, 0x1d, 0x1c, 0x1b, 0x1a, 0x19, 0x18, 0x17, 0x16, 0x15, 0x14, 0x13, 0x12, 0x11,0x10, 0x02 and value 0x1f is in low address, the tag in register shows in following figure.

Figure 25-7 CHACHA20 Tag Position and Register Address

index	4		3	2	1 0
Tag section	[129:128]	[127:96]	[95:64]	[63:32]	[31:0]
Tag register value	0x02	0x10111213	0x14151617	0x18191a1b	0x1c1d1e1f
register address	0x90	0x8c	0x88	0x84	0x80

The 1st least significant word tag writes to index_0, mapping to register address 0x80.

The 2nd word tag writes to index_1, mapping to register address 0x84.

The 3rd word tag writes to index_2, mapping to register address 0x88.

The 4th word tag writes to index_3, mapping to register address 0x8C.

The most significant 2_bit tag writes to index_4, mapping to register address 0x90.

25.2.3 Input Data

Input data includes aad or payload.In CPU mode, before sending aad or payload, users need to query DIN_RDY register first. When the rdata is 1, users need to send 4 words to the core. Then for aad, users need to query that register again for the next 4 words. But for payload, after sending payload, users need to poll the RISR register. When the rdata is 1, it means that Dout has been generated, and then read the data. After reading, users need to query that register again for the next 4 words. In CPU mode, before the last word of aad or payload, CPU assert the LAST register to 1. After sending the last word, the LAST register needs to be deasserted immediately. In DMA mode, Users need to configure DMA length and address into the relevant register firstly.

If aad stream is 0x01, 0x02, 0x03, 0x04, 0x05, that 0x01 is LSB and 0x05 is MSB, and payload stream is 0x21, 0x22, 0x23, 0x24, 0x25, that 0x21 is LSB and 0x25 is MSB, DIN register, whose address is 0x94, is written to 0x04030201, 0x00000005, 0x24232221 and 0x00000025 in turn.

25.2.4 Output Result

Output result includes dout and tag_out. In each computation, there are a tag_out and serveral dout if payload length > 0.

Each 128-bit payload is operated to produce 128-bit dout, followed by the next 128-bit payload. In DMA mode, dout is automatically transmitted to the destination address. In CPU mode, once the dout is generated, an interrupt signal is generated at the same time. Users can poll RISR register to determine whether the 128-bits calculation completed. At this time, dout can be read by following AHB bus,

Dout[31:0] is mapping to register address 0x100.

Dout[63:32] is mapping to register address 0x104.

Dout[95:64] is mapping to register address 0x108.

Dout[127:96] is mapping to register address 0x10C.

After input the end payload, in addition to producting dout, the core also outputs 130-bit tag_out. Tag_out can be read by following AHB bus,

tag_out[31:0] is mapping to register address 0x110.

tag_out[63:32] is mapping to register address 0x114.

tag_out[95:64] is mapping to register address 0x118.

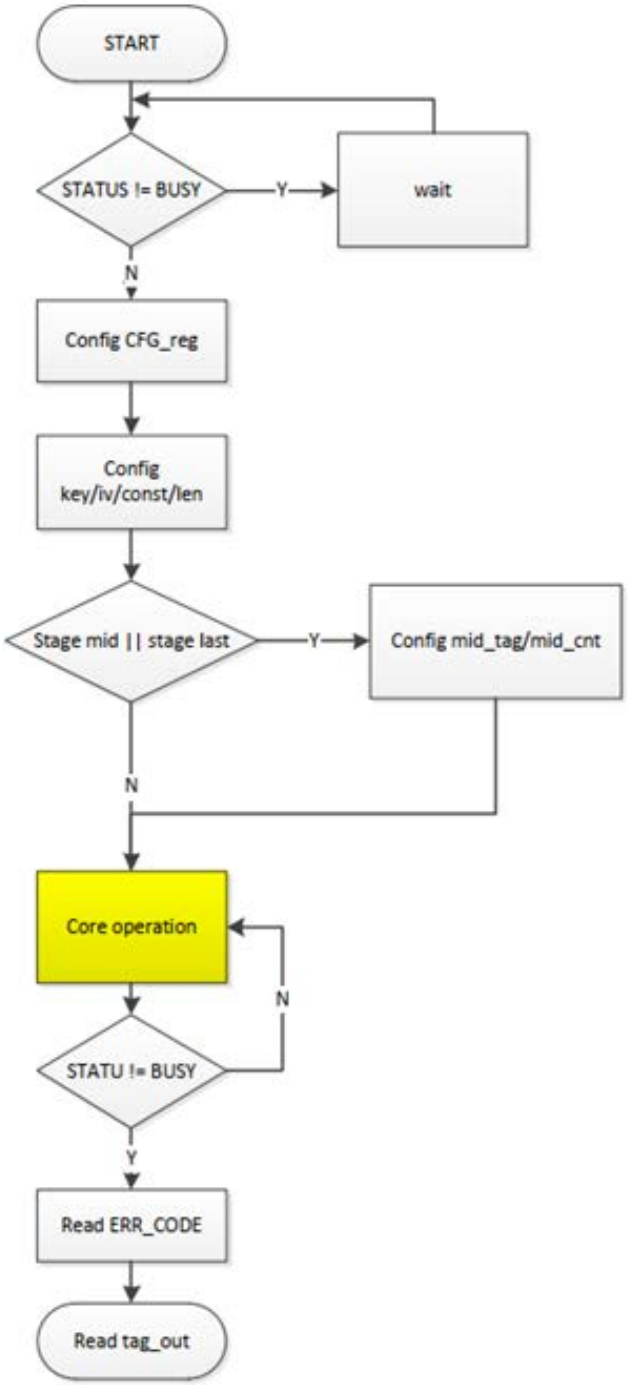
tag_out[127:96] is mapping to register address 0x11C.

tag_out[129:128] is mapping to register address 0x120.

25.2.5 Usage Flow

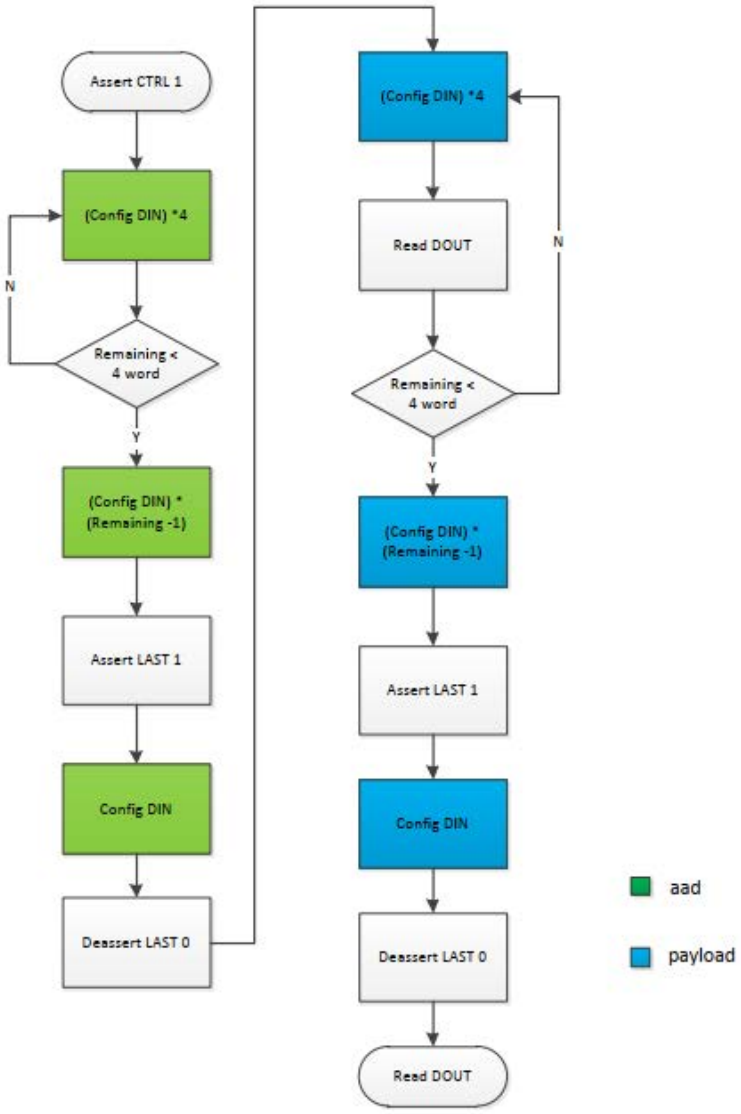
This usage flow of this module is described in following figures.

Figure 25-8 Usage Flow of Chacha20



Core operation in CPU mode.

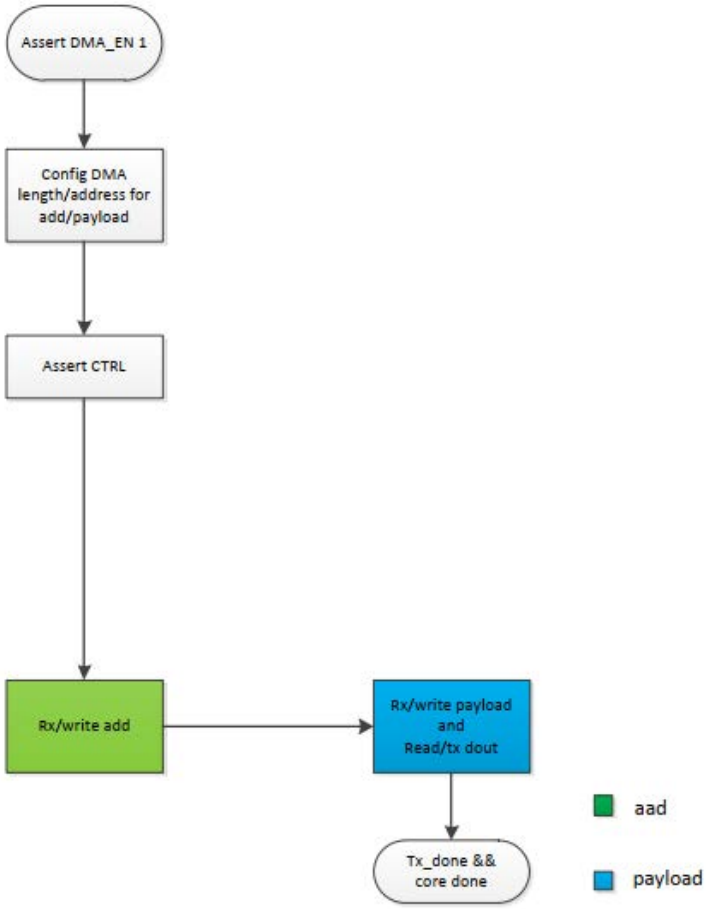
Figure 25-9 Chacha20 Core Operation in CPU Mode



Core operation in DMA mode.

In some cases, the length of aad or payload may be required to be 0. At this time, DMA also needs to output the corresponding Done signal after DMA starts.

Figure 25-10 Chacha20 Core Operation in DMA Mode



25.3 CHACHA20 Register Description

The CHACHA20 related registers are listed in the following table. The base address for the following CHACHA20 related registers is 0x80105000.

Table 25-1 CHACHA20 Related Registers

Address Offset	Name	Type	Description	Default Value
0x00	CHACHA20_CTRL_B0	W1S	[0]: start, start calculation, active high	0x00

Address Offset	Name	Type	Description	Default Value
0x04	CHACHA20_CFG_B0	RW	[1:0]: SEC_STAGE, indicate data interleave stage 0: one stage, din contains all chacha20_poly1305 data. 1: stage init. In current stage, din must inputs all AAD. Payload is not required. If payload exist, it must be aligned to 64 byte. 2: stage middle. In current stage, din is part payload of a transition. It must be aligned to 64 bytes. 3: stage last. In current stage, user can inputs payload if there are payload. This stage supports empty payload. [2]: DECODE,indicate decode or encode mode. 1: decode mode, 0: encode mode [3]: DMA_EN, enable DMA function, active high	0x00
0x08	CHACHA20_SOFT_RSTNB0	WO	[0]:SOFT_RST_N, soft reset calculate_core and related control logic, active low. Writing 0 resets the core and control logic for 1 cycle. After 1 cycle, reset is released automatically. In DMA mode, users need to reset DMA outside synchronously.	0x01
0x0c	CHACHA20_DIN_RDY	RO	[0]: DIN_RDY, ready signal of data in 0: can not send message to core 1: can send message to core	0x00
0x10	CHACHA20_RISR_B0	W0C	[0]: RISR, indicate calculation done, when a dout or tag_out is generated, this register indicates that the calculation is done. It is mainly used to indicate the output of dout in CPU mode. 0: cal undone 1: cal done	0x00
0x14	CHACHA20_IMCR_B0	RW	[0]: IMCR, indicate the irq signal output. 0:disable output 1:enable output	0x00

Address Offset	Name	Type	Description	Default Value
0x18	CHACHA20_STATUS_B0	RO	[0]: STATUS, indicate core's status, the core remains busy until the tag_out is generated. It is mainly used for DMA mode to indicate the core has completed all the operations after start. 0: core is idle 1: core is busy	0x00
0x1c	CHACHA20_ERR_CODE_B0	RO	[2:0]: ERR_CODE, indicate core's error code. 1: current stage byte length of aad and payload are 0, but section_stage is 1; 2: current stage byte length of last aad > 0, which means aad is not empty, but section_stage is 2 or 3; 3: current stage byte length of payload is 0, but sec tion_stage is 2; 4: current length of payload is not align to 64 bytes, but section_stage is 1 or 2; 5: write counter of chacha20 to 0 when stage is mid dle or last; 6: when stage is 0, length[133:128] is 0, but length[127:0] is not 0; or length[127:0] is 0, but length[133:128] is not 0.	0x00
0x20	CHACHA20_LAST_B0	RW	[0]: LAST, indicate whether the next data is the end data. 0: not the end data 1: the end data	0x00
0x24	CHACHA20_SE_DONE_B0	RO	[0]: SE_DONE, indicate the secure key is configured or not. 0: not configured or configuring 1: configuration done	0x00
0x30	CHACHA20_KEY_B0	WO	[7:0]: key	0x00
0x31	CHACHA20_KEY_B1	WO	[7:0]: key	0x00
0x32	CHACHA20_KEY_B2	WO	[7:0]: key	0x00
0x33	CHACHA20_KEY_B3	WO	[7:0]: key	0x00

Address Offset	Name	Type	Description	Default Value
0x34	CHACHA20_KEY_B4	WO	[7:0]: key	0x00
0x35	CHACHA20_KEY_B5	WO	[7:0]: key	0x00
0x36	CHACHA20_KEY_B6	WO	[7:0]: key	0x00
0x37	CHACHA20_KEY_B7	WO	[7:0]: key	0x00
0x38	CHACHA20_KEY_B8	WO	[7:0]: key	0x00
0x39	CHACHA20_KEY_B9	WO	[7:0]: key	0x00
0x3a	CHACHA20_KEY_B10	WO	[7:0]: key	0x00
0x3b	CHACHA20_KEY_B11	WO	[7:0]: key	0x00
0x3c	CHACHA20_KEY_B12	WO	[7:0]: key	0x00
0x3d	CHACHA20_KEY_B13	WO	[7:0]: key	0x00
0x3e	CHACHA20_KEY_B14	WO	[7:0]: key	0x00
0x3f	CHACHA20_KEY_B15	WO	[7:0]: key	0x00
0x40	CHACHA20_KEY_B16	WO	[7:0]: key	0x00
0x41	CHACHA20_KEY_B17	WO	[7:0]: key	0x00
0x42	CHACHA20_KEY_B18	WO	[7:0]: key	0x00
0x43	CHACHA20_KEY_B19	WO	[7:0]: key	0x00
0x44	CHACHA20_KEY_B20	WO	[7:0]: key	0x00
0x45	CHACHA20_KEY_B21	WO	[7:0]: key	0x00
0x46	CHACHA20_KEY_B22	WO	[7:0]: key	0x00
0x47	CHACHA20_KEY_B23	WO	[7:0]: key	0x00
0x48	CHACHA20_KEY_B24	WO	[7:0]: key	0x00
0x49	CHACHA20_KEY_B25	WO	[7:0]: key	0x00
0x4a	CHACHA20_KEY_B26	WO	[7:0]: key	0x00
0x4b	CHACHA20_KEY_B27	WO	[7:0]: key	0x00
0x4c	CHACHA20_KEY_B28	WO	[7:0]: key	0x00
0x4d	CHACHA20_KEY_B29	WO	[7:0]: key	0x00
0x4e	CHACHA20_KEY_B30	WO	[7:0]: key	0x00

Address Offset	Name	Type	Description	Default Value
0x4f	CHACHA20_KEY_B31	WO	[7:0]: key	0x00
0x50	CHACHA20_IV_B0	WO	[7:0]: iv	0x00
0x51	CHACHA20_IV_B1	WO	[7:0]: iv	0x00
0x52	CHACHA20_IV_B2	WO	[7:0]: iv	0x00
0x53	CHACHA20_IV_B3	WO	[7:0]: iv	0x00
0x54	CHACHA20_IV_B4	WO	[7:0]: iv	0x00
0x55	CHACHA20_IV_B5	WO	[7:0]: iv	0x00
0x56	CHACHA20_IV_B6	WO	[7:0]: iv	0x00
0x57	CHACHA20_IV_B7	WO	[7:0]: iv	0x00
0x58	CHACHA20_CONST_B0	WO	[7:0]: const	0x00
0x59	CHACHA20_CONST_B1	WO	[7:0]: const	0x00
0x5a	CHACHA20_CONST_B2	WO	[7:0]: const	0x00
0x5b	CHACHA20_CONST_B3	WO	[7:0]: const	0x00
0x60	CHACHA20_LENTH_B0	WO	[7:0]: lenth	0x00
0x61	CHACHA20_LENTH_B1	WO	[7:0]: lenth	0x00
0x62	CHACHA20_LENTH_B2	WO	[7:0]: lenth	0x00
0x63	CHACHA20_LENTH_B3	WO	[7:0]: lenth	0x00
0x64	CHACHA20_LENTH_B4	WO	[7:0]: lenth	0x00
0x65	CHACHA20_LENTH_B5	WO	[7:0]: lenth	0x00
0x66	CHACHA20_LENTH_B6	WO	[7:0]: lenth	0x00
0x67	CHACHA20_LENTH_B7	WO	[7:0]: lenth	0x00
0x68	CHACHA20_LENTH_B8	WO	[7:0]: lenth	0x00
0x69	CHACHA20_LENTH_B9	WO	[7:0]: lenth	0x00
0x6a	CHACHA20_LENTH_B10	WO	[7:0]: lenth	0x00
0x6b	CHACHA20_LENTH_B11	WO	[7:0]: lenth	0x00
0x6c	CHACHA20_LENTH_B12	WO	[7:0]: lenth	0x00
0x6d	CHACHA20_LENTH_B13	WO	[7:0]: lenth	0x00

Address Offset	Name	Type	Description	Default Value
0x6e	CHACHA20_LENTH_B14	WO	[7:0]: lenth	0x00
0x6f	CHACHA20_LENTH_B15	WO	[7:0]: lenth	0x00
0x70	CHACHA20_LENTH_B16	WO	[7:0]: lenth	0x00
0x71	CHACHA20_LENTH_B17	WO	[7:0]: lenth	0x00
0x72	CHACHA20_LENTH_B18	WO	[7:0]: lenth	0x00
0x73	CHACHA20_LENTH_B19	WO	[7:0]: lenth	0x00
0x74	CHACHA20_CNT_B0	WO	[7:0]: cnt	0x00
0x75	CHACHA20_CNT_B1	WO	[7:0]: cnt	0x00
0x76	CHACHA20_CNT_B2	WO	[7:0]: cnt	0x00
0x77	CHACHA20_CNT_B3	WO	[7:0]: cnt	0x00
0x80	CHACHA20_TAG_IN_B0	WO	[7:0]: tag_in	0x00
0x81	CHACHA20_TAG_IN_B1	WO	[7:0]: tag_in	0x00
0x82	CHACHA20_TAG_IN_B2	WO	[7:0]: tag_in	0x00
0x83	CHACHA20_TAG_IN_B3	WO	[7:0]: tag_in	0x00
0x84	CHACHA20_TAG_IN_B4	WO	[7:0]: tag_in	0x00
0x85	CHACHA20_TAG_IN_B5	WO	[7:0]: tag_in	0x00
0x86	CHACHA20_TAG_IN_B6	WO	[7:0]: tag_in	0x00
0x87	CHACHA20_TAG_IN_B7	WO	[7:0]: tag_in	0x00
0x88	CHACHA20_TAG_IN_B8	WO	[7:0]: tag_in	0x00
0x89	CHACHA20_TAG_IN_B9	WO	[7:0]: tag_in	0x00
0x8a	CHACHA20_TAG_IN_B10	WO	[7:0]: tag_in	0x00
0x8b	CHACHA20_TAG_IN_B11	WO	[7:0]: tag_in	0x00
0x8c	CHACHA20_TAG_IN_B12	WO	[7:0]: tag_in	0x00
0x8d	CHACHA20_TAG_IN_B13	WO	[7:0]: tag_in	0x00
0x8e	CHACHA20_TAG_IN_B14	WO	[7:0]: tag_in	0x00
0x8f	CHACHA20_TAG_IN_B15	WO	[7:0]: tag_in	0x00
0x90	CHACHA20_TAG_IN_B16	WO	[7:0]: tag_in	0x00

Address Offset	Name	Type	Description	Default Value
0x91	CHACHA20_TAG_IN_B17	WO	[7:0]: tag_in	0x00
0x92	CHACHA20_TAG_IN_B18	WO	[7:0]: tag_in	0x00
0x93	CHACHA20_TAG_IN_B19	WO	[7:0]: tag_in	0x00
0x94	CHACHA20_DIN_B0	WO	[7:0]: din, data_in	0x00
0x95	CHACHA20_DIN_B1	WO	[7:0]: din, data_in	0x00
0x96	CHACHA20_DIN_B2	WO	[7:0]: din, data_in	0x00
0x97	CHACHA20_DIN_B3	WO	[7:0]: din, data_in	0x00
0xfc	CHACHA20_VERSION_B0	RO	[3:0]: MIR [7:4]: MAR	0x11
0xfe	CHACHA20_VERSION_B2	RO	[7:0]: PROJECT	0xe4
0xff	CHACHA20_VERSION_B3	RO	[7:0]: PROJECT	0x00
0x100	CHACHA20_DOUT_B0	RO	[7:0]: dout	0x00
0x101	CHACHA20_DOUT_B1	RO	[7:0]: dout	0x00
0x102	CHACHA20_DOUT_B2	RO	[7:0]: dout	0x00
0x103	CHACHA20_DOUT_B3	RO	[7:0]: dout	0x00
0x104	CHACHA20_DOUT_B4	RO	[7:0]: dout	0x00
0x105	CHACHA20_DOUT_B5	RO	[7:0]: dout	0x00
0x106	CHACHA20_DOUT_B6	RO	[7:0]: dout	0x00
0x107	CHACHA20_DOUT_B7	RO	[7:0]: dout	0x00
0x108	CHACHA20_DOUT_B8	RO	[7:0]: dout	0x00
0x109	CHACHA20_DOUT_B9	RO	[7:0]: dout	0x00
0x10a	CHACHA20_DOUT_B10	RO	[7:0]: dout	0x00
0x10b	CHACHA20_DOUT_B11	RO	[7:0]: dout	0x00
0x10c	CHACHA20_DOUT_B12	RO	[7:0]: dout	0x00
0x10d	CHACHA20_DOUT_B13	RO	[7:0]: dout	0x00
0x10e	CHACHA20_DOUT_B14	RO	[7:0]: dout	0x00

Address Offset	Name	Type	Description	Default Value
0x10f	CHACHA20_DOUT_B15	RO	[7:0]: dout	0x00
0x110	CHACHA20_TOUT_B0	RO	[7:0]: tout, tag_out	0x00
0x111	CHACHA20_TOUT_B1	RO	[7:0]: tout, tag_out	0x00
0x112	CHACHA20_TOUT_B2	RO	[7:0]: tout, tag_out	0x00
0x113	CHACHA20_TOUT_B3	RO	[7:0]: tout, tag_out	0x00
0x114	CHACHA20_TOUT_B4	RO	[7:0]: tout, tag_out	0x00
0x115	CHACHA20_TOUT_B5	RO	[7:0]: tout, tag_out	0x00
0x116	CHACHA20_TOUT_B6	RO	[7:0]: tout, tag_out	0x00
0x117	CHACHA20_TOUT_B7	RO	[7:0]: tout, tag_out	0x00
0x118	CHACHA20_TOUT_B8	RO	[7:0]: tout, tag_out	0x00
0x119	CHACHA20_TOUT_B9	RO	[7:0]: tout, tag_out	0x00
0x11a	CHACHA20_TOUT_B10	RO	[7:0]: tout, tag_out	0x00
0x11b	CHACHA20_TOUT_B11	RO	[7:0]: tout, tag_out	0x00
0x11c	CHACHA20_TOUT_B12	RO	[7:0]: tout, tag_out	0x00
0x11d	CHACHA20_TOUT_B13	RO	[7:0]: tout, tag_out	0x00
0x11e	CHACHA20_TOUT_B14	RO	[7:0]: tout, tag_out	0x00
0x11f	CHACHA20_TOUT_B15	RO	[7:0]: tout, tag_out	0x00
0x120	CHACHA20_TOUT_B16	RO	[7:0]: tout, tag_out	0x00
0x121	CHACHA20_TOUT_B17	RO	[7:0]: tout, tag_out	0x00
0x122	CHACHA20_TOUT_B18	RO	[7:0]: tout, tag_out	0x00
0x123	CHACHA20_TOUT_B19	RO	[7:0]: tout, tag_out	0x00
0x124	CHACHA20_COUT_B0	RO	[7:0]: cout, counter_out	0x00
0x125	CHACHA20_COUT_B1	RO	[7:0]: cout, counter_out	0x00
0x126	CHACHA20_COUT_B2	RO	[7:0]: cout, counter_out	0x00
0x127	CHACHA20_COUT_B3	RO	[7:0]: cout, counter_out	0x00
0x1c0	CHACHA20_DMA_SADDR_A_B0	RW	[7:0]: dma_saddr_a_b0	

Address Offset	Name	Type	Description	Default Value
0x1c1	CHACHA20_DMA_SADDR_A_B1	RW	[7:0]: dma_saddr_a_b1	
0x1c2	CHACHA20_DMA_SADDR_A_B2	RW	[7:0]: dma_saddr_a_b2	
0x1c3	CHACHA20_DMA_SADDR_A_B3	RW	[7:0]: dma_saddr_a_b3	
0x1c4	CHACHA20_DMA_RLEN_A_B0	RW	[7:0]: dma_rlen_a, DMA byte length of reading aad	0x00
0x1c5	CHACHA20_DMA_RLEN_A_B1	RW	[7:0]: dma_rlen_a, DMA byte length of reading aad	0x00
0x1c6	CHACHA20_DMA_RLEN_A_B2	RW	[7:0]: dma_rlen_a, DMA byte length of reading aad	0x00
0x1c7	CHACHA20_DMA_RLEN_A_B3	RW	[7:0]: dma_rlen_a, DMA byte length of reading aad	0x00
0x1d8	CHACHA20_DMA_LEN_D_B0	RW	[7:0]: dma_len_d, DMA byte length of payload, the write length and read length of payload are equal	0x00
0x1d9	CHACHA20_DMA_LEN_D_B1	RW	[7:0]: dma_len_d, DMA byte length of payload, the write length and read length of payload are equal	0x00
0x1da	CHACHA20_DMA_LEN_D_B2	RW	[7:0]: dma_len_d, DMA byte length of payload, the write length and read length of payload are equal	0x00
0x1db	CHACHA20_DMA_LEN_D_B3	RW	[7:0]: dma_len_d, DMA byte length of payload, the write length and read length of payload are equal	0x00

The additional CHACHA20 related registers are listed in the following table. The base address for the following registers is 0x80100800.

Table 25-2 Additional CHACHA20 Related Registers

Address Offset	Name	Type	Description	Reset Value
0x10	ROSR_CTRL10	RW	[7:0]:chacha20_fifo_b0, CHACHA20 DMA mode read/write data entry	0x00

Address Offset	Name	Type	Description	Reset Value
0x11	ROSR_CTRL11	RW	[7:0]:chacha20_fifo_b1, CHACHA20 DMA mode read/write data entry	0x00
0x12	ROSR_CTRL12	RW	[7:0]:chacha20_fifo_b2, CHACHA20 DMA mode read/write data entry	0x00
0x13	ROSR_CTRL13	RW	[7:0]:chacha20_fifo_b3, CHACHA20 DMA mode read/write data entry	0x00
0x14	ROSR_CTRL14	RW	[0]: chacha20_txdma_en [1]: chacha20_rxdma_en	0x00
0x15	ROSR_CTRL15	RW	[3:0]: chacha20_txf_thres [7:4]: chacha20_rxf_thres	0x11

26 Secure Boot, Firmware Encryption and Secure Debug

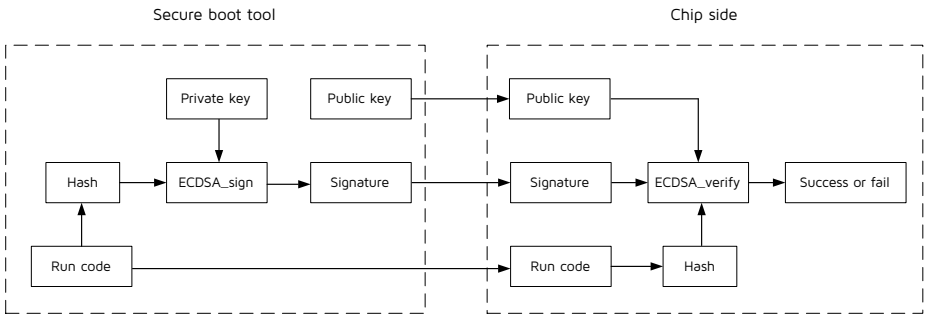
26.1 Introduction

The SoC supports security solution including secure boot, firmware encryption and secure debug.

The Secure Boot prevents the chip from running any unauthorized firmware by checking that the firmware being booted is verified by Elliptic Curve Digital Signature Algorithm (ECDSA). The key pair (public key/private key) is generated by Elliptic Curve Cryptography (ECC). The user signs the firmware with a private key through the secure boot tool, and the chip side verifies the signature with a public key before running the firmware.

The signature verification ensures that only verified code can be executed, otherwise the firmware is considered tampered and is not executed.

Figure 26-1 Secure boot verification flow

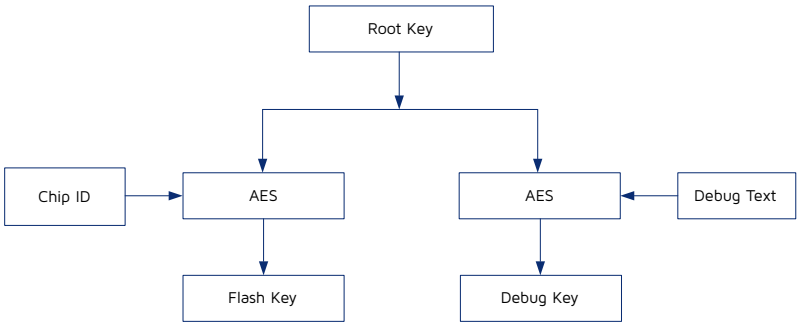


In order to protect the firmware from being cloned, the firmware encryption can be used. The firmware is encrypted by hardware adopts an AES-128-like Light Crypto encryption algorithm when it is downloaded to flash. The firmware stored in the flash is Ciphertext. When the chip is running, it decrypts the firmware in real time.

The Secure Debug allows all debug interfaces such as SWS and JTAG to be locked so that hackers is not able to access any on-chip information (registers or memories) from these debug interfaces. The user can use the tool to re-enable debug interface again by sending debug key sequence to the SWS interface according to the single wire protocol.

26.2 Key Management

Figure 26-2 Key Management



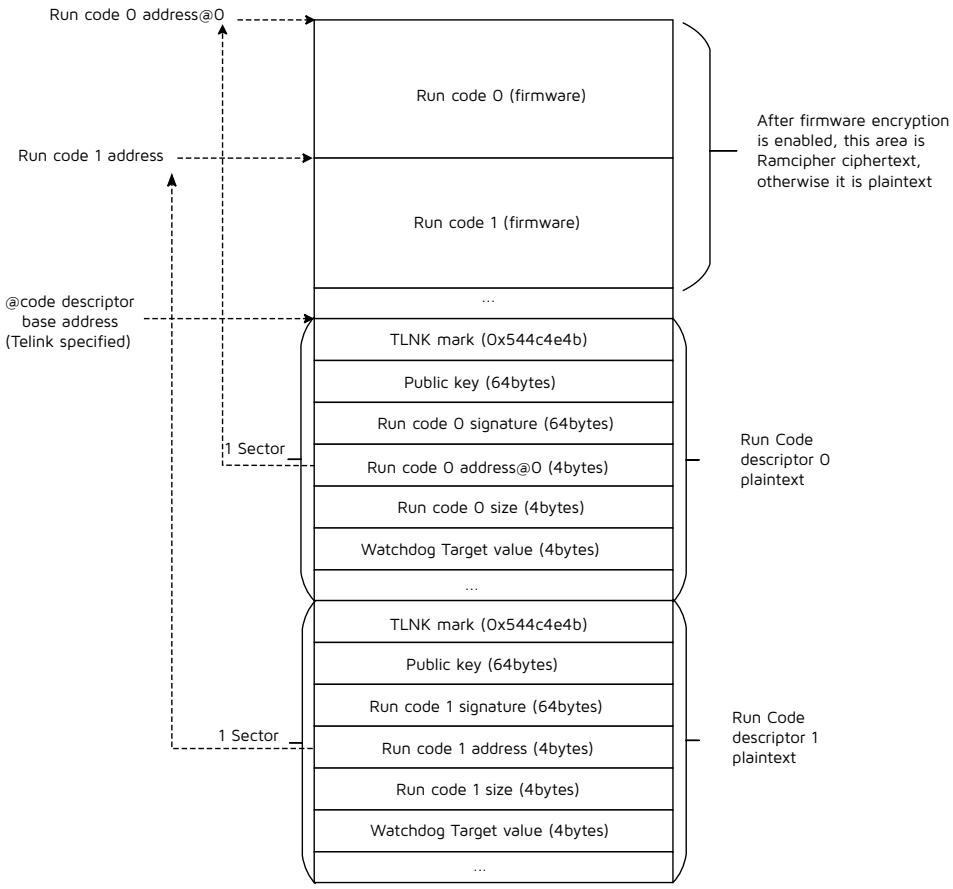
The Flash Key is used for Firmware Encryption, the Debug Key is used to re-enable debug interface, shown as the figure above. The Flash Key is derived from the Root Key (provided by customer) and the Chip ID (unmodifiable) through AES128 (ECB, Electronic Code Book) derivation. The Debug Key is derived from the Root Key (provided by customer) and the Debug Text (provided by customer) through AES128 (ECB) derivation.

The chip has a key lock mechanism, once the key lock function is enabled, the software cannot read the root key and debug text in the OTP.

26.3 Flash Space and OTP Definition

26.3.1 Flash Space in Secure Boot Mode

Figure 26-3 Flash Space for Secure Boot



The flash contains several segments of data including code 0 & 1, and their corresponding code descriptor block 0 & 1. When allocating flash, each segment area should not overlap, and the first address of each segment address is recommended to be aligned with the smallest erase unit of flash. The code descriptor block is only present if secure boot with firmware signature verification is enabled.

The run code 0, which has a fixed starting address of 0, is the firmware code to be executed. The starting address of run code 1 should be an integer multiple of 4K bytes. It is in plaintext if firmware encryption is not

enabled; If the firmware encryption is enabled, the code is stored as ciphertext (encrypted firmware). The code is automatically loaded after power on, and the user should leave a corresponding size of flash space.

The code descriptor block provides information about the corresponding firmware code.

- The Telink Mark is a fixed string pattern stored in the register of 0x544c4e4b. Bootloader only execute the corresponding code 0 or code 1 with a valid corresponding Telink Mark, and realize switching the code to be executed.
- The public key is the public key used to verify the code signature.
- The run code signature is the calculated signature over the running code.
- The run code address is the starting address of the running code.
- The run code size is the length of the code field in Bytes.
- The watchdog target value is the watchdog capture value. This value should be set large enough to allow firmware signature verification process to finish before watchdog resets, please use the default value provided by Telink.

26.3.2 OTP Definition in Secure Boot Mode

The OTP definition in secure boot mode is shown in the table below. Please also check 4.1.3 OTP for details on OTP.

Table 26-1 OTP Data for Secure Boot

Definition	Length (Byte)	Description
mode_selection	4	[31:1] Reserved [0] mode selection, 0: secure boot mode (Signature Verification), 1: normal mode
public_key_hash	32	Public key hash, is the hash calculated over the public key used by customer for firmware signature verification. Once provisioned by the customer, it cannot be changed. It is used by the chip to verify that the correct public key for firmware signature verification is used.
Chip_ID	16	Unique chip ID
Debug_Text	16	Debug Text, is a text string provided by the customer to control the debug lock feature for a customer project
root_key	16	Root key, a customer provided key for Flash key and Debug key derivation
firmware_encryption	4	[31:1] Reserved [0] Firmware_encryption_disable, 1'b0: firmware encryption enable 1'b1: firmware encryption disable

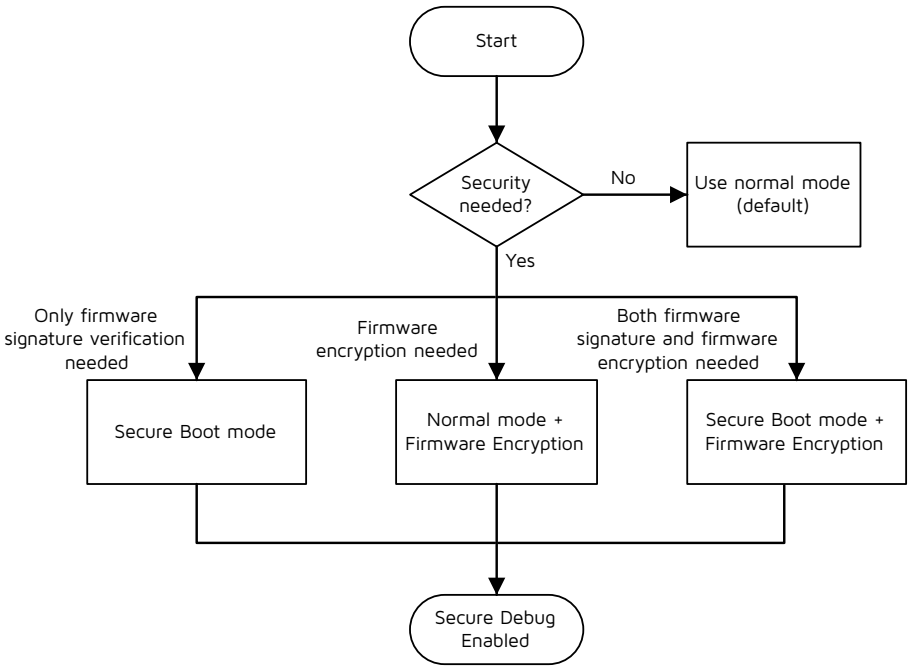
Definition	Length (Byte)	Description
Interface functions	4	[31:5] Reserved [4] key_lock_enable 1'b0: otp key can't be read; 1'b1: otp key can be read [3] sspi_dbg_enable, SPI slave function enable: 1'b0: disable; 1'b1: enable. [2] sws_dbg_enable, SWS function enable: 1'b0: disable; 1'b1: enable. [1] jtag_dbg_enable, JTAG function enable: 1'b0: disable; 1'b1: enable. [0] usb_dbg_enable, USB debug function enable: 1'b0: disable; 1'b1: enable.

26.4 Usage

The chip has two modes: normal mode and secure boot mode. The two modes are configured by bit [0] of mode_selection in OTP. If bit [0] is 0, it is secure boot mode. If bit [0] is 1, it is normal mode.

The following decision tree can be used to select appropriate secure mechanisms.

Figure 26-4 Decision Tree for Secure Boot



In normal mode, the firmware runs from address offset 0K/64K/128K/256K/512K/1M/2M/4M/8M bytes of Flash without the code descriptor. In secure boot mode, the firmware needs to verify the signature with the code descriptor.

When security is required, typically these 3 combinations below satisfy most use cases.

- Firmware Signature Verification + Secure Debug
- Firmware Encryption + Secure Debug
- Firmware Signature Verification +Firmware Encryption +Secure Debug

There may be other possible configuration bits combinations, but it is not recommended using them except for the above cases.

1. Signature verification only without encryption: Firmware encryption disabled, mode selection set to secure mode.

   In this case, firmware is stored in plaintext on the Flash, the code descriptor information is used to verify its signature.

2. Encryption only without signature verification: Firmware encryption enabled, mode selection set to normal mode.

   In this case, the firmware stored on the Flash is encrypted. It is decrypted in real-time during running.

3. Encryption and signature verification: Firmware encryption enabled, mode selection set to secure mode.

   In this case, firmware is stored in ciphertext on the Flash, and the code descriptor information is used to verify its signature on the original plaintext.

After setting up the case above, disable all the debugging interfaces including sspi_dbg_disable, sws_dbg_enable, jtag_dbg_enable, and usb_dbg_enable.

If the user wants to re-enable debug interface function, send Debug Key sequence to SWS interface according to the single wire protocol.