Datasheet for Telink

Multi-Standard Wireless SoC

TLSR9228

DS-TLSR9228-E5

Ver 0.5.1

2023/08/31

Keyword

Bluetooth LE; Indoor positioning; Bluetooth LE Mesh; Zigbee; HomeKit; 6LoWPAN; Thread; Matter; 2.4 GHz

Brief

This datasheet is dedicated for Telink multi-standard wireless SoC TLSR9228.

In this datasheet, function block diagram, key features, typical applications, electrical specifications and details of each function module for TLSR9228 are introduced.

Published by

Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,

Zhangjiang Hi-Tech Park, Shanghai, China

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
0.1.1	• Section 1.1 Block Diagram: Added QDEC to Figure 1-1; updated SRAM size to 512 KB and added voice activity detection, capacity touch button interface, QDEC to the description of the system • Section 1.2.1 General Features: Added the cache size for instruction and data cache controller; updated maximum flash size and SRAM size, bit width of RSA; removed AES-CCM support; added a low-frequency 2 MHz ~ 4 MHz clock to RTC and other timers; updated GPIO number, SPI channels, DMIC, AMIC, I2S, stereo audio codec, I2C, UART and package; added QDEC and capacity touch button interface • Section 1.2.3 Features of Power Management Module: Updated power consumption • Section 1.2.4 Bluetooth Features: Updated Bluetooth ISO channel support and removed audio support • Section 1.2.7 HomeKit Features: Updated Apple HomeKit ADK support • Section 1.2.8 Wireless Audio Features: Updated the maximum sampling rate of stereo audio codec and added voice activity detection • Section 2.1 DC Characteristics: Updated Table 2-1
0.1.2	• Some minor edits and corrections
0.5.0	• Added Chapter 3 to Chapter 20
0.5.1	• Added 1.2.9 Matter Features, Chapter 8 Trap and PLIC, Chapter 9 DMA, Chapter 19 Quadrature Decoder • Updated Table 2-2 Recommend Operating Conditions • Other minor edits and corrections

Table of Contents

1 Overview 18
1.1 Block Diagram 18
1.2 Key Features 19
1.2.1 General Features 19
1.2.2 CPU and Memory 20
1.2.3 RF Features 20
1.2.4 Features of Power Management Module 21
1.2.5 Bluetooth Features 21
1.2.6 Zigbee Features 22
1.2.7 OpenThread Features 22
1.2.8 HomeKit Features 22
1.2.9 Matter Features 22
1.2.10 Concurrent Mode Feature 23
1.3 Typical Applications 23
1.4 Ordering Information 23
1.5 Package 23
1.6 Pin Layout 25
2 Electrical Specifications 35
2.1 Absolute Maximum Rating 35
2.2 Recommended Operating Conditions 35
2.3 DC Characteristics 35
2.4 AC Characteristics 37
2.4.1 RF Performance 37
2.5 Storage Conditions 42
3 Reference Design 43
3.1 Reference Schematic for TLSR9228H 43
3.2 BOM (Bill of Material) for TLSR9228H 43
4 Memory, MCU and PMU 45
4.1 Memory 45
4.1.1 SRAM 45
4.1.2 Flash 47
4.1.3 eFuse 47
4.1.4 Unique ID 48
4.2 MCU 48
4.2.1 Physical Memory Protection 48
4.3 Working Mode 51

4.4 Reset 54
4.5 Power Management 55
4.5.1 Power-on-Reset (POR) and Brown-out Detect 55
4.5.2 Working Mode Switch 59
4.5.3 LDO and DCDC 60
4.5.4 VBAT and VANT Power-Supply Mode 61
4.6 Wakeup Source 61
5 BLE/2.4GHz RF Transceiver 64
5.1 Overview 64
5.2 Air Interface Data Rate and RF Channel Frequency 64
5.3 Baseband 65
5.3.1 Packet Format 65
5.3.2 BLE Location Function - Direction 66
5.3.3 RSSI and Frequency Offset 66
6 Clock 67
6.1 Clock Sources 67
6.2 System Clock 69
6.3 Module Clock 69
6.3.1 System Timer Clock 69
6.3.2 USB Clock 69
6.3.3 I2S0 Clock 69
6.3.4 I2S1 Clock 69
6.3.5 DMIC Clock 69
6.3.6 clkzb32k 70
6.3.7 clk_7816 70
6.3.8 clk_mspi 70
6.3.9 clk_lspi 70
6.3.10 clk_gspi 70
6.4 Register Table 70
7 Timer 74
7.1 Timer0 ~ Timer1 74
7.1.1 Mode 0 (System Clock Mode) 74
7.1.2 Mode 1 (GPIO Trigger Mode) 74
7.1.3 Mode 2 (GPIO Pulse Width Mode) 75
7.1.4 Mode 3 (Tick Mode) 76
7.1.5 Watchdog 76
7.1.6 Register Table 76
7.2 32K LTimer 78

7.3 System Timer 80
7.4 Platform-Level Machine Timer (PLMT) 81
7.4.1 Introduction 81
7.4.2 Access To Mtime 81
7.4.3 Access To Mtimecmp 82
7.4.4 Register Description 82
8 Trap and PLIC 83
8.1 Trap 83
8.1.1 Introduction 83
8.1.2 Interrupt 83
8.1.2.1 Local Interrupts 83
8.1.2.2 Interrupt Status and Masking 84
8.1.2.3 Interrupt Priority 84
8.1.3 Exception 84
8.1.4 Trap Handling 85
8.1.4.1 Entering the Trap Handler 85
8.1.4.2 Returning from the Trap Handler 85
8.1.5 Machine Trap Related CSRs 85
8.1.5.1 Machine Status 85
8.1.5.2 Machine Interrupt Enable 86
8.1.5.3 Machine Trap Vector Base Address 86
8.1.5.4 Machine Exception Program Counter 87
8.1.5.5 Machine Cause Register 87
8.1.5.6 Machine Trap Value 88
8.1.5.7 Machine Interrupt Pending 89
8.1.5.8 Machine Detailed Trap Cause 89
8.1.5.9 Machine Miscellaneous Control Register 90
8.2 Platform-Level Interrupt Controller (PLIC) 91
8.2.1 Introduction 91
8.2.2 External Interrupt Sources 93
8.2.3 Support for Preemptive Priority Interrupt 94
8.2.3.1 Interrupt Claims with Preemptive Priority 94
8.2.3.2 Interrupt Completion with Preemptive Priority 94
8.2.3.3 Programming Sequence to Allow Preemption of Interrupts 94
8.2.4 Vectored Interrupts 96
8.2.5 Support for Software-Generated Interrupt 96
8.2.6 Interrupt Flow 96
8.2.7 Register Description 97

8.3 Software Platform-Level Interrupt Controller (PLIC_SW) 99
8.3.1 Introduction 99
8.3.2 Register Description 99
9 DMA 101
9.1 Introduction 101
9.1.1 Function Description 101
9.1.1.1 Channel Arbitration 102
9.1.1.2 Chain Transfer 102
9.1.1.3 Data Order 103
9.2 Registers 103
9.2.1 Register Summary 103
9.2.2 Interrupt Status Register (Offset 0x30) 104
9.2.3 Channel Abort Register (Offset 0x40) 105
9.2.4 Channel n Control Register (Offset 0x44+n*0x14) 105
9.2.5 Channel n Source Address Register (Offset 0x48+n*0x14) 109
9.2.6 Channel n Destination Address Register (Offset 0x4C+n*0x14) 109
9.2.7 Channel n Transfer Size Register (Offset 0x50+n*0x14) 110
9.2.8 Channel n Linked List Pointer Register (Offset 0x54+n*0x14) 110
9.2.9 Baseband Related Register 110
9.2.10 Miscellaneous Register 113
9.3 Usage Guide 113
9.3.1 From SRAM to SRAM 113
9.3.2 From SRAM to Peripherals 114
9.3.3 From Peripherals to SRAM 115
9.3.4 From SRAM to Baseband 116
9.3.5 From Baseband to SRAM 117
10 Interface 118
10.1 GPIO 118
10.1.1 Basic Configuration 118
10.1.2 Connection Relationship between GPIO and Related Modules 123
10.1.3 Drive Strength 127
10.1.4 Polarity 128
10.1.5 GPIO IRQ Signal 128
10.1.6 GPIO2RISC IRQ Signal 128
10.1.7 GPIO GROUP IRQ Signal 129
10.1.8 Pull-up/Pull-down Resistors 129
10.1.9 GPIO Driving LED Description 131
10.2 Swire 132

10.3 I2C 133
10.3.1 Communication Protocol 133
10.3.2 I2C Slave Mode 134
10.3.3 I2C Master Mode 134
10.3.3.1 I2C Master Write Transfer in NDMA Mode 135
10.3.3.2 I2C Master Read Transfer in NDMA Mode 135
10.3.3.3 I2C Master Writer Transfer in DMA Mode 135
10.3.3.4 I2C Master Read Transfer in DMA Mode 135
10.3.4 Register Description 135
10.4 Memory SPI 139
10.4.1 Memory SPI Diagram 139
10.4.2 MSPI Register Description 140
10.5 LSPI 145
10.5.1 LSPI Diagram 145
10.5.2 LSPI Register Description 146
10.6 GSPI 156
10.6.1 GSPI Diagram 156
10.6.2 GSPI Register Description 157
10.7 SPI Slave (SPI_SLV) 165
10.7.1 Diagram 165
10.7.2 Features 166
10.7.3 Function Description 166
10.8 UART 167
10.8.1 Introduction 167
10.8.2 Block Diagram 168
10.8.3 Function Description 168
10.8.3.1 Pin Configuration 168
10.8.3.2 Transmitter 168
10.8.3.3 Receiver 169
10.8.3.4 Baud Rate Generator 169
10.8.3.5 Loopback Mode 170
10.8.3.6 Error Conditions 171
10.8.3.7 Hardware Flow Control 171
10.8.3.8 Receiver Timeout 171
10.8.4 Register Description 172
10.9 USB 177
11 PWM 185
11.1 Enable PWM 185

11.2 Set PWM Clock 185
11.3 PWM Waveform, Polarity and Output Inversion 185
11.3.1 Waveform of Signal Frame 185
11.3.2 Invert PWM Output 186
11.3.3 Polarity for Signal Frame 186
11.4 PWM Mode 186
11.4.1 Select PWM Modes 186
11.4.2 Continuous Mode 186
11.4.3 Counting Mode 187
11.4.4 IR Mode 187
11.4.5 IR FIFO Mode 188
11.4.6 IR DMA FIFO Mode 189
11.5 PWM Interrupt 190
11.6 PWM Center Align Mode 190
11.7 Register Description 191
12 SAR ADC 195
12.1 Power On/Down 195
12.2 ADC Clock 195
12.3 ADC Control in Auto Mode 195
12.3.1 Set Max State and Enable Channel 195
12.3.2 "Set" State 196
12.3.3 "Capture" State 196
12.3.4 Usage Case with Detailed Register Setting 197
12.4 Battery Voltage Sampling 198
12.5 Register Table 198
13 Temperature Sensor 203
14 Low Power Comparator 204
14.1 Power On/Down 204
14.2 Select Input Channel 204
14.3 Select Mode and Input Channel for Reference 205
14.4 Select Scaling Coefficient 205
14.5 Low Power Comparator Output 205
14.6 Register Description 205
15 Secure Boot, Firmware Encryption and Debug Lock 208
15.1 Key Management 208
15.2 Flash Space and eFuse Definition 209
15.2.1 Flash Space in Secure Boot Mode 209
15.2.2 eFuse Definition in Secure Boot Mode 210

15.2.3 Use of Debug Key 210
15.3 Usage 210
16 AES 212
17 Public Key Engine (PKE) 214
17.1 Calculation Model Overview 214
17.2 Function Description 214
17.2.1 Module Description 214
17.2.2 Software Interface (Programming Model) 215
17.3 Register Description 217
18 PTA Interface 222
18.1 BLE Two-Wire Signaling 222
18.2 BLE Three-Wire or Four-Wire Signaling 223
19 Quadrature Decoder 225
19.1 Input Pin Selection 225
19.2 Common Mode and Double Accuracy Mode 225
19.3 Read Real Time Counting Value 227
19.4 QDEC Reset 228
19.5 Other Configuration 228
19.6 Timing Sequence 229
19.7 Register Table 230
20 True Random Number Generator (TRNG) 231
20.1 Model Overview 231
20.2 Interrupt Description 231
20.2.1 CPU Reads RBG_DR without Data 231
20.2.2 Data Valid 232
20.3 Usage Procedure 232
20.3.1 Normal Operation 232
20.3.2 Entropy Source 232
20.4 Register Description 232

List of Figures

Figure 1-1 Block Diagram of the System 18
Figure 1-2 Package of TLSR9228H 24
Figure 1-3 Pin assignment of TLSR9228H 25
Figure 3-1 Reference Schematic of TLSR9228H 43
Figure 4-1 Memory Map 46
Figure 4-2 Control Logic of Power up/down 55
Figure 4-3 Initial Power-up Sequence 57
Figure 4-4 Initial Power-down Sequence 58
Figure 4-5 LDO and DCDC 60
Figure 4-6 Wake up Sources 61
Figure 5-1 Block Diagram of RF Transceiver 64
Figure 6-1 Clock Sources 67
Figure 7-1 Block Diagram of PLMT 81
Figure 8-1 Block Diagram of Interrupt 83
Figure 8-2 Block Diagram of PLIC 91
Figure 8-3 Detailed Block Diagram of PLIC 92
Figure 8-4 Interrupt Flow 96
Figure 8-5 Block Diagram of PLIC_SW 99
Figure 9-1 Example of DMA Data Transfers 101
Figure 9-2 Linked List Structure for Chain Transfers 102
Figure 10-1 Logic Relationship between GPIO and Related Modules 124
Figure 10-2 Detailed Circuit for Part A 125
Figure 10-3 Detailed Circuit for Part B 126
Figure 10-4 Detailed Circuit of Part C 127
Figure 10-5 One GPIO drives two LEDs 132
Figure 10-6 I2C Timing 133
Figure 10-7 Byte Consisted of Slave Address and R/W Flag Bit 134

Figure 10-8 Read Format in Slave Mode 134
Figure 10-9 Write Format in Slave Mode 134
Figure 10-10 Memory SPI Diagram 140
Figure 10-11 LSPI Diagram 146
Figure 10-12 GSPI Diagram 157
Figure 10-13 SPI_SLV Diagram 166
Figure 10-14 SPI_SLV Write Format 166
Figure 10-15 SPI_SLV Read Format 166
Figure 10-16 Block Diagram of UART 168
Figure 10-17 UART Protocol Formats and Sampling Point 170
Figure 10-18 Hardware Flow Control between 2 UARTs 171
Figure 10-19 Timeout Flag Used for Data Transmission 172
Figure 11-1 Signal Frame 185
Figure 11-2 PWM Output Waveform Chart 186
Figure 11-3 Continuous Mode 187
Figure 11-4 Counting Mode (n=0) 187
Figure 11-5 IR Mode (n=0) 188
Figure 11-6 IR Format Examples 189
Figure 11-7 Center Align Mode 190
Figure 12-1 Diagram of ADC 195
Figure 13-1 Block Diagram of Temperature Sensor 203
Figure 14-1 Block Diagram of Low Power Comparator 204
Figure 15-1 Key Management 208
Figure 15-2 Flash Space for Secure Boot 209
Figure 16-1 AES Address 212
Figure 17-1 Block Diagram of PKE Module 215
Figure 18-1 BLE Two-Wire Signaling 222
Figure 18-2 Example of BLE Two-Wire PTA Timing Diagram 222
Figure 18-3 BLE Three-Wire or Four-Wire Signaling 223

Figure 18-4 Example of BLE Four-Wire PTA Timing Diagram 224

Figure 19-1 Common Mode 226

Figure 19-2 Double Accuracy Mode 227

Figure 19-3 Read Real Time Counting Value 228

Figure 19-4 Shuttle Mode 228

Figure 19-5 Timing Sequence Chart 229

Figure 20-1 Module Structure 231

List of Tables

Table 1-1 Ordering Information of TLSR9228 23
Table 1-2 Mechanical Dimension of TLSR9228H 24
Table 1-3 Pin Function of TLSR9228H 26
Table 1-4 GPIO Pin Mux of TLSR9228H 27
Table 1-5 PWM Signal Description 29
Table 1-6 I2C Signal Description 30
Table 1-7 I2S Signal Description 30
Table 1-8 UART Signal Description 30
Table 1-9 GSPI Signal Description 30
Table 1-10 LSPI Signal Description 31
Table 1-11 SPI Slave Signal Description 31
Table 1-12 7816 Signal Description 31
Table 1-13 DMIC Signal Description 31
Table 1-14 Swire Signal Description 31
Table 1-15 External Power Amplifier, Low Noise Amplifier Signal Description 32
Table 1-16 USB Signal Description 32
Table 1-17 JTAG Signal Description 32
Table 1-18 SDP Signal Description 32
Table 1-19 PTA Signal Description 32
Table 1-20 ATSEL Signal Description 33
Table 1-21 Low Current Comparator Signal Description 33
Table 1-22 SAR ADC Signal Description 33
Table 1-23 Crystal Signal Description 34
Table 2-1 Absolute Maximum Rating 35
Table 2-2 Recommend Operating Conditions 35
Table 2-3 RX/TX Current and Sleep Current 36
Table 2-4 Digital Inputs/Outputs 36

Table 2-5 RF Performance Characteristics 37
Table 2-6 USB Characteristics 41
Table 2-7 RSSI Characteristics 41
Table 2-8 Crystal Characteristics 41
Table 2-9 RC Oscillator Characteristics 41
Table 2-10 ADC Characteristics 42
Table 3-1 BOM Table for TLSR9228H Reference Design 43
Table 4-1 eFuse Definition 47
Table 4-2 PMP Configuration Registers 49
Table 4-3 PMPiCFG Description 49
Table 4-4 PMP Address Registers 50
Table 4-5 D25 NAPOT Range Encoding in PMP Address and Configuration Registers 51
Table 4-6 Working Mode 51
Table 4-7 Retention Analog Registers in Deep Sleep 53
Table 4-8 Register Configuration for Software Reset 54
Table 4-9 Analog Register to Control Logic of Power Up/Down 56
Table 4-10 Characteristics of Initial Power-up/Power-down Sequence 58
Table 4-11 3.3 V Analog Register for Module Power up/down Control 59
Table 4-12 Analog Register for Wakeup 62
Table 5-1 Packet Format in Standard 1 Mbps BLE Mode 65
Table 5-2 Packet Format in Standard 2 Mbps BLE Mode 65
Table 5-3 Packet Format in Standard 500 kbps/125 kbps BLE Mode 65
Table 5-4 Packet Format in 802.15.4 Mode 65
Table 5-5 Packet Format in Proprietary Mode 66
Table 6-1 Clock Sources of Each Module 68
Table 6-2 Clock Related Registers 70
Table 7-1 Registers for Timer 0 ~ Timer 1 77
Table 7-2 32K LTimer Related Registers 79
Table 7-3 System Timer Related Registers 80

Table 7-4 PLMT Related Registers 82
Table 8-1 Interrupt Priority 84
Table 8-2 Register Description of mstatus 86
Table 8-3 Register Description of mie 86
Table 8-4 Register Description of mtvec 87
Table 8-5 Register Description of mepc 87
Table 8-6 Register Description of mcause 87
Table 8-7 Possible Values of mcause 87
Table 8-8 Possible values of mcause after reset 88
Table 8-9 Possible values of mcause after vector interrupt 88
Table 8-10 Register Description of mtval 89
Table 8-11 Register Description of mip 89
Table 8-12 Register Description of mdcause 89
Table 8-13 Detailed mdcause meaning in different mcause condition 90
Table 8-14 Register Description of mdcause 91
Table 8-15 Interrupt Sources 93
Table 8-16 Register Configuration for PLIC 97
Table 8-17 Register Configuration for PLIC_SW 100
Table 9-1 Format of Linked List Descriptor 103
Table 9-2 DMA Register Summary 103
Table 9-3 DMAC Control Register 104
Table 9-4 Interrupt Status Register 104
Table 9-5 Channel Abort Register 105
Table 9-6 Channel n Control Register 105
Table 9-7 Request/Ack Handshake Pair for Hardware Connection 108
Table 9-8 Channel n Source Address Register 109
Table 9-9 Channel n Destination Address Register 109
Table 9-10 Channel n Transfer Size Register 110
Table 9-11 Channel Linked List Pointer Register 110

Table 9-12 Baseband Related Registers 110
Table 9-13 Linked List Interrupt Mode 113
Table 9-14 Register Configuration for SRAM to SRAM 113
Table 9-15 Register Configuration for SRAM to Peripherals 114
Table 9-16 Register Configuration for Peripherals to SRAM 115
Table 10-1 GPIO Pad Function Mux 118
Table 10-2 GPIO Setting 119
Table 10-3 GPIO Function Mux Configuration Registers 121
Table 10-4 GPIO IRQ Table 128
Table 10-5 Analog Registers for Pull-up/Pull-down Resistor Control 130
Table 10-6 Swire Related Registers 132
Table 10-7 I2C Related Registers 135
Table 10-8 Memory SPI Related Registers 140
Table 10-9 LSPI Related Registers 146
Table 10-10 GSPI Related Registers 157
Table 10-11 SPI_SLV Commands 167
Table 10-12 Clock Variation Tolerance Factor 170
Table 10-13 UART Related Registers 172
Table 10-14 USB Related Registers 178
Table 11-1 PWM Registers 191
Table 12-1 Overall Register Setting 197
Table 12-2 SAR ADC Registers 198
Table 13-1 Analog Register for Temperature Sensor 203
Table 14-1 Analog Register Related to Low Power Comparator 205
Table 15-1 eFuse Data for Secure Boot 210
Table 16-1 AES Related Registers 212
Table 17-1 Dual Port RAM Address Map 216
Table 17-2 PKE Related Registers 217
Table 18-1 Register Configuration for t1/t2 224

Table 19-1 Input Pin Selection 225

Table 19-2 Timing..... 229

Table 19-3 Register Table for QDEC 230

Table 20-1 TRNG Related Register 233

1 Overview

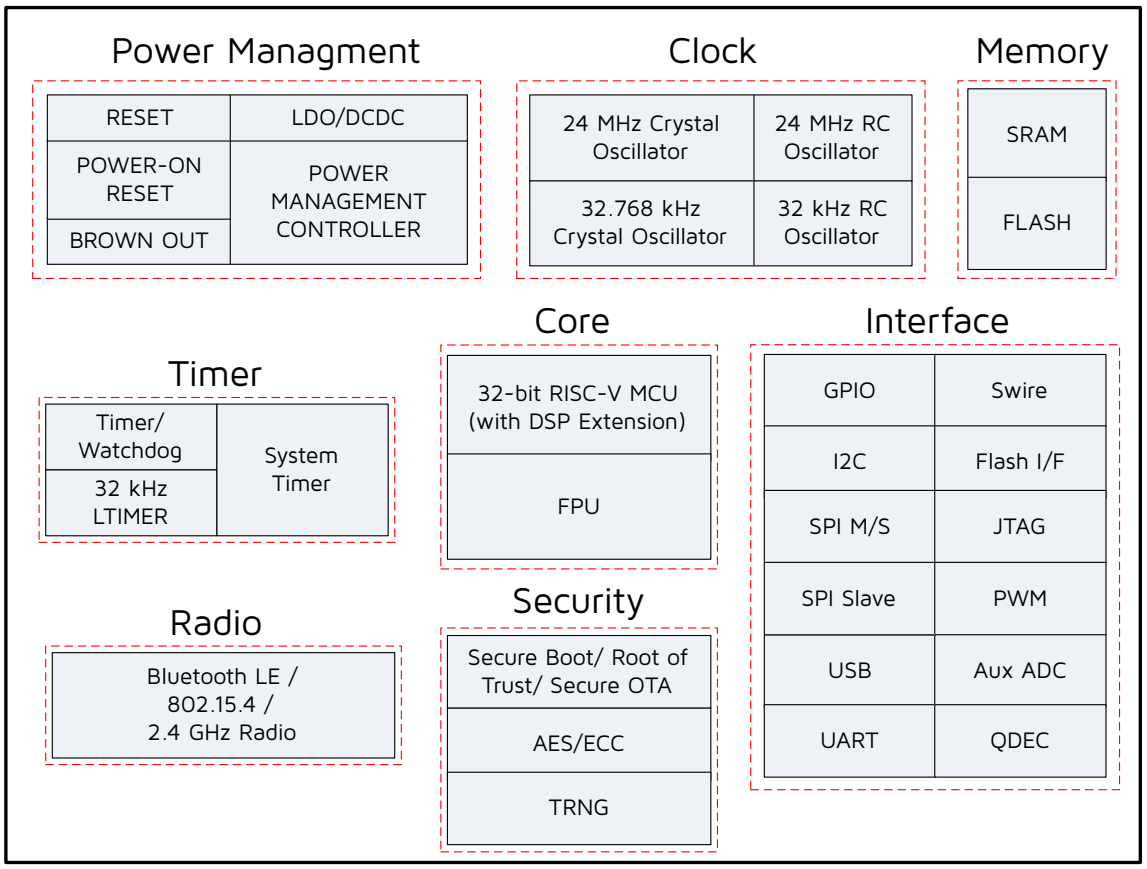
The TLSR9228 supports standards and industrial alliance specifications including Bluetooth low energy (LE), indoor positioning, Bluetooth LE Mesh, Zigbee 3.0, RF4CE, HomeKit, 6LoWPAN, Thread, Matter, 2.4 GHz proprietary, RFFE and Wi-Fi Coexistence Interface.

The TLSR9228 combines the features and functions needed for high quality wearable devices into a single System-on-Chip (SoC).

1.1 Block Diagram

The TLSR9228 is designed to offer high integration, ultra-low power application capabilities. The system's block diagram is as shown in Figure 1-1.

Figure 1-1 Block Diagram of the System



The TLSR9228 integrates a powerful 32-bit RISC-V (RISC-Five) MCU, 2.4 GHz ISM Radio, 512 KB SRAM, embedded 2 MB flash, Aux ADC, analog and digital microphone input, PWM, quadrature decoder (QDEC), flexible IO interfaces, and other peripheral blocks required for advanced wireless applications.

The TLSR9228 also includes multi-stage power management design allowing ultra-low power operation and making it the ideal candidate for wearable and power-constraint applications.

With the high integration level of TLSR9228, few external components are needed to satisfy customers' complicated application requirements.

1.2 Key Features

1.2.1 General Features

General features are as follows:

1. Supports 128-bit Unique Chip ID
2. Security solution
   - Root of Trust
   - Secure Boot
   - Secure OTA
   - Firmware encryption
   - Secure Debug Port Control
   - Prevent any unauthorized or maliciously modified software from running
   - Signature and verification based on RSA1024
   - 1024-bit eFuse
   - Embedded hardware AES block cipher with 128 bit keys and software AES-CCM, supports ECB/CCM/GCM/CTR modes
   - Hardware acceleration for elliptical curve cryptography (ECC), supports ECDH and ECDSA
   - Hardware true random number generator (TRNG)
3. RTC and other timers
   - Clock source of 24 MHz & 32.768 kHz Crystal and 32 kHz / 24 MHz 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)
   - A low-frequency 2 MHz ~ 4 MHz clock configurable on 24 MHz RC oscillator in low power mode
4. A rich set of digital and analog interfaces
   - Up to 30 GPIOs
   - Configurable to select 2-wire SDP or 4-wire JTAG debug interface
   - 4 different SPI channels
     - MSPI: Memory SPI dedicated for NOR flash
       - Up to 64 MHz SPI clock
       - Supports quad/dual/signal data line
     - GSPI: Global SPI
       - Up to 48 MHz SPI clock
       - Supports quad/dual/single data line
       - 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

  - Supports TFT panel interface
  - Supports TFT panel interface without internal RAM
- SPI Slave: SPI Slave for external accessing
- 2-channel I2C
- 2-channel UART with hardware flow control and 7816 protocol support
- USB 2.0 Full Speed
- Swire
- Up to 6 channels of differential PWM
- IR transmitter with DMA
- One quadrature decoder (QDEC), two-phase input selectable
- 10-channel (only GPIO input), 14-bit auxiliary ADC
- Low power comparator
- Internal 3-channel LED resistance
  - 2 channels are able to drive up to 8 mA
  - 1 channel is able to drive up to 4 mA
  - Shrink current resistance

5. Operating temperature range: -40°C ~ +125°C
6. Completely RoHS-compliant package
   - TLSR9228H, 48-pin QFN 6x6x0.75mm
7. Supports 2.4 GHz IoT standards into a single SoC, including Bluetooth, Bluetooth LE Mesh, Zigbee 3.0, Homekit, 6LoWPAN, Thread, Matter and 2.4 GHz proprietary technologies

1.2.2 CPU and Memory

1. 32-bit RISC-V (IMACFDP) micro-controller
   - Instruction and data cache controller (8K I-Cache and 8K D-Cache)
   - Maximum running speed up to 96 MHz
   - Integrated DSP extension instructions
   - Integrated "F" standard extensions for single-precision floating-point
2. Memory architecture
   - Program memory: 2MB flash
   - 512 KB SRAM including up to 96 KB retention SRAM
3. DSP features
   - SIMD Data Processing Instructions
   - Partial-SIMD Data Processing Instructions
   - 64-bit Profile Instructions
   - Non-SIMD Instructions
   - Overflow Status Manipulation Instructions

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 compliant, Bluetooth LE 1 Mbps and 2 Mbps, Long Range 125 kbps and 500 kbps
3. High accuracy ranging and distance measurement with channel sounding
4. IEEE 802.15.4 compliant, 250 kbps
5. 2.4 GHz proprietary 1 Mbps/2 Mbps/250 kbps/500 kbps mode with Adaptive Frequency Hopping feature
6. RX sensitivity: -96 dBm @ Bluetooth LE 1 Mbps mode, -93.5 dBm @ Bluetooth LE 2 Mbps mode, -100.5 dBm @ Long Range 125 kbps mode, -98.5 dBm @ Long Range 500 kbps mode, -99.5 dBm @ IEEE 802.15.4 250 kbps mode
7. TX output power: -24 to +10 dBm @ Bluetooth LE mode
8. 50 Ω matched single-pin antenna input
9. RSSI monitoring with +/-1 dB resolution and ranging from -100 dBm to 0 dBm
10. Auto acknowledgment, retransmission and flow control
11. Supports full-function Bluetooth LE AoA/AoD location features
12. Supports PTA (Packet Traffic Arbitrator) for Wi-Fi co-existence

1.2.4 Features of Power Management Module

Features of power management module include:

1. Power supply
   - VBAT (battery): 1.9 V ~ 4.3 V
   - VBUS (USB): 4.5 V ~ 5.5 V
2. Battery monitor for low battery voltage detection
3. Brownout detection/shutdown and Power-On-Reset
4. Multiple-power-state to optimize power consumption
5. Low power consumption:
   - Whole chip, RX BLE mode: 5.0 mA @ 3.3 V DCDC
   - Whole chip, TX BLE mode: 6.0 mA @ 0 dBm, 3.3 V DCDC
   - Deep sleep with external wakeup (without SRAM retention): 0.7 μA
   - Deep sleep with SRAM retention: 1.7 μA (with 32 KB SRAM retention), 2.7 μA (with 64 KB SRAM retention), 4.7 μA (with 96 KB SRAM retention)
   - Deep sleep with external wakeup (without SRAM retention, with 32K RC): 1.1 μA
   - Deep sleep with SRAM retention: 2.1 μA (with 32 KB SRAM retention, with 32K RC), 3.1 μA (with 64 KB SRAM retention, with 32K RC), 5.1 μA (with 96 KB SRAM retention, with 32K RC)
6. Supports VBAT communication: Data transfer through VBAT supply modulation for easy communication

1.2.5 Bluetooth Features

Bluetooth features include:

1. Qualified Bluetooth LE 5.3, 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 (a.k.a Bluetooth 5.3)
4. Bluetooth ISO channel support (a.k.a Bluetooth 5.3) with broadcast and connected mode

1.2.6 Zigbee Features

Zigbee features include:

1. Zigbee 3.0 platform with Zigbee 3.0 device support, compatible with ZHA and ZLL
2. Green Power support for low power devices
3. Supports High Data Rate Modes (500/1000/2000 kbps) for applications beyond IEEE 802.15.4 compliant networks
4. Supports Automatic Rate Recognition feature based on SFD field

1.2.7 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

1.2.8 HomeKit Features

HomeKit features include:

1. Single-chip solution with hardware acceleration for all HomeKit security operations
2. Supports Apple HomeKit ADK over Bluetooth LE and Thread
3. Conformant to latest HomeKit specification
4. Tested against Apple HomeKit Accessory Tester and Apple iOS HomeKit HOME application
5. Supports all HAP defined services and characteristics
6. Supports custom defined HAP services and characteristics
7. HomeKit custom update over-the-air (OTA) profile for secure software upgrade over the air implemented

1.2.9 Matter Features

Matter features include:

1. Supports Matter over Thread
2. Supports Multi-Admin works across & with multiple ecosystems
3. Supports rich applications
   - Smart home devices
   - Mobile apps

- Cloud services

1.2.10 Concurrent Mode Feature

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

Typical combination is Bluetooth LE + 802.15.4 based standard (e.g. Zigbee, Thread, or 6LoWPAN): Bluetooth LE and 802.15.4 based stacks can run concurrently with one application state based on time division technology, e.g. Bluetooth LE stack and Thread stack will run alternately during the divided time slots.

1.3 Typical Applications

The TLSR9228 is an ideal SoC for advanced wireless solutions and Internet of Things (IoT) applications. Its typical applications include, but are not limited to the following:

- IoT applications
  - Smart lighting, smart home devices
  - RF and IR remote control
  - Smart grid
  - Intelligent logistics/transportation/city
  - Health care
  - Wearable devices
  - Matter application

1.4 Ordering Information

Table 1-1 Ordering Information of TLSR9228

Product Series	Ordering No.	Package Type	SRAM Size	Flash Size	Temperature Range	Packing Method	Minimum Order Quantity
TLSR9228	TLSR9228HAR	QFN48 6x6x0.75mm	512KB	2MB	-40°C~+125°C	TR[a]	3000

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

1.5 Package

Package dimensions of TLSR9228H are shown below.

Figure 1-2 Package of TLSR9228H

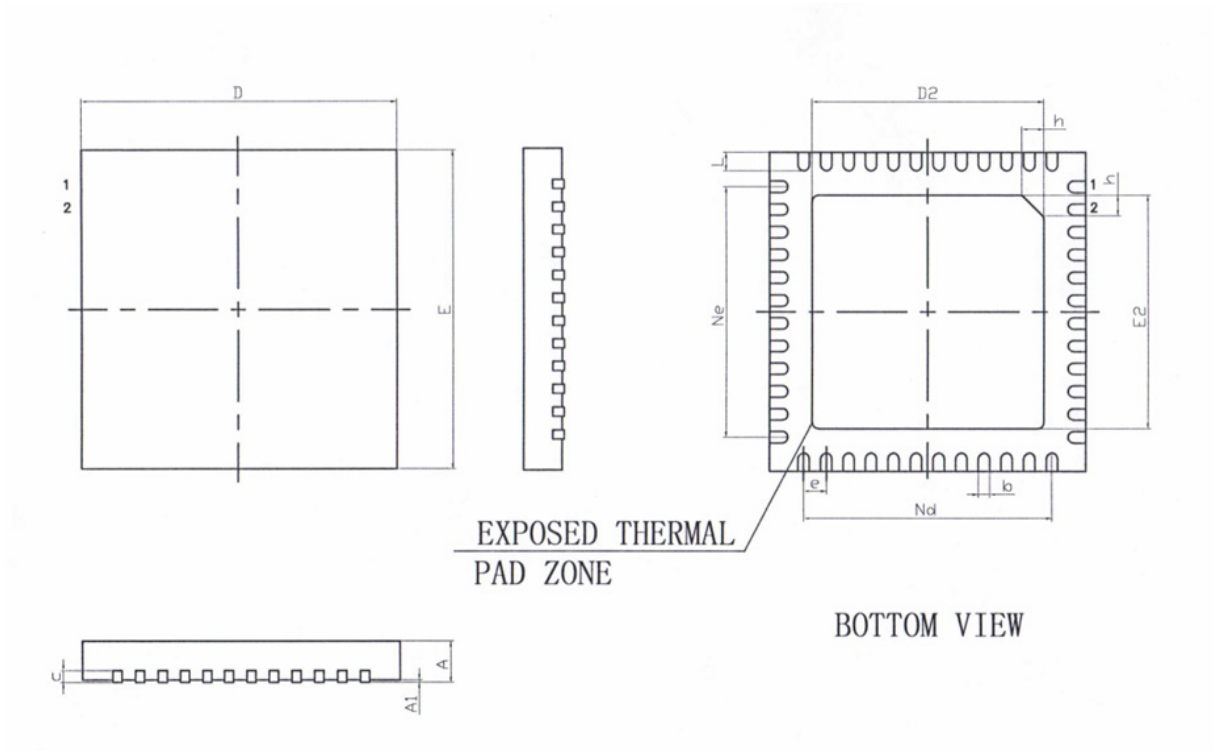


Table 1-2 Mechanical Dimension of TLSR9228H

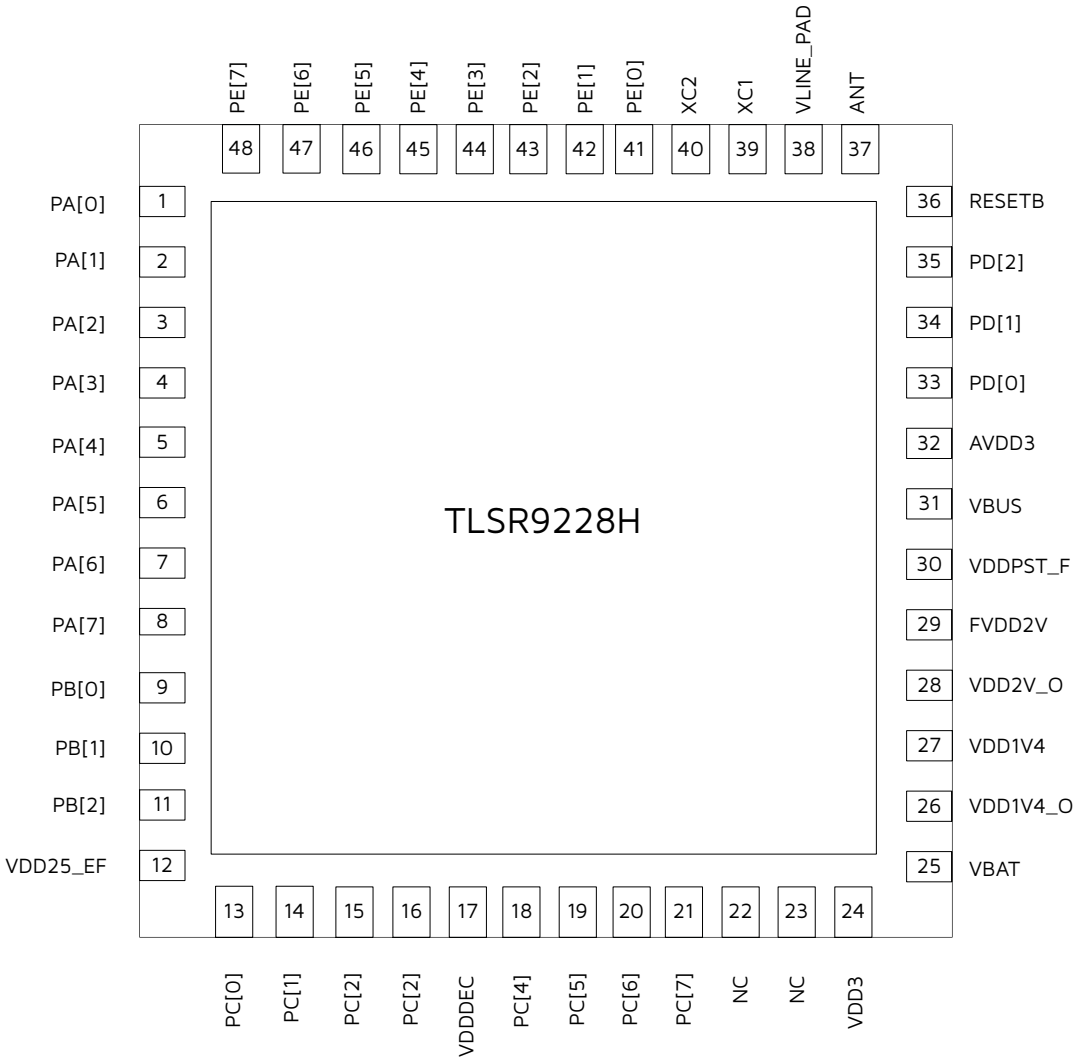
SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	0.70	0.75	0.80
A1	-	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.23
D	5.90	6.00	6.10
D2	4.10	4.20	4.30
e	0.40 BSC		
Ne	4.40 BSC		
Nd	4.40 BSC		
E	5.90	6.00	6.10
E2	4.10	4.20	4.30
L	0.35	0.40	0.45

SYMBOL	MILLIMETER		
	MIN	NOM	MAX
h	0.30	0.35	0.40
L/F carrier (mil)	177x177		

1.6 Pin Layout

Pin assignment of TLSR9228H is shown below.

Figure 1-3 Pin assignment of TLSR9228H



Functions of 48 pins of TLSR9228H are described in table below.

Table 1-3 Pin Function of TLSR9228H

No.	Pin Name	Type	Description
1	PA[0]	GPIO	GPIO PA[0]
2	PA[1]	GPIO	GPIO PA[1]
3	PA[2]	GPIO	GPIO PA[2]
4	PA[3]	GPIO	GPIO PA[3]
5	PA[4]	GPIO	GPIO PA[4]
6	PA[5]	GPIO	GPIO PA[5]
7	PA[6]	GPIO	GPIO PA[6]
8	PA[7]	GPIO	GPIO PA[7]
9	PB[0]	GPIO	GPIO PB[0]
10	PB[1]	GPIO	GPIO PB[1]
11	PB[2]	GPIO	GPIO PB[2]
12	VDD25_EF	PWR	eFuse power supply
13	PC[0]	GPIO	GPIO PC[0]
14	PC[1]	GPIO	GPIO PC[1]
15	PC[2]	GPIO	GPIO PC[2]
16	PC[3]	GPIO	GPIO PC[3]
17	VDDDEC	PWR	Digital power supply
18	PC[4]	GPIO	GPIO PC[4]
19	PC[5]	GPIO	GPIO PC[5]
20	PC[6]	GPIO	GPIO PC[6]
21	PC[7]	GPIO	GPIO PC[7]
22	NC	NC	Not connected
23	NC	NC	Not connected
24	VDD3	PWR	3.3V power supply
25	VBAT	PWR	Lion-Battery power supply
26	VDD1V4_O	PWR	1.4V LDO/DCDC output
27	VDD1V4	PWR	Power supply for internal digital LDO and analog LDO

No.	Pin Name	Type	Description
28	VDD2V_O	PWR	2V LDO/DCDC output
29	FVDD2V	PWR	Flash power supply
30	VDDPST_F	PWR	Integrated flash LDO 1.8V output
31	VBUS	PWR	5V USB power supply
32	AVDD3	PWR	3.3V analog power supply
33	PD[0]	GPIO	GPIO PD[0]
34	PD[1]	GPIO	GPIO PD[1]
35	PD[2]	GPIO	GPIO PD[2]
36	RESETB	Reset	Reset pin, active low
37	ANT	Analog	Pin to connect to the antenna through the matching network
38	VLINE_PAD	PWR	1.4V RF power supply
39	XC1	Analog	Crystal oscillator pin 1
40	XC2	Analog	Crystal oscillator pin 2
41	PE[0]	GPIO	GPIO PE[0]
42	PE[1]	GPIO	GPIO PE[1]
43	PE[2]	GPIO	GPIO PE[2]
44	PE[3]	GPIO	GPIO PE[3]
45	PE[4]	GPIO	GPIO PE[4]
46	PE[5]	GPIO	GPIO PE[5]
47	PE[6]	GPIO	GPIO PE[6]
48	PE[7]	GPIO	GPIO PE[7]

GPIO pin mux functions of TLSR9228H are shown in the table below.

Table 1-4 GPIO Pin Mux of TLSR9228H

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	-

Pad	Default	Function1	Function2	Analog Function
PA[4]	GPIO	All functions	SWM	-
PA[5]	GPIO	-	DM	-
PA[6]	GPIO	-	DP	-
PA[7]	SWS	-	SWS	-
PB[0]	GPIO	All functions	SWM	lp_comp<0>/sar_in<0>
PB[1]	GPIO	All functions	SWM	lp_comp<1>/sar_in<1>
PB[2]	GPIO	All functions	SWM	lp_comp<2>/sar_in<2>
PC[0]	SSPI_CN	All functions	SSPI_CN	-
PC[1]	SSPI_CK	All functions	SSPI_CK	-
PC[2]	SSPI_SI	All functions	SSPI_SI	-
PC[3]	SSPI_SO	All functions	SSPI_SO	-
PC[4]	TDI	All functions	TDI	-
PC[5]	GPIO	All functions	SWM	-
PC[6]	GPIO	All functions	SWM	-
PC[7]	GPIO	All functions	SWM	-
PD[0]	GPIO	All functions	SWM	xtl32k_in/sar_in<8>
PD[1]	GPIO	All functions	SWM	xtl32k_out/sar_in<9>
PD[2]	GPIO	All functions	SWM	cdc_ldo_vo_test
PE[0]	GPIO	-	LSPI_CN	-
PE[1]	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	-

a. "All functions" include 60 functions: GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL_IO, I2C1_SDA_IO, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_INBAND, BT_STATUS, BT_ACTIVITY, WIFI_DENY_I, TX_CYC2PA, DMIC1_DAT, DMIC1_CLK, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT_IN, I2S1_LR_IN, I2S1_DAT_OUT, I2S1_LR_OUT, I2S1_BCK, I2S0_CLK, I2S0_DAT_IN I2S0_LR_IN, I2S0_DAT_OUT, I2S0_LR_OUT, 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, and PWM0.

NOTE:

- The default function of GPIO pins PC[5], PG[1], PG[2], PG[3] and PG[5] is outputting voltage, however, these pins cannot maintain high level or low level after deep sleep or deep retention sleep although they have pull-up/down resistor, therefore these pins are not suitable to be used for the applications that require stable voltage such as wakeup source.
- The GPIO pins PG[0...5] for MSPI interface are not suitable to be used as wakeup source.
- When the IO voltage is configured to 1.8V and the chip is powered by the 5V VBUS, the default output GPIO PC[5] will have a voltage of 3.3V for a period of time after power-on or reset. It is important to ensure that the subsequent module can tolerate a 3.3V input.

Descriptions of each signal are listed below:

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

Table 1-5 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

Table 1-6 I2C Signal Description

Signal	Type	Description
I2C_SCL	DIO	I2C SCL
I2C_SDA	DIO	I2C SDA

Table 1-7 I2S Signal Description

Signal	Type	Description
I2S_BCK	DIO	I2S bit CLK
I2S_CLK	DO	I2S base CLK
I2S_LR_IN	DIO	I2S left and right channel SEL
I2S_LR_OUT	DIO	I2S left and right channel SEL
I2S_DAT_IN	DI	I2S data IN
I2S_DAT_OUT	DO	I2S data OUT

Table 1-8 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 1-9 GSPI Signal Description

Signal	Type	Description
GSPI_CK	DIO	GSPI CLK
GSPI_CN	DIO	GSPI CN
GSPI_MISO	DIO	GSPI MISO
GSPI_MOSI	DIO	GSPI MOSI
GSPI_IO2	DIO	GSPI IO2
GSPI_IO3	DIO	GSPI IO3

Table 1-10 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 1-11 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 1-12 7816 Signal Description

Signal	Type	Description
CLK_7816	DO	7816 CLK

Table 1-13 DMIC Signal Description

Signal	Type	Description
DMIC_CLK	DO	DMIC CLK
DMIC_DAT	DI	DMIC DATA IN

Table 1-14 Swire Signal Description

Signal	Type	Description
SWM	DIO	Swire Master
SWS	DIO	Swire Slave

Table 1-15 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 1-16 USB Signal Description

Signal	Type	Description
DP	DIO	USB DP
DM	DIO	USB DM

Table 1-17 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 1-18 SDP Signal Description

Signal	Type	Description
TCK	DI	Test clock input
TMS	DIO	Test mode selection

Table 1-19 PTA Signal Description

Signal	Type	Description
BT_ACTIVITY	DIO	Bluetooth activity
BT_STATUS	DIO	Bluetooth status
BT_INBAND	DIO	Bluetooth inband
WIFI_DENY	DI	WIFI deny

Table 1-20 ATSEL Signal Description

Signal	Type	Description
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 1-21 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 1-22 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
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 1-23 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

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	-
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, such as VDDO3. Generally, the range of the IO voltage is 1.8 V ~ 3.6 V, and the typical value is 3.3 V.

2.2 Recommended Operating Conditions

Table 2-2 Recommend Operating Conditions

Item	Sym.	Min.	Typ.	Max.	Unit	Condition
Power supply voltage	VBAT	1.9	3.7	4.3	V	-
USB supply voltage	VBUS	4.5	5.0	5.5	V	-
Supply rise time (from 1.6 V to 1.8 V)	T_R	-	-	10	ms	-
Operating temperature range	T_{Opr}	-40	-	125	°C	-

2.3 DC Characteristics

Unless otherwise stated, the general test conditions are: VDD = 3.3 V, T = 25°C.

Table 2-3 RX/TX Current and Sleep Current

Item	Sym.	Min	Typ.	Max	Unit	Conditions
RX current	I_{Rx}	-	5.0	-	mA	Whole chip, BLE mode
TX current	I_{Tx}	-	6.0	-	mA	Whole chip @ 0 dBm with 3.3 V DCDC, BLE mode
Deep sleep with 32 KB SRAM retention	I_{Deep1}	-	1.7	-	μA	Without 32K RC[a]
Deep sleep with 64 KB SRAM retention		-	2.7	-	μA	
Deep sleep with 96 KB SRAM retention		-	4.7	-	μA	
Deep sleep without SRAM retention	I_{Deep2}	-	0.7	-	μA	
Deep sleep with 32 KB SRAM retention	I_{Deep3}	-	2.1	-	μA	With 32K RC[b]
Deep sleep with 64 KB SRAM retention		-	3.1	-	μA	
Deep sleep with 96 KB SRAM retention		-	5.1	-	μA	
Deep sleep without SRAM retention	I_{Deep4}	-	1.1	-	μA	

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

b. 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 will be updated according to actual test results.*

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		-	-	7	-	dB	Wanted signal at -67 dBm
In-band blocking rejection (equal modulation interference)	+1/-1 MHz offset	-	-	-3/-2	-	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	-

Item		Sym.	Min.	Typ.	Max.	Unit	Conditions
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	-	-	-93.5	-	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	-	-	-7/-7	-	dB	Wanted signal at -67 dBm
	+4/-4 MHz offset	-	-	-31/-31	-	dB	
	> 6 MHz offset	-	-	-35	-	dB	
Image rejection		-	-	-25	-	dB	Wanted signal at -67 dBm; image frequency = RF_channel - 3 MHz
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	-	-	-100.5	-	dBm	-
Frequency offset tolerance		-	-200	-	+200	kHz	-

Item		Sym.	Min.	Typ.	Max.	Unit	Conditions
Co-channel rejection		-	-	5	-	dB	Wanted signal at -79 dBm
In-band blocking rejection (equal modulation interference)	+1/-1 MHz offset	-	-	-6/-2	-	dB	Wanted signal at -79 dBm
	+2/-2 MHz offset	-	-	-42/-27	-	dB	
	≥ 3 MHz offset	-	-	-42/-36	-	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	-
Bluetooth LE 500 kbps RF_RX Performance (±250 kHz Deviation)							
Sensitivity	500 kbps	-	-	-98.5	-	dBm	-
Frequency offset tolerance		-	-200	-	+200	kHz	-
Co-channel rejection		-	-	10	-	dB	Wanted signal at -72 dBm
In-band blocking rejection (equal modulation interference)	+1/-1 MHz offset	-	-	1/2	-	dB	Wanted signal at -72 dBm
	+2/-2 MHz offset	-	-	-42/-41	-	dB	
	≥ 3 MHz offset	-	-	-50	-	dB	

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
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	-	-	-99.5	-	dBm	-
Frequency offset tolerance	-	-300	-	+300	kHz	-
Adjacent channel rejection (-1/+1 channel)	-	-	-42/-42	-	dB	Wanted signal at -82 dBm
Adjacent channel rejection (-2/+2 channel)	-	-	-42/-42	-	dB	Wanted signal at -82 dBm
Zigbee/RF4CE/6LoWPan/Thread RF_TX Performance (IEEE 802.15.4 250 kbps)						
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}	-20	-	+20	ppm	-
Load capacitance	C_L	5	12	18	pF	Programmable on chip load cap
Equivalent series resistance	ESR	-	50	100	Ohm	-
32.768 kHz Crystal						
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	Programmable on chip load cap
Equivalent series resistance	ESR	-	50	80	kOhm	-

Table 2-9 RC Oscillator Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
24 MHz RC Oscillator						

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
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}	-	0.03	-	%	On chip calibration
Calibration time	-	-	3	-	ms	-

Table 2-10 ADC Characteristics

Item	Sym.	Min.	Typ.	Max.	Unit	Conditions
Differential nonlinearity	DNL	-	-	1	LSB	10-bit resolution mode
Integral nonlinearity	INL	-	-	2	LSB	10-bit resolution mode
Signal-to-noise and distortion ratio	SINAD	-	70	-	dB	f_{IN} = 1 kHz, f_S= 16 kHz
Effective number of bits	ENOB	-	10.5	-	bits	-
Sampling frequency	Fs	-	-	200	ksps	-

2.5 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 will be 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

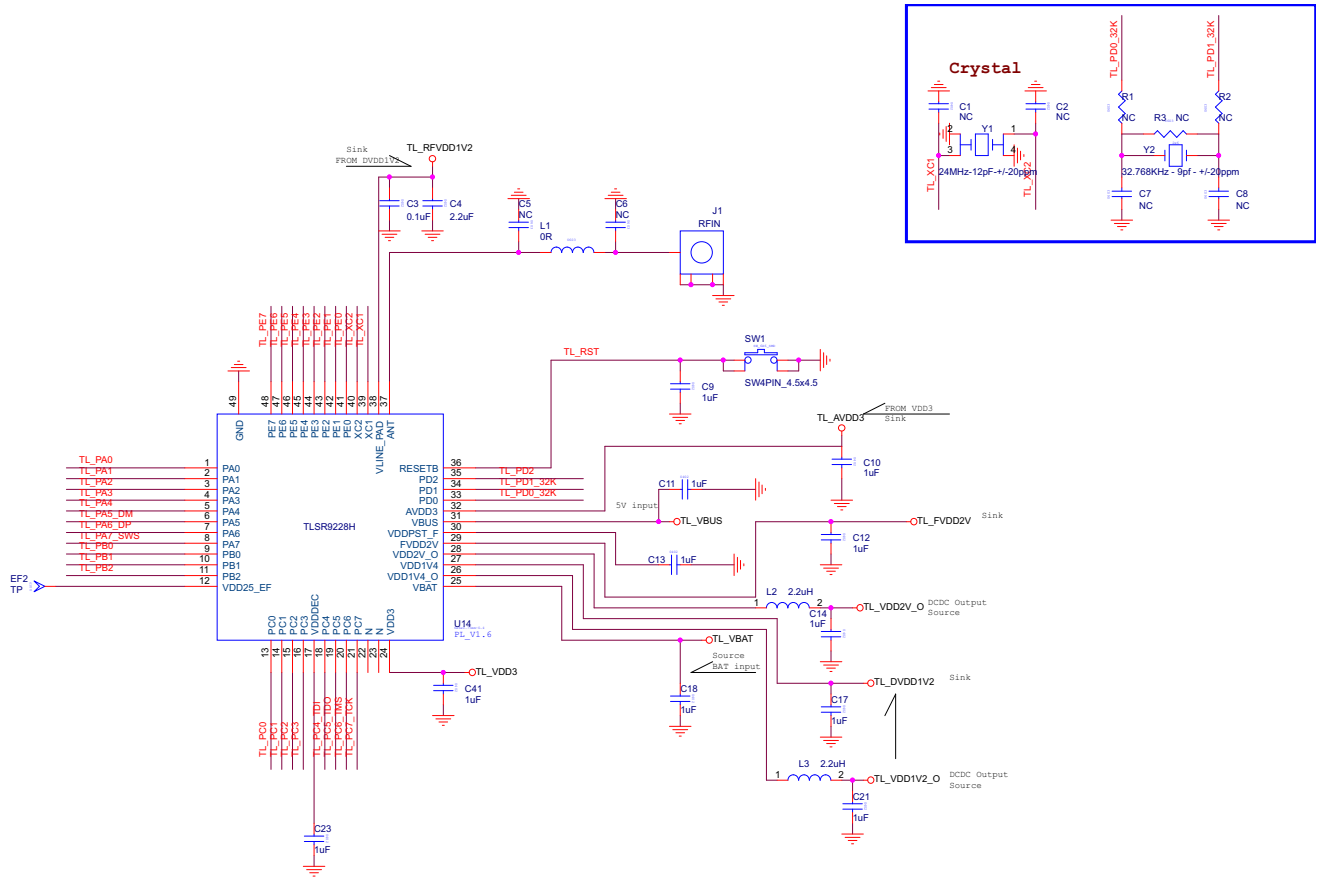
Note: If device containers cannot be subjected to high temperature or shorter bake times are desired, please refer to IPC/JEDEC J-STD-033 for bake condition.

3 Reference Design

3.1 Reference Schematic for TLSR9228H

The reference schematic of TLSR9228H is shown as below.

Figure 3-1 Reference Schematic of TLSR9228H



3.2 BOM (Bill of Material) for TLSR9228H

The bill of material table for TLSR9228H reference design is listed as below.

Table 3-1 BOM Table for TLSR9228H Reference Design

Quantity	Reference	Value	Description
1	C3	0.1uF	Capacitance,X5R, ±10%
1	C4	2.2uF	Capacitance,X5R, ±10%
11	C9,C10,C11,C12,C13,C14,C17,C18,C21, C23,C41	1uF	Capacitance,X5R, ±10%

Quantity	Reference	Value	Description
1	J1	RFIN	Connector for RF input
1	L1	0R	Resistance,SMD,±5%
2	L2,L3	2.2uH	High frequency chip inductor,SMD,20%
1	SW1	SW4PIN_4.5x4.5	Button
1	U14	TLSR9228H	BLE SoC
1	Y1	24MHz-12pF-+/-20ppm	XTAL SMD 3225, 24 MHz, CI=12pF, total tol.±20ppm
1	Y2	32.768KHz - 9pf - +/-20ppm	XTAL RADIAL 2x6mm, 32.768KHz, CI=9pF, total tol.±20ppm

4 Memory, MCU and PMU

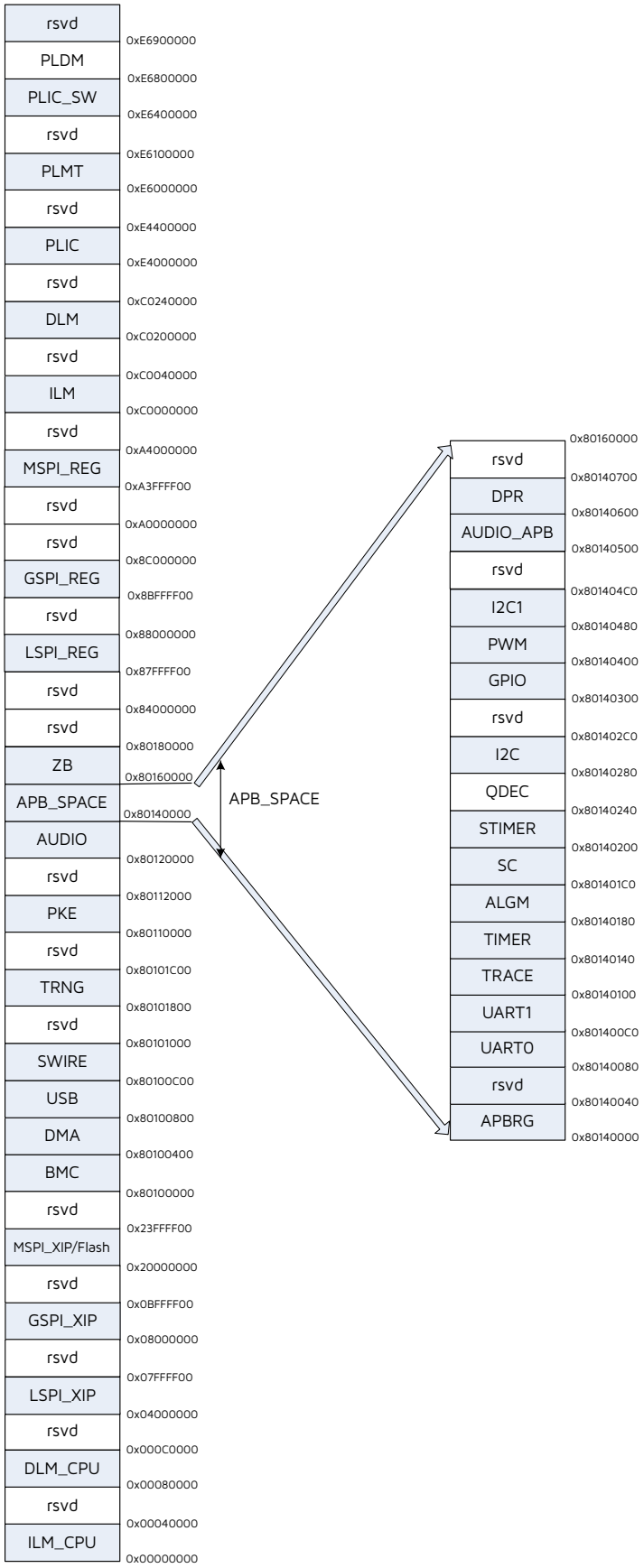
4.1 Memory

The SoC embeds 256 KB SRAM (including 96 KB with retention in deep sleep) as instruction memory, 256 KB SRAM as data memory, and external flash (depending on detailed 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 ILM, the lower 96 KB is with retention in deep sleep. ILM and DLM have different addressing address for MCPU (shown as ILM_CPU and DLM_CPU). 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 no more 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 the flash area ranging from 0x1FE000 to 0x1FFFFF is reserved for Telink internal use.

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 eFuse

The non-volatile eFuse is defined as below.

Table 4-1 eFuse Definition

Name	Default	Description
MAC_ADDR	64'd0	64 bits Telink-Assigned MAC Address (including 48 bits MAC Address for Bluetooth or 64 bits MAC Address for 802.15.4)
Reserved	-	Reserved for Telink internal use
Security_Feature	32'd0	[31] Flash Encryption Enable [30] Reserved [29] mode selection, 1: secure boot mode (verified signature), 0: normal mode [28:0] Reserved for Telink internal use
pub_key_hash	256'd0	Hash public key, unmodifiable
debug_text	128'd0	Plaintext for debugging
chip_id	128'd0	128-bit chip ID, unmodifiable
Reserved	128'd0	Reserved
root_key	128'd0	Root key
Interface functions	32'd0	[31:16] Reserved [15] if_sws_disable: SWS function disable. 1'b1: disable; 1'b0: enable. [14] if_jtag_disable: JTAG function disable. 1'b1: disable; 1'b0: enable. [13:0] Reserved

4.1.4 Unique ID

For chip identification and traceability, the SoC 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
   - 32-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 (hardware thread). 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	-	-

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

Table 4-4 PMP Address Registers

CSR Address	CSR Name	Bit	Description	Type	Reset
0x3B0	pmpaddr0	[31:0]	PMP entry 0 address register	R/W	0
0x3B1	pmpaddr1	[31:0]	PMP entry 1 address register	R/W	0
0x3B2	pmpaddr2	[31:0]	PMP entry 2 address register	R/W	0
0x3B3	pmpaddr3	[31:0]	PMP entry 3 address register	R/W	0
0x3B4	pmpaddr4	[31:0]	PMP entry 4 address register	R/W	0
0x3B5	pmpaddr5	[31:0]	PMP entry 5 address register	R/W	0
0x3B6	pmpaddr6	[31:0]	PMP entry 6 address register	R/W	0
0x3B7	pmpaddr7	[31:0]	PMP entry 7 address register	R/W	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}
a011…1111	2^{33}
0111…1111	2^{34}
1111…1111	2^{35}

4.3 Working Mode

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

- 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	Shutdown
MCU	active	stall	stall	off	off	off
Radio	available	available	off	off	off	off
USB	available	available/off	stall/off	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	10 ms

Mode	Active	Idle	Suspend	Deep Sleep With SRAM Retention	Deep Sleep without SRAM Retention	Shutdown
Wakeup Time to Active Mode in DCDC Mode	-	-	-	-	-	-
Retention SRAMs (with retention in deep sleep)	full	full	full	full	off	off
Wakeup on RTC (32K Timer wakeup)	-	-	available	available	available	off
Wakeup on pin (IO wakeup)	-	-	available	available	available	off
Wakeup on interrupt	-	available	-	-	-	-
Wakeup on reset pin (RESETB)	-	available	available	available	available	on

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/on for wakeup: Corresponding wakeup method is supported.
- off for wakeup: Corresponding wakeup method is not supported.
- full/off for SRAMs:
  - full: Full speed. In Active, Idle and Suspend mode, the three 32KB retention SRAMs are powered on and work normally (can be accessed); in Deep sleep with SRAM retention, 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 and Shutdown 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 will be cleared by watchdog reset or chip software reset.
- After POR (Power-On-Reset), all registers will be 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 watchdog	11111111
afe_0x36	R/W	buffer clean at watchdog	00000000
afe_0x37	R/W	buffer clean at watchdog	00000000
afe_0x38	R/W	buffer clean at watchdog	00000000
afe_0x39	R/W	buffer clean at watchdog	00000000
afe_0x3a	R/W	buffer clean at power on	00000000
afe_0x3b	R/W	buffer clean at power on	00000000
afe_0x3c	R/W	buffer clean at power on	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 will be reset, and all registers will be 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 will be 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 1'b1 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 will be cleared by chip software reset.
   - Addresses 0x20~0x23 serve to reset individual modules: if some bit is set to logic "1", the corresponding module is reset.

The base address of the following reset related registers is 0x801401c0.

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	0xa0
0x21	RST1	R/W	[0]: rsvd [1]: System Timer [2]: DMA [3]: ALGM [4]: PKE [5]: rsvd [6]: GSPI, apb spi [7]: SPISLV, spi slave	0x80
0x22	RST2	R/W	[0]: Timer [1]: rsvd [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

Address Offset	Name	Type	Description	Reset Value
0x23	RST3	R/W	[0]: ZB [1]: ZB_MSTCLK [2]: ZB_LPCLK [3]: ZB_CRYPT [4]: MSPI [5]: QDEC [6]: SARADC [7]: ALG, analog module reset	0x90
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

4.5 Power Management

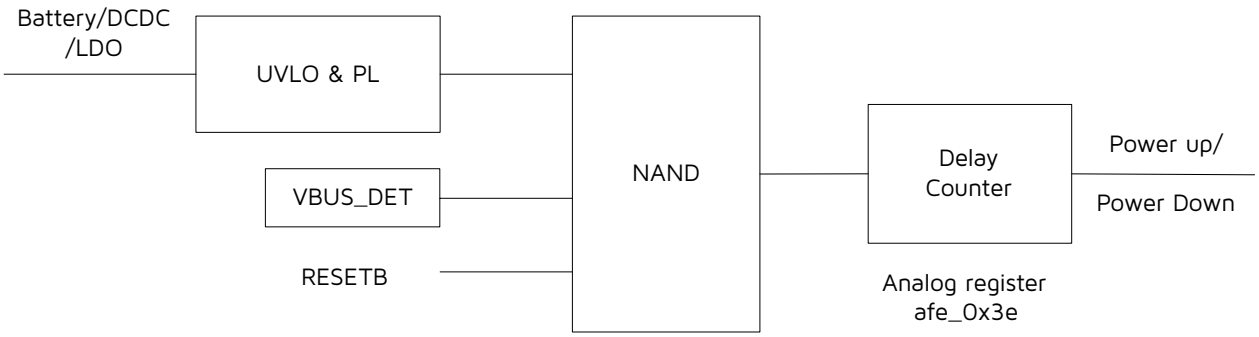
The multiple-stage Power Management (PM) module is flexible to control power state of the whole chip or individual functional blocks such as MCU, RF Transceiver, and peripherals by the following methods:

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 figure above, the whole power up and down is controlled by the UVLO (Ultra-low Voltage Lockout) & PL (Power Logic) module, VBUS detector and the external RESETB pin via the logic shown in the above diagram. UVLO takes the external power supply as input and releases the lock only when the power supply voltage is higher than a preset threshold. When there is USB plugged in, the VBUS_DET detects that VBUS has power and may generate reset after 5ms, however, it can writer pm top register 0x69 bit [6] to 1 to disable this reset generation. The RESETB pin has an internal pull-up resistor; an external capacitor can be connected on the RESETB pin to control the POR delay.

After UVLO, VBUS_DET and RESETB release, there is a further configurable delay before the system reset signal ("Sysrst") is released. The delay is adjusted by analog register afe_0x3e. Since the content of afe_0x3e is reset to default only after power cycle, watchdog reset, or software reset, the delay change using afe_0x3e is only applicable when the chip has not gone through these reset conditions. For example, after deep sleep wakeup, the setting in afe_0x3e will take effect.

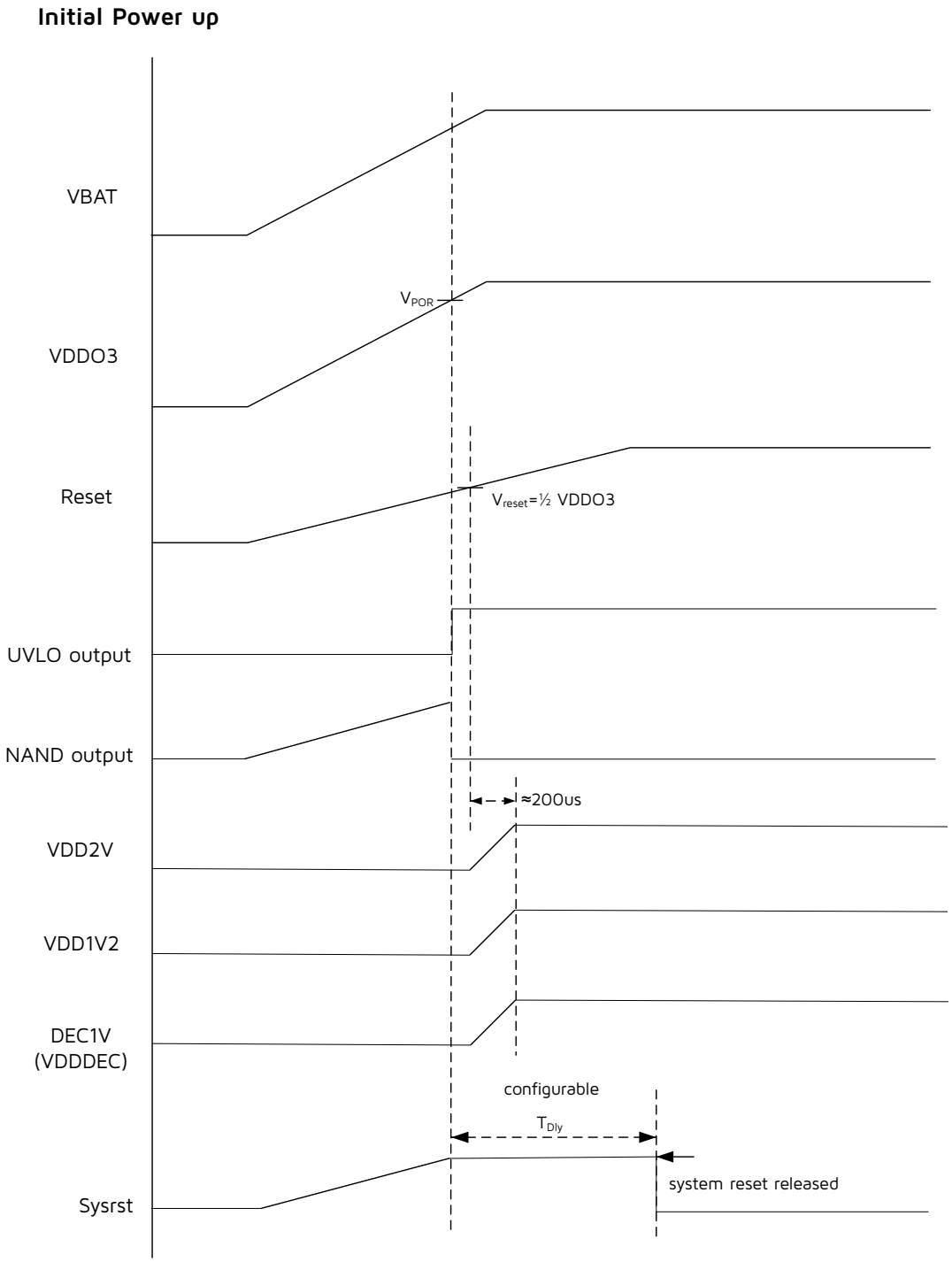
The related analog registers are described in table below.

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

Address	Name	Type	Description	Default Value
afe_0x3e	r_dly	R/W	base on 16KHz frequency increase counter (8ms)	10000000
afe_0x69	pg status	R	[6] vbus_detect (write 1 to disable vbus rst timer)	-

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

Figure 4-3 Initial Power-up Sequence



NOTE:

- *The VDDO3 actually has no delay relative to VBAT, as the internal circuit is powered by VBAT.*

Figure 4-4 Initial Power-down Sequence

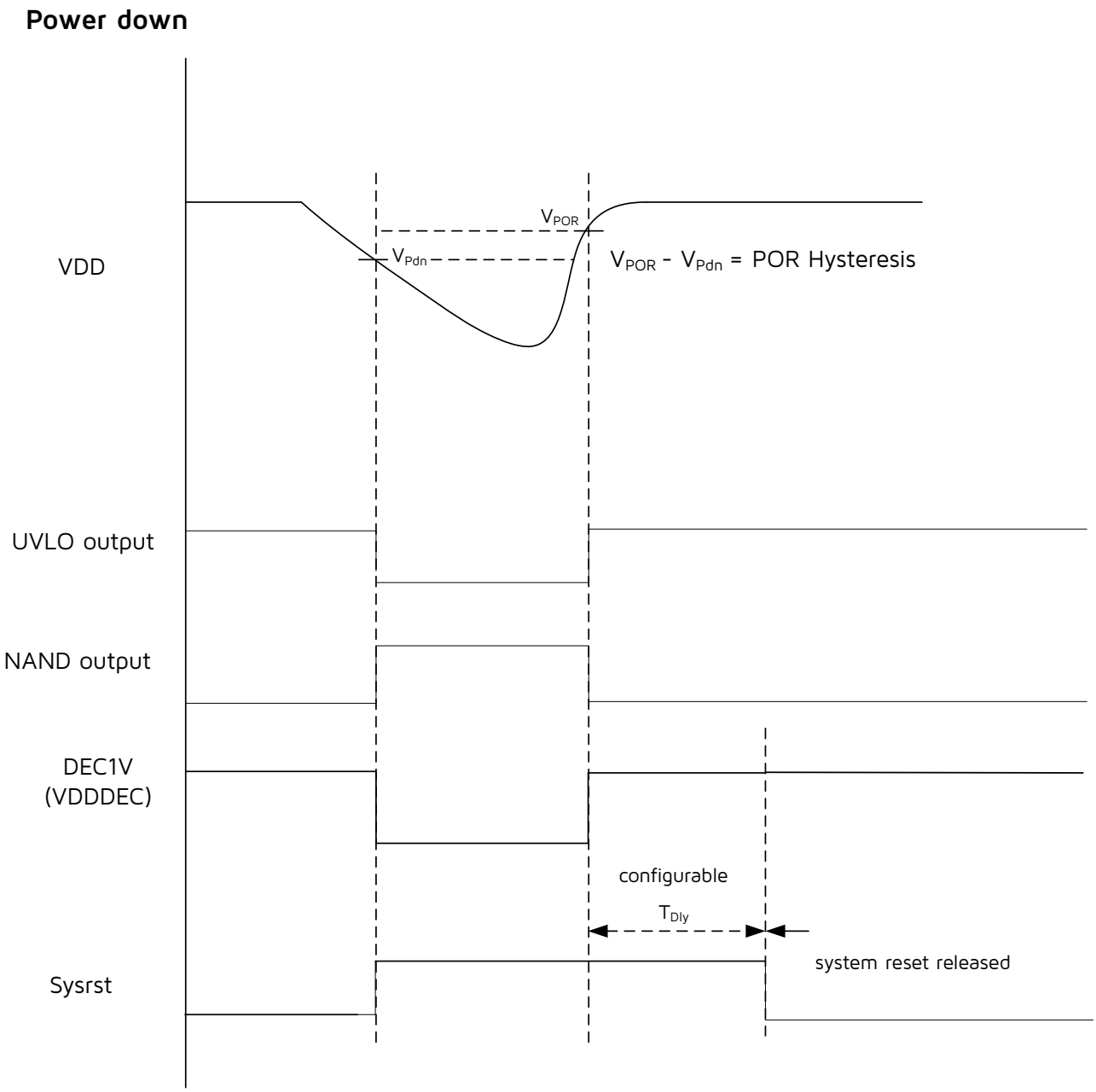


Table 4-10 Characteristics of Initial Power-up/Power-down Sequence

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{POR}	VDD voltage when V_{UVLO} turns to high level	-	1.73	-	V
V_{PDN}	VDD voltage when V_{UVLO} turns to low level	-	1.61	-	V
T_{DLY}	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 RESETB 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, Shutdown) 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 RESETB 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 RESETB 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 RESETB pin. The time to switch to Active mode is 1 ms or so.
- In Shutdown mode, all digital and analog modules are powered down, and only the PM module is active. The chip can be triggered to Active mode by RESETB pin only. The time to switch to Active mode is 10 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

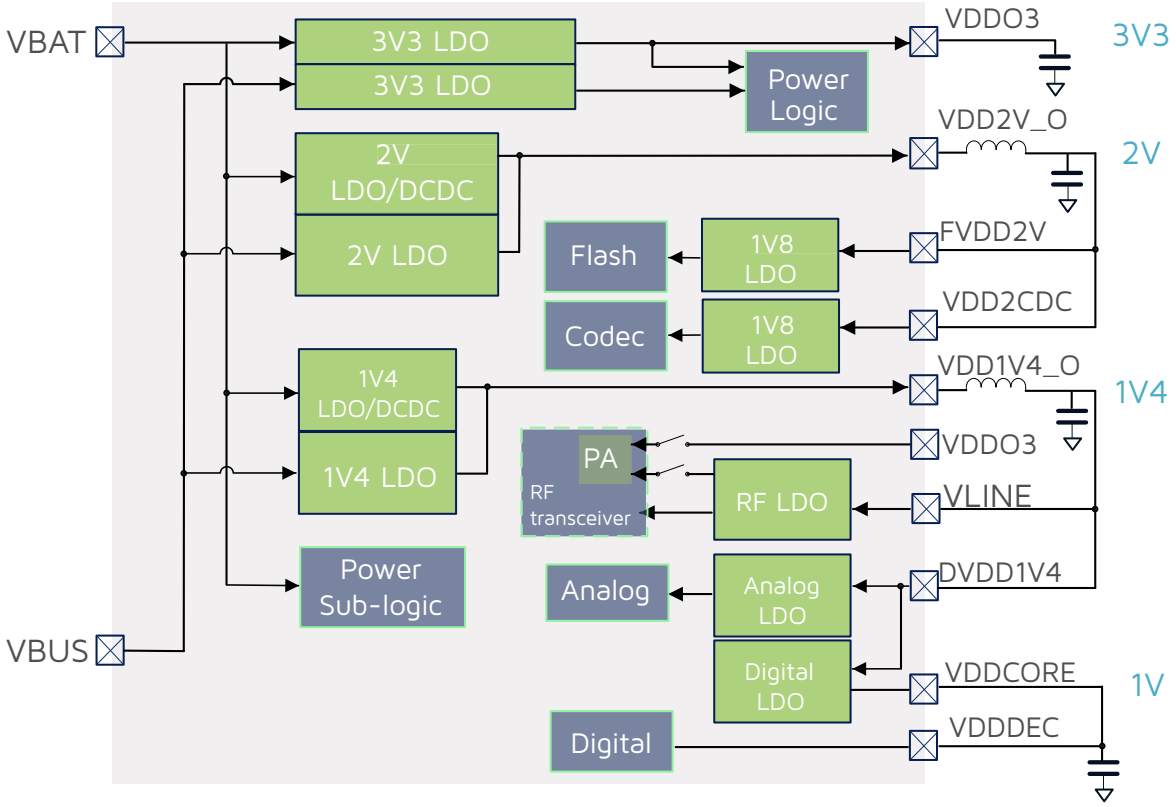
Address	Type	Description	Reset Value
afe_0x4c	R/W	[0] 1: auto power down 32KRC [1] 1: auto power down 32K xtal [2] 1: auto power down 4M rcosc [3] 1: auto power down 24M xtal [4] 1: auto power power logic [5] 1: auto power down dcdc [6] 1: auto power down vbus LDO [7] 1: auto power down ana/BBPLL/ temp_sensor LDO	0x0

Address	Type	Description	Reset Value
afe_0x4d	R/W	[0] 1: auto power down low power comparator [1] 1: auto power down dcore/sram LDO [2] 1: auto power down UVLO ib [3] 1: auto power down vbus switch [4] 1: auto power down flash LDO [5] rsvd [6] 1: power down sequence enable [7] 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 has two modes of power supply VBAT and VBUS. After power up, the 3.3 V LDO can generate 3.3 V output and supply power for Power logic module; the 2 V LDO or 2 V DCDC can generate 2 V voltage output and through internal 1.8V LDOs to supply power for flash and CODEC modules; the 1.4 V LDO or 1.4 V DCDC can generate 1.4 V voltage output as the input of the internal RF LDO, analog LDO and digital LDO; the three LDOs can supply power for RF, Analog, and Digital modules respectively; the digital LDO can generate 1.2 V output as VDDCORE, as well as VDDDEC to supply power for digital signals.

4.5.4 VBAT and VANT Power-Supply Mode

The chip provides two power-supply modes including VBAT mode and VANT mode.

- In VBAT mode, the chip is directly supplied with power by its battery voltage. The maximum output power is related to power supply voltage, for example, the maximum power is 10 dBm or so at 3.3 V power supply.
- In VANT mode, the chip is supplied with 1.2 V voltage by the embedded DCDC and LDO. In this mode, output power won't change with AVDD basically, and the maximum power is 5 dBm or so. Corresponding to the VBAT mode, the VANT mode is more power-saving at the same TX power.

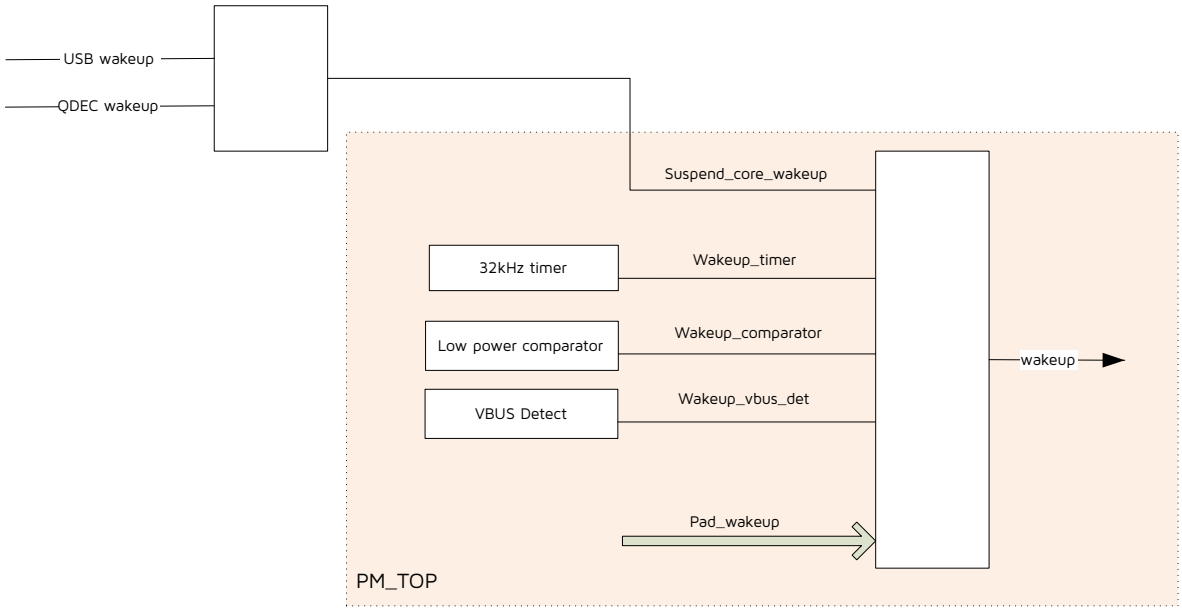
NOTE:

- When the VBAT is in VBAT_MAX_VALUE_LESS_THAN_3V6 mode, the IO voltage of GPIO cannot be configured to 1.8V.
- The VBAT channel detection does not support using 1.8V.

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

These wakeup sources can only wake up the system from suspend mode.

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

For QDEC wakeup, it is mainly used in mouse applications, details refer to Chapter 19 Quadrature Decoder.

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.

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

Address	Type	Description	Default Value
afe_0x4b	R/W	[0] pad wakeup enable [1] rsvd [2] timer wakeup enable [3] comparator wakeup enable [4] rsvd [5] rsvd [6] VAD wakeup enable [7] shutdown wakeup enable	0x0
afe_0x64	R	write 1 to clean the status: [0]: wkup pad [1]: rsvd [2]: wkup timer [3]: wkup cmp [4]: rsvd [5]: rsvd [6]: wkup vad [7]: watch_dog	-
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 BLE/2.4GHz RF Transceiver

5.1 Overview

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

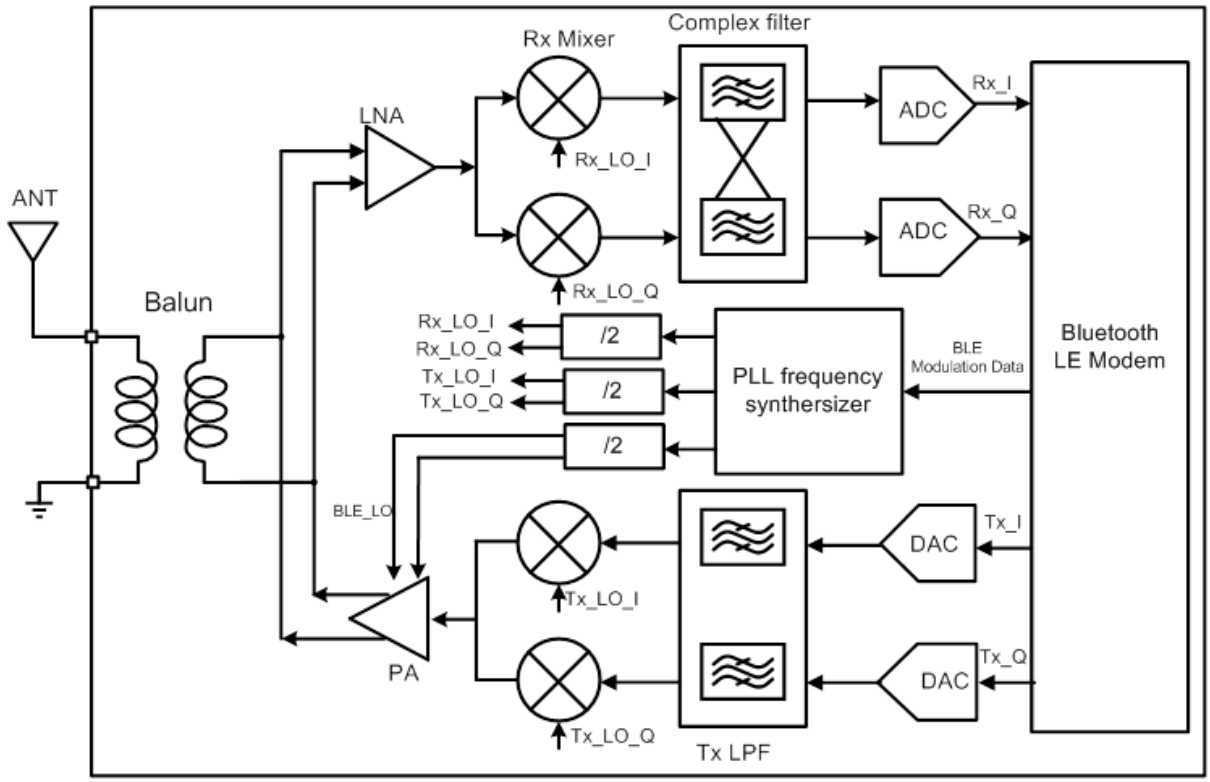
This RF transceiver works in the worldwide 2.4 GHz ISM band and it consists of a fully integrated RF synthesizer, a Power Amplifier (PA), a Low Noise Amplifier (LNA), a TX LPF, a Rx complex filter, a TX DAC, a RX ADC, BLE modulator/Demodulator and on-chip balun.

The BLE mode works in standard-compliant 1 Mbps BLE mode, 2 Mbps enhancement BLE mode,125 Kbps BLE long range mode (S8), 500kbps BLE long range mode (S2).

The Zigbee mode works in IEEE 802.15.4 standard-compliant 250 Kbps mode.

The block diagram of the transceiver is shown below.

Figure 5-1 Block Diagram of RF Transceiver



5.2 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, 3 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.

5.3 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 LE and 802.15.4 specification.

5.3.1 Packet Format

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

Table 5-1 Packet Format in Standard 1 Mbps BLE Mode

LSB MSB

Preamble (1 octet)	Access Address (4 octets)	PDU (2 ~ 257 octets)	CRC (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 5-2 Packet Format in Standard 2 Mbps BLE Mode

LSB MSB

Preamble (2 octet)	Access Address (4 octets)	PDU (2 ~ 257 octets)	CRC (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 5-3 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 5-4 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 5-5 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)

5.3.2 BLE Location Function - Direction

A Bluetooth LE device can make its direction available for a peer device by transmitting direction finding enabled packets. Using direction information from several transmitters and profile-level information giving their locations, an LE radio can calculate its own position. The Angle of Arrival (AoA) and Angle of Departure (AoD) are Bluetooth Low Energy feature, which can be used to determine the direction of a peer device. The feature is available for the Bluetooth Low Energy 1 and 2 Mbps modes. When using this feature, the transmitter sends a packet with a constant tone extension (CTE) appended to the packet, after the CRC. During the CTE, the receiver can take IQ samples of the incoming signal.

In the location mode of operation, the chip transmits a training sequence concatenated to the normal packet transmissions. In AoA mode of operation, the receiving side has multiple antennas and will be switched during the training sequence period. In AoD mode of operation, the transmitting side has multiple antennas and will be switched during the training sequence period. In either mode, the receiving side will be able to determine based on the phase variations of the received training sequences, the angle of location of the peer device.

5.3.3 RSSI and Frequency Offset

The SoC provides accurate RSSI (Receiver Signal Strength Indicator) and frequency offset indication.

- RSSI can be read from the 1 byte at the tail of each received data packet.
- If no data packet is received (e.g. to perform channel energy measurement when no desired signal is present), real-time RSSI can also be read from specific registers which will be updated automatically.
- RSSI monitoring resolution can reach +/-1 dB.
- Frequency offset can be read from the 2 bytes at the tail of the data packet. Valid bits of actual frequency offset may be less than 16 bits, and different valid bits correspond to different tolerance range.

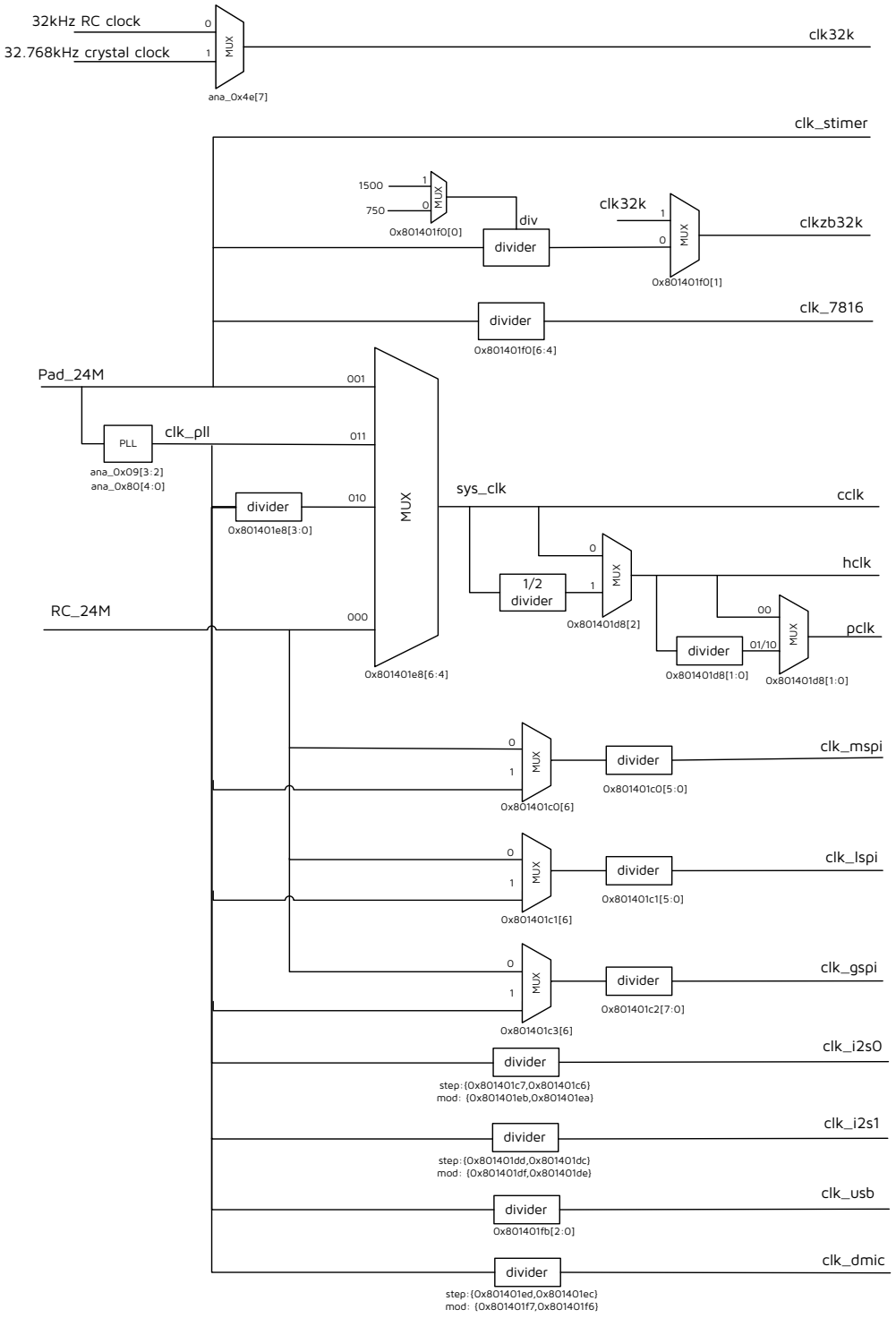
Telink supplies corresponding drivers for user to read RSSI and frequency offset as needed.

6 Clock

6.1 Clock Sources

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

Figure 6-1 Clock Sources



NOTE: *The maximum frequency supported by cclk is 96 MHz.*

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

Table 6-1 Clock Sources of Each Module

Module	Clock Source(s)
HSPI	hclk
PLMT	clk32k, hclk
PLIC	hclk
MCU	cclk
ILM	cclk
DLM	cclk
MSPI	hclk, clk_mspi
GSPI	hclk, clk_gspi
LSPI	hclk, clk_lspi
ZB	clkzb32k, hclk
PKE	hclk
TRNG	hclk
SWIRE	hclk
USB	hclk, clk_usb
DMA	hclk
BMC	hclk
I2C	pclk
PWM	pclk, clk32k
GPIO	pclk
QDEC	pclk
STIMER	pclk, clk32k, clk_stimer
ALGM	pclk
TIMER	pclk
UART1	pclk
UART0	pclk

Module	Clock Source(s)
SPI_SLV	hclk

6.2 System Clock

There are four selectable clock sources for MCU system clock: RC_24M derived from 24 MHz RC oscillator, 24 MHz crystal, sclk_div, and pll clk. The sources are selectable via register CLKSEL0.

6.3 Module Clock

Registers CLKEN0~CLKEN3 are used to enable or disable clock for various modules. By disable the clocks of unused modules, current consumption could be reduced.

6.3.1 System Timer Clock

System timer clock is derived from 24 MHz crystal oscillator.

6.3.2 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)

6.3.3 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].

6.3.4 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].

6.3.5 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].

6.3.6 clkzb32k

When CLK_DIV[0] = 1, Fclkzb32k=Fpad_24m/1500; when CLK_DIV[0] = 0, Fclkzb32k=Fpad_24m/700;when register CLK_DIV[1] = 1, clkzb32k chooses clk32k, when CLK_DIV[1] = 0, clkzb32 chooses the clock generated by pad_24M via a frequency divider.

6.3.7 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)

6.3.8 clk_mspi

The clk_mspi is the system clock of MSPI module. The default value of CLKSEL0[7] is 1, and clk_mspi chooses sys_clk.

If MSPI_MODE[6] = 1, Fclk_mspi = Fpllclk/n (n=MSPI_MODE[5:0], n=1~63);

If MSPI_MODE[6] = 0, Fclk_mspi = F24M/n (n=MSPI_MODE[5:0], n=1~63).

6.3.9 clk_lspi

If LSPI_MODE[6] = 1, Fclk_lspi = Fpllclk/n (n=LSPI_MODE[5:0], n=1~63);

If LSPI_MODE[6] = 0, Fclk_lspi = F24M/n (n=LSPI_MODE[5:0], n=1~63).

6.3.10 clk_gspi

If GSPI_MODE_H[6] = 1, Fclk_gspi = Fpllclk/n (n=GSPI_MODE_L[7:0], n=1~255);

If GSPI_MODE_H[6] = 0, Fclk_gspi = F24M/n (n=GSPI_MODE_L[7:0], n=1~255).

6.4 Register Table

Clock related registers are listed in table below. The base address of the following registers is 0x801401c0.

Table 6-2 Clock Related Registers

Address Offset	Type	Description	Reset Value
0x12	R/W	CLKMOD [3:0]: zb_mst_mod, zb master clock divider mod output clk is 1/(zb_mst_mod+1)	0x03

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

Address Offset	Type	Description	Reset Value
0x27	R/W	CLKEN3 [0]:zb_pclk [1]:zb_mstclk [2]:zb_lpclk [3]: rsvd [4]:mspi [5]:qdec [6]:rsvd2 [7]:rsvd3	0x10
0x28	R/W	CLKSEL0 [3:0]:sclk_div [6:4]:sclk_sel, 000:24M_rc, 001:24M xtal, 010:sclk_div, 011:pll clk	0x02
0x2a	R/W	I2S0_MOD_L [7:0]: i2s0_mod_l	0x02
0x2b	R/W	I2S0_MOD_H [7:0]: i2s0_mod_h	0x00
0x2c	R/W	DMIC_STEP_L [7:0]:dmic_step_l	0x01
0x2d	R/W	DMIC_STEP_H [6:0]:dmic_step_h [7]: dmic_clk_sel	0x00
0x2e	R/W	WAKEUPEN [0]: usb_pwdn_i, enable wakeup from usb [1]: gpio_wakeup_i, enable wakeup from gpio [2]: qdec_wakeup_i, enable wakeup from qdec [4]: usb resume, enable remote wakeup from USB [5]: standby ex [7:6]: reserved	0x00

Address Offset	Type	Description	Reset Value
0x2f	R/W	PWDNEN [0]: suspend_en_o [4]: ramcrc_clren_tgl [5]: rst_all [7]: stall_en_trg	0x00
0x30	R/W	CLK_DIV [0:2]: clkzb32k_sel [6:4]: r_7816_mod, 7816 clk div [7]: r_7816_clk_en, 7816 clk enable	0x62

7 Timer

7.1 Timer0 ~ Timer1

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).

7.1.1 Mode 0 (System Clock Mode)

In Mode 0, system clock is employed as clock source.

After Timer is enabled, 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.

Once current Timer Tick value matches the preset Timer Capture (i.e. timing value), an interrupt is generated, Timer stops 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 Capture value of Timer0

Set registers TMR_CAPT0_0~TMR_CAPT0_3, from lowest byte to highest byte respectively.

Step 3 Set Timer0 to Mode 0 and enable Timer0

Set register TMR_CTRL0 [2:1] to 2'b00 to select Mode 0; Meanwhile set TMR_CTRL0 [0] to 1'b1 to enable Timer0. Timer0 starts counting upward, and Tick value is increased by 1 on each positive edge of system clock until it reaches Timer0 Capture value.

7.1.2 Mode 1 (GPIO Trigger Mode)

In Mode 1, GPIO is employed as clock source. The "**M0**"/"**M1**"/"**M2**" register specifies the GPIO which generates counting signal for Timer0/Timer1/Timer2.

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive/negative (configurable) edge of GPIO from preset initial Tick value. Generally the initial Tick value is set to 0. The "**Polarity**" register specifies the GPIO edge when Timer Tick counting increases.

Once current Timer Tick value matches the preset Timer Capture (i.e. timing value), an interrupt is generated and timer stops counting.

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 Capture value of Timer1

Set registers TMR_CAPT1_0~TMR_CAPT1_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.

Select positive edge or negative edge of GPIO input to trigger Timer1 Tick increment via setting "Polarity" register.

Step 4 Set Timer1 to Mode 1 and enable Timer1

Set TMR_CTRL0 [5:4] to 2'b01 to select Mode 1; Meanwhile set TMR_CTRL0 [3] 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 until it reaches Timer1 Capture value.

7.1.3 Mode 2 (GPIO Pulse Width Mode)

In Mode 2, system clock is employed as the unit to measure the width of GPIO pulse. The "**M0**"/"**M1**"/"**M2**" register specifies the GPIO which generates control signal for Timer0/Timer1/Timer2.

After Timer is enabled, Timer Tick is triggered by a positive/negative (configurable) edge 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. The "**Polarity**" register specifies the GPIO edge when Timer Tick starts counting.

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 system clock.

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.

Select positive edge or negative edge of GPIO input to trigger Timer2 counting start via setting "Polarity" register.

Step 3 Set Timer2 to Mode 2 and 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 system clock.

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

Step 4 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

7.1.4 Mode 3 (Tick Mode)

In Mode 3, system clock is employed.

After Timer is enabled, Timer Tick starts counting upward, and Timer Tick value is increased by 1 on each positive edge of system clock.

This mode could be used as time indicator. There will be 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 [2:1] to 2'b11 to select Mode 3, meanwhile set address TMR_CTRL0 [0] 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 fromTMR_TICK0_1 ~TMR_TICK0_3.

7.1.5 Watchdog

Programmable watchdog could reset chip from unexpected hang up or malfunction.

Watchdog Capture has 24bits, which consists of WT_TARGET_1~WT_TARGET_3 as byte 1 ~byte 3. Chip will be reset when TMR_CTRL3[2] is set to 1.

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_CTRL2[7] to 1'b1 to enable Watchdog.

During normal working condition, TMR_CTRL3[3] need write 1 to clean the watchdog before the watchdog hits WT_TARGET3-1, or it will reboot the whole chip, and the TMR_CTRL3[2] will be assert to 1, this bit will be clean when write 1.

7.1.6 Register Table

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

Table 7-1 Registers for Timer 0 ~ Timer 1

Offset	Type	Description	Reset Value
0x00	R/W	TMR_CTRL0 [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, Timer0 nowrap [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, Timer1 nowrap	0x0
0x02	R/W	TMR_CTRL2 [6:0] reserved [7] watchdog_en	0x0
0x03	R/W	TMR_CTRL3 [0] tmr0, tmr0_o=tmr0 [1] tmr1, tmr1_o=tmr1 [2] wdstat, hit watchdog target [3] wd_cnt_clr, clear wd_cnt [6:4] reserved [7] software_irq	0x0
0x04	R/W	TMR_CAPT0_0 capt0[7:0] Byte 0 of timer0 capture	0x0
0x05	R/W	TMR_CAPT0_1 capt0[15:8] Byte 1 of timer0 capture	0x0
0x06	R/W	TMR_CAPT0_2 capt0[23:16] Byte 2 of timer0 capture	0x0
0x07	R/W	TMR_CAPT0_3 capt0[31:24] Byte 3 of timer0 capture	0x0
0x08	R/W	TMR_CAPT1_0 capt1[7:0] Byte 0 of timer1 capture	0x0

Offset	Type	Description	Reset Value
0x09	R/W	TMR_CAPT1_1 capt1[15:8] Byte 1 of timer1 capture	0x0
0x0a	R/W	TMR_CAPT1_2 capt1[23:16] Byte 2 of timer1 capture	0x0
0x0b	R/W	TMR_CAPT1_3 capt1[31:24] Byte 3 of timer1 capture	0x0
0x0d	R/W	WT_TARGET_1 watchdog_target2[15:8] Byte 1 of watchdog target value	0x0
0x0e	R/W	WT_TARGET_2 watchdog_target2[23:16] Byte 2 of watchdog target value	0x0
0x0f	R/W	WT_TARGET_3 watchdog_target2[31:24] Byte 3 of watchdog target value	0x0
0x10	R/W	TMR_TICK0_0 ticko[7:0] Byte 0 of timer0 ticker	0x0
0x11	R/W	TMR_TICK0_1 ticko[15:8] Byte 1 of timer0 ticker	0x0
0x12	R/W	TMR_TICK0_2 ticko[23:16] Byte 2 of timer0 ticker	0x0
0x13	R/W	TMR_TICK0_3 ticko[31:24] Byte 3 of timer0 ticker	0x0
0x14	R/W	TMR_TICK1_0 tick1[7:0] Byte 0 of timer1 ticker	0x0
0x15	R/W	TMR_TICK1_1 tick1[15:8] Byte 1 of timer1 ticker	0x0
0x16	R/W	TMR_TICK1_2 tick1[23:16] Byte 2 of timer1 ticker	0x0
0x17	R/W	TMR_TICK1_3 tick1[31:24] Byte 3 of timer1 ticker	0x0

7.2 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 corresponding register configuration is as follows.

Table 7-2 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]:wkup vad [7]:watch_dog	-
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 watch dog enable signal, write 1 to enable the watch dog function.

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 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 will reset the whole system after 5 seconds).

The afe_0x64[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.

7.3 System Timer

The SoC also supports a System Timer, the clock frequency for System Timer is fixed as 24 MHz irrespective of system clock.

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 will continue counting from the number when they stops; In contrast, System Timer will continue 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.

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

Table 7-3 System Timer Related Registers

Offset	Type	Description	Reset Value
0x00	R/W	SYS_TIMER0	0x00
0x01	R/W	SYS_TIMER1	0x00
0x02	R/W	SYS_TIMER2	0x00
0x03	R/W	SYS_TIMER3	0x00
0x0a	R/W	SYS_TIMER_CTRL [0] wr_32k, 1:32k write mode; 0:32k read mode [1] timer_en, system timer enable [2] timer_auto [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	R/W	SYS_TIMER_ST [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

7.4 Platform-Level Machine Timer (PLMT)

7.4.1 Introduction

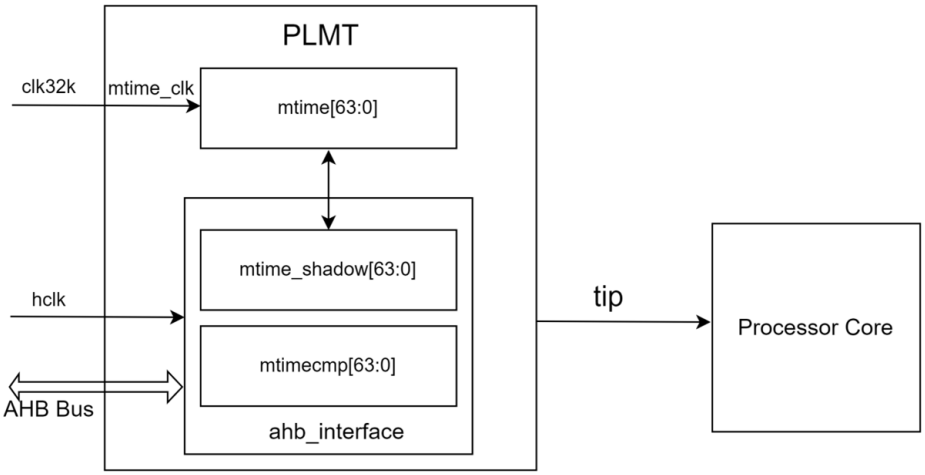
The RISC-V architecture defines a machine timer that provides a real-time counter and generates timer interrupts. Platform-Level Machine Timer (PLMT) is an implementation of the machine timer.

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 7-1 Block Diagram of PLMT



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.

7.4.2 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;

7.4.3 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].

7.4.4 Register Description

PLMT related registers are listed in table below. The base address for the following registers is 0xE6000000.

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 7-4 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

8 Trap and PLIC

8.1 Trap

8.1.1 Introduction

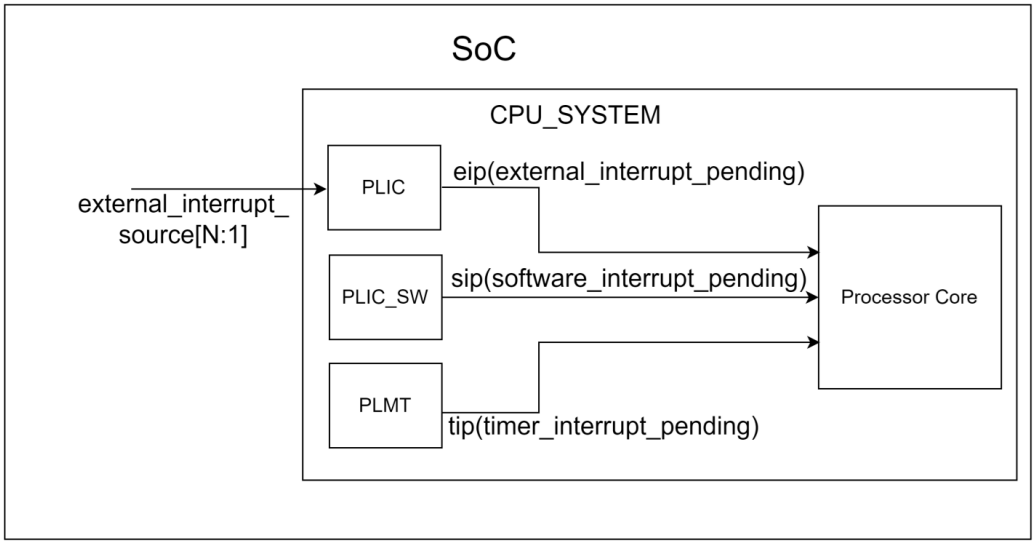
According to the RISC-V Privileged Architecture, a trap is a control flow change of normal instruction execution caused by an interrupt or an exception. An interrupt is a control flow change event initiated by an external source. An exception is a control flow change event generated as a by-product of 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.

8.1.2 Interrupt

Figure 8-1 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.

8.1.2.1 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

8.1.2.2 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.

8.1.2.3 Interrupt Priority

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

Table 8-1 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)

8.1.3 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

8.1.4 Trap Handling

8.1.4.1 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 will be 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.

8.1.4.2 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.

8.1.5 Machine Trap Related CSRs

8.1.5.1 Machine Status

Mnemonic Name: mstatus

Access Mode: Machine

CSR Address: 0x300

Table 8-2 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

8.1.5.2 Machine Interrupt Enable

Mnemonic Name: mie

Access Mode: Machine

CSR Address: 0x304

Table 8-3 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

8.1.5.3 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 will be 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 8-4 Register Description of mtvec

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

8.1.5.4 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 8-5 Register Description of mepc

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

8.1.5.5 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 8-6 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 8-7 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 8-8 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 8-9 Possible values of mcause after vector interrupt

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

8.1.5.6 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 will be updated with the address pointing to the portion of the instruction that caused the fault, while the mepc register will be updated with the address pointing to the beginning of the instruction.

Table 8-10 Register Description of mtval

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

8.1.5.7 Machine Interrupt Pending

Mnemonic Name: mip

Access Mode: Machine

CSR Address: 0x344

Table 8-11 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

8.1.5.8 Machine Detailed Trap Cause

Mnemonic Name: mdcause

Access Mode: Machine

CSR Address: 0x7c9

Table 8-12 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 8-13 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

8.1.5.9 Machine Miscellaneous Control Register

Mnemonic Name: mmisc_ctl

Access Mode: Machine

CSR Address: 0x7d0

Table 8-14 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

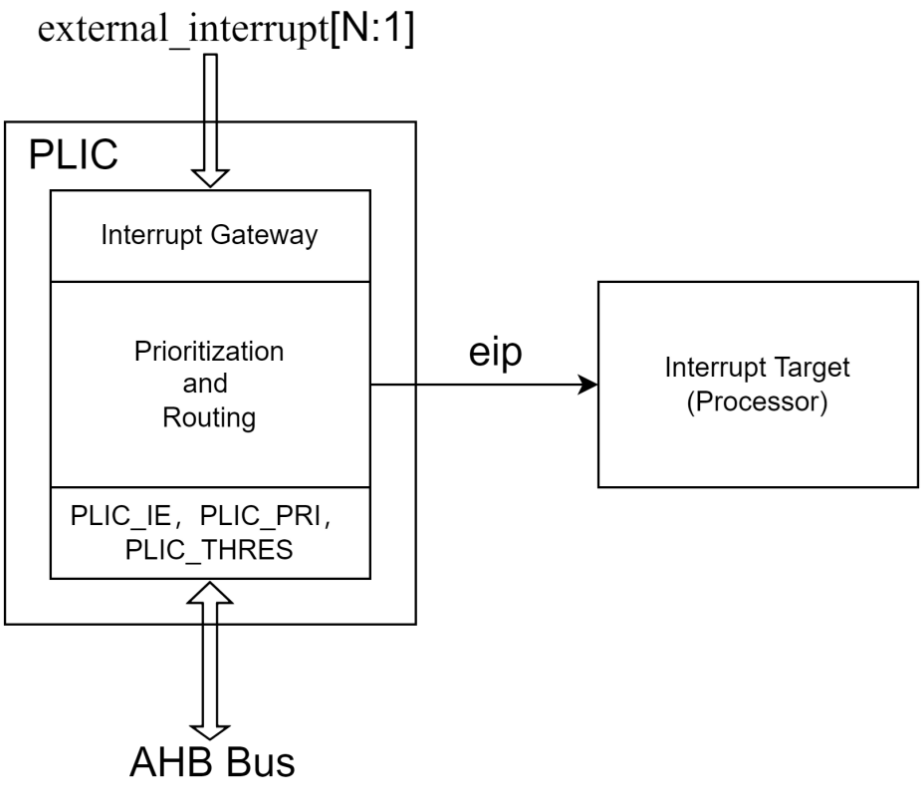
8.2 Platform-Level Interrupt Controller (PLIC)

8.2.1 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: 46
- Programmable interrupt priority: 1/2/3
- Preemptive priority interrupt extension
- Vectored interrupt extension
- Software-programmable interrupt generation

Figure 8-2 Block Diagram of PLIC



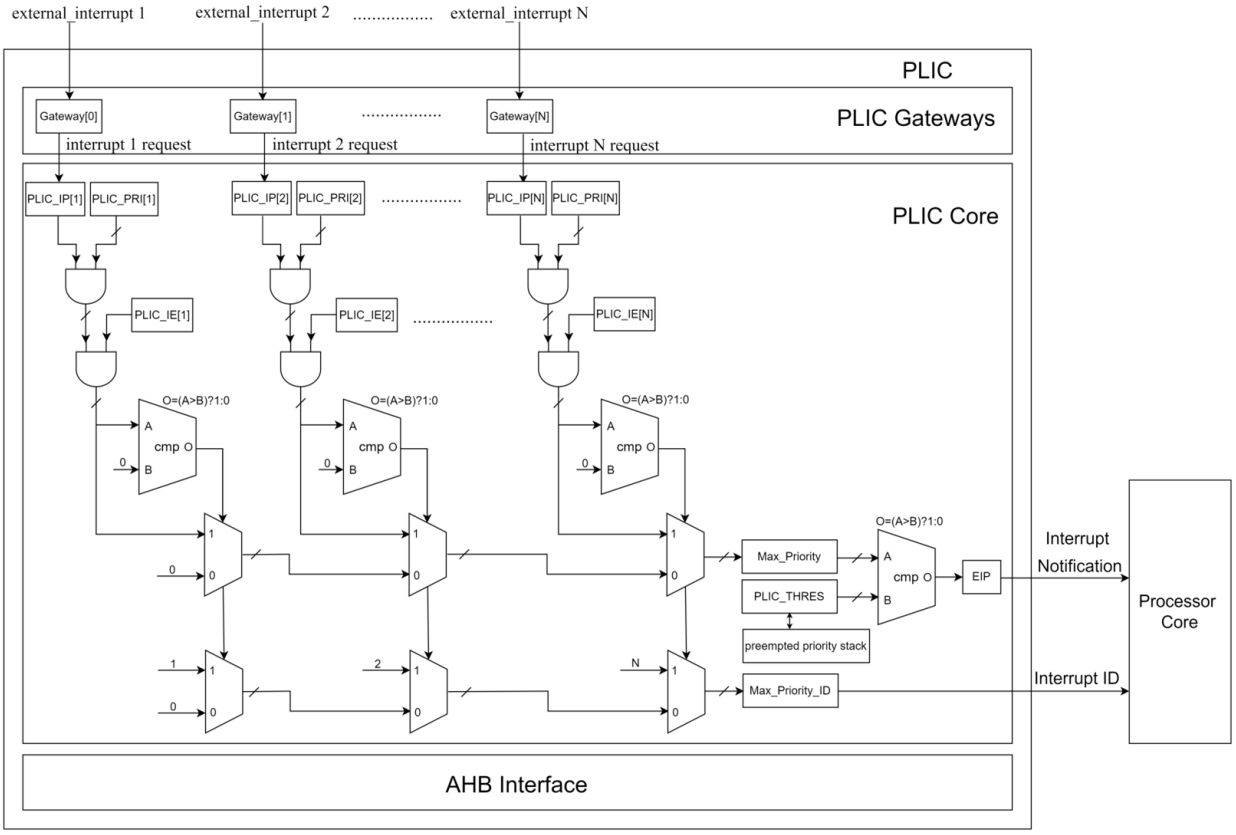
The above figure shows the block diagram of PLIC. External interrupt sources (e.g., peripherals) send interrupt requests to PLIC through external_interrupt[N:1] signals. The signals are level-triggered, and they are converted to interrupt requests by the interrupt gateway. Interrupt requests are prioritized 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 will be cleared and eip will be 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 8-3 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 will generate 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.

8.2.2 External Interrupt Sources

There are 46 external interrupt sources, listed in table below.

Table 8-15 Interrupt Sources

No.	Interrupt Source	No.	Interrupt Source
1	stimer_irq: system timer interrupt	24	usb_pwdn_irq: USB suspend interrupt
2	algm_irq: analog register master interface interrupt	25	gpio_irq
3	timer1_irq	26	gpio2risc[0]_irq
4	timer0_irq	27	gpio2risc[1]_irq
5	dma_irq	28	soft_irq: software interrupt
6	bmc_irq: ahb bus matrix controller interrupt	29	mspi_irq
7	usb_setup_irq: USB setup interrupt	30	usb_reset_irq: USB reset interrupt
8	usb_data_irq: USB data interrupt	31	usb_250us_sof_irq: USB 250us or SOF interrupt
9	usb_status_irq: USB status interrupt	32	reserved
10	usb_setinf_irq: USB set interface interrupt	33	qdec_irq
11	usb_edp_irq: USB edp(1-8) interrupt	34	gpio_src_irq[0]: gpio_group_irq[0]
12	reserved	35	gpio_src_irq[1]: gpio_group_irq[1]
13	reserved	36	gpio_src_irq[2]: gpio_group_irq[2]
14	zb_bt_irq:BR/EDR sub-system interrupt-	37	gpio_src_irq[3]: gpio_group_irq[3]
15	zb_ble_tl_irq:BLE(TL) sub-system interrupt	38	gpio_src_irq[4]: gpio_group_irq[4]
16	pwm_irq	39	gpio_src_irq[5]: gpio_group_irq[5]
17	pke_irq	40	gpio_src_irq[6]: gpio_group_irq[6]
18	uart1_irq	41	gpio_src_irq[7]: gpio_group_irq[7]

No.	Interrupt Source	No.	Interrupt Source
19	uart0_irq	42	reserved
20	dfifo_irq: audio dma fifo interrupt	43	reserved
21	i2c_irq	44	pm_wkup_irq: PM wakeup interrupt
22	lspi_irq	45	pm_mix_irq: PM mixed interrupt
23	gspi_irq	46	dpr_irq

8.2.3 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 will stop servicing the current low-priority interrupt and begin servicing this new high-priority interrupt. The handling of the suspended lower-priority interrupts will resume only after the handling of the higher-priority interrupt ends. Interrupts of same or lower priorities will 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.

8.2.3.1 Interrupt Claims with Preemptive Priority

When the target sends an interrupt claim message to the PLIC core, the PLIC core will atomically determine 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 will then return the ID to the target.

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

8.2.3.2 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 will restore 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.

8.2.3.3 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 will trigger 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

8.2.4 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.

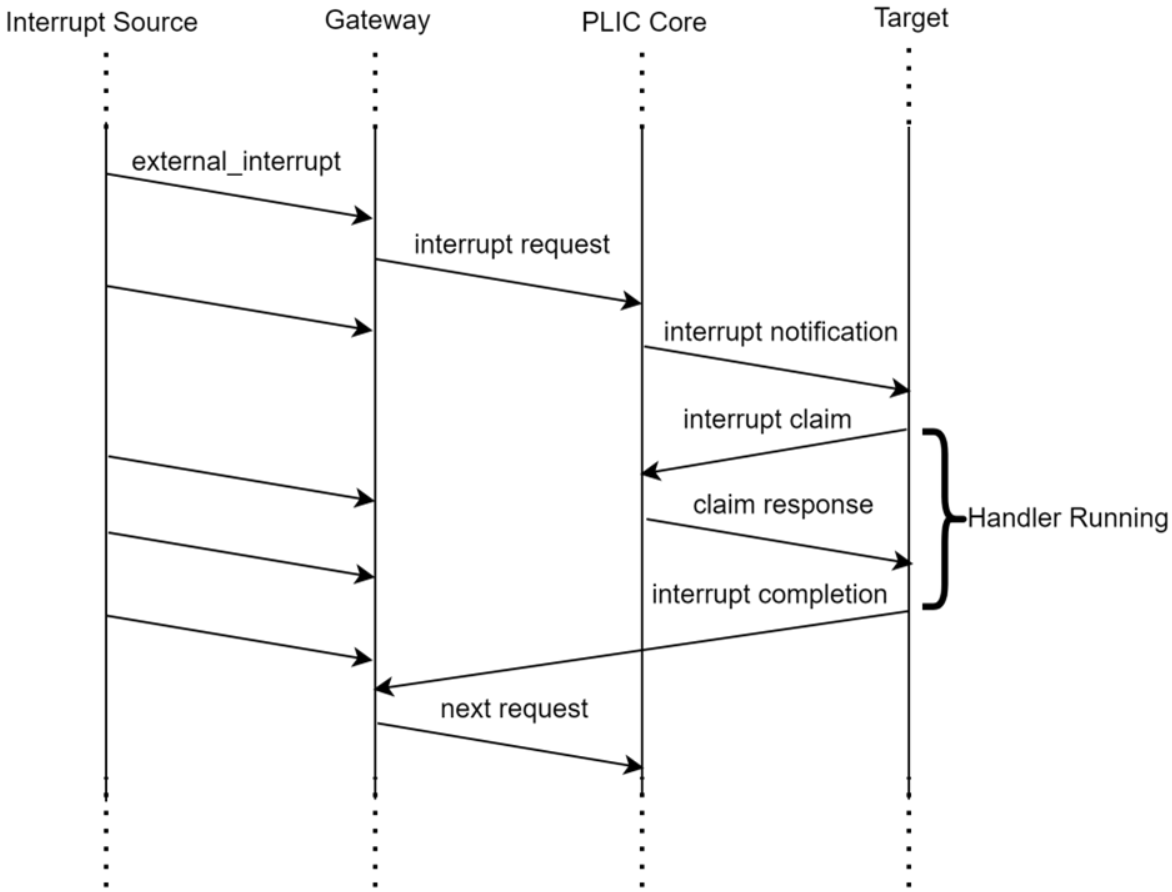
8.2.5 Support for Software-Generated Interrupt

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.

8.2.6 Interrupt Flow

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

Figure 8-4 Interrupt Flow



The gateway will only forward a single interrupt request at a time to the PLIC, and not forward subsequent interrupts requests until an interrupt completion is received. The PLIC will set 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 will then send an interrupt claim request to the PLIC core to obtain the interrupt ID. The PLIC core will atomically return 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.

8.2.7 Register Description

PLIC related register are listed in table below. The base address for the following registers is 0xE4000000.

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 8-16 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 32~46 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 32~46.	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 32~46 Every interrupt source occupies 1 bit. [14:0]: 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

8.3 Software Platform-Level Interrupt Controller (PLIC_SW)

8.3.1 Introduction

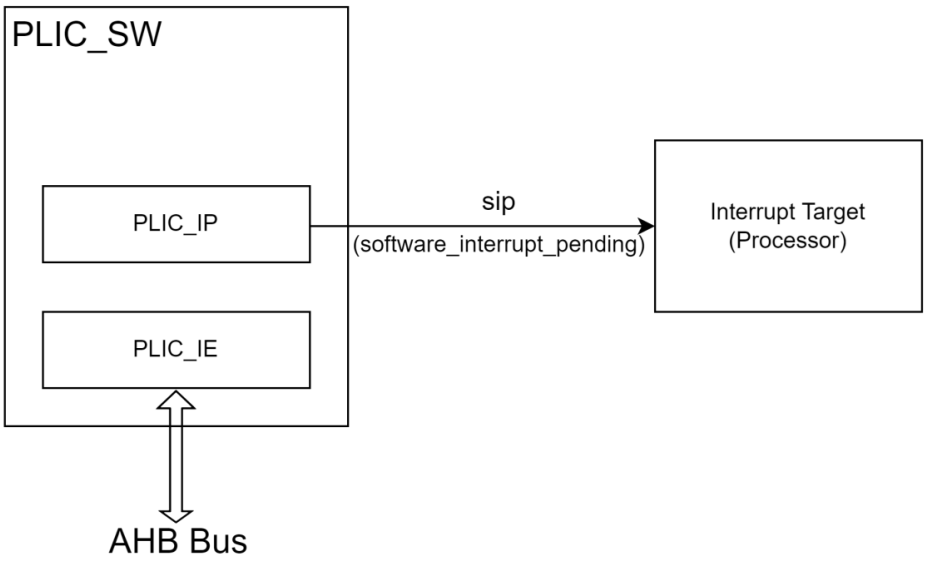
The SoC embeds another one PLIC named PLIC_SW for Software-programmable interrupt generation.It is compatible with RISC-V PLIC with the following features:

- 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 8-5 Block Diagram of PLIC_SW



8.3.2 Register Description

The PLIC_SW related register are listed in table below. The base address for the following registers is 0xE6400000.

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 8-17 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

9 DMA

9.1 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

9.1.1 Function Description

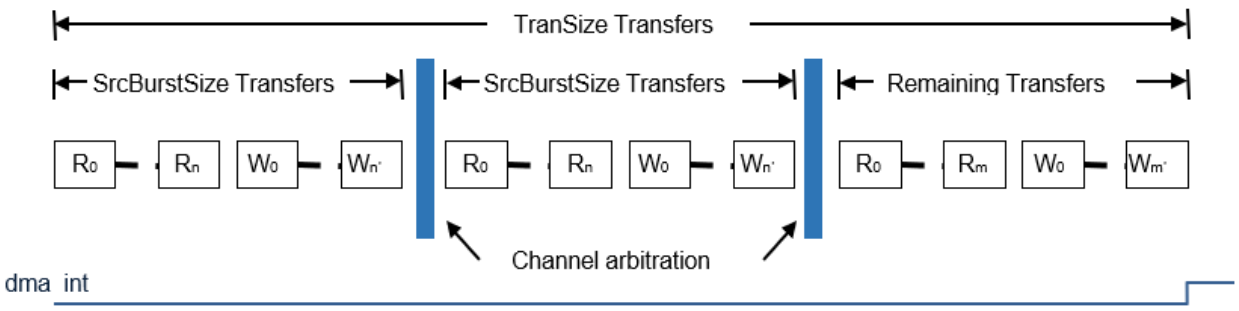
DMAC supports up to 8 DMA channels. Each DMA channel provides a set of registers to describe the intended data transfers. Multiple DMA channels can be enabled concurrently, but the DMA controller services one channel at a time.

Figure 9-1 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 9-6. 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 will update 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 will be stopped when an error occurs. The data transfers of a channel can also be aborted by software. In either case, the DMA controller will disable the channel, and assert the interrupt signal if the corresponding interrupt is enabled.

Figure 9-1 Example of DMA Data Transfers



9.1.1.1 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 will be selected by the round-robin scheme.

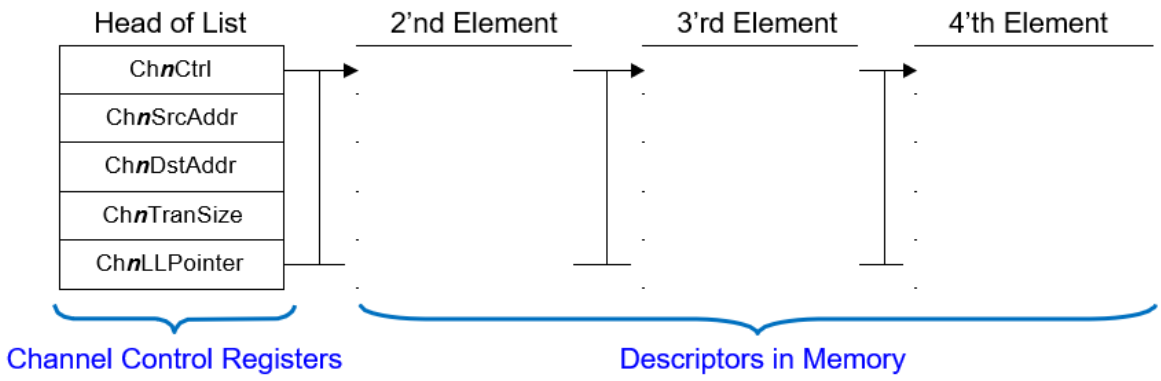
9.1.1.2 Chain Transfer

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 will first transfer data according to the channel control registers. After the data transfer completes, the DMA controller will continue the data transfer by following the Ch*n*LLPointer. The content of the linked list descriptor pointed by Ch*n*LLPointer will be 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 9-2 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 will generate 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 will be 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 will generate 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 will be 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 will generate 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 will be generated after each channel control register is completed, and the linked list will be 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 9-1 Format of Linked List Descriptor

Name	Offset	Description	Format
Ctrl	0x00	Channel control	See Table 9-6
SrcAddr	0x04	Source address	See Table 9-8
DstAddr	0x08	Destination address	See Table 9-9
TranSize	0x0C	Total transfer size	See Table 9-10
LLPointer	0x10	Linked list pointer	See Table 9-11

The peripherals supporting chain transfer include: RX of UART, and audio.

9.1.1.3 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.

9.2 Registers

9.2.1 Register Summary

The table below shows a summary of the DMA registers. The base address of DMA is 0x80100400.

Table 9-2 DMA Register Summary

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	

Table 9-3 DMAC Control Register

Name	Bit	Type	Description	Reset
Reserved	31:1	-	-	-
Soft_Reset	0	WO	DMA internal software reset control. Set this bit to 1 to reset the DMA core and disable all channels.	0x0

9.2.2 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 9-4 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

9.2.3 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 9-5 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 will be ignored for channels that are not enabled.	0x0

9.2.4 Channel n Control Register (Offset 0x44+n*0x14)

Table 9-6 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 will write 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 will write 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 will be 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 9-7.	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 9-7.	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 9-7 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

9.2.5 Channel n Source Address Register (Offset 0x48+n*0x14)

Table 9-8 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 will be triggered.	0x0

9.2.6 Channel n Destination Address Register (Offset 0x4C+n*0x14)

Table 9-9 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 will be 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.

9.2.7 Channel n Transfer Size Register (Offset 0x50+n*0x14)

Table 9-10 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 will be triggered and the transfer will be 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.

9.2.8 Channel n Linked List Pointer Register (Offset 0x54+n*0x14)

Table 9-11 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	-	-	-

9.2.9 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 9-12 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

Name	Address	Bit	Type	Description	Reset
BB_RX_WPTR	0xf4	4:0	R/W	RX_WPTR pointer	0x0
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 will automatically load 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 will check 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 will read 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

9.2.10 Miscellaneous Register

Table 9-13 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

9.3 Usage Guide

9.3.1 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) will be 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 9-14 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 9-6.
22-23	srcwidth	Set to any value stated in Table 9-6.

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.

9.3.2 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 9-15 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 9-7.
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.

9.3.3 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 9-16 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 9-7.
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.

9.3.4 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 will automatically load the source address register. After enabling rnum_en, DMA will automatically load the transize register (also needing to set ch_0_rnum_en_bk at dma_base+0x10c to 1 since rnum_en will be cleared to 0 after every first loading of transize, and it will automatically load 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 will be 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 will be 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 will read data from the default buff. If the current channel's FIFO is not empty, DMA will go 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.

9.3.5 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 will then be written into the buffer pointed to by rx_wptr. If the rxfifo's depth is zero, incoming data will continue to be written to the same location.

10 Interface

10.1 GPIO

The SoC supports up to 30 GPIOs. All digital IOs can be used as general purpose IOs.

10.1.1 Basic Configuration

All GPIOs can be configured with related registers, as described as following.

Table 10-1 GPIO Pad Function Mux

Pad	Default	Register = [63:1]	Register = 0	Register
PA[0]	GPIO	All functions	SWM	0x80140348[5:0]
PA[1]	GPIO	All functions	SWM	0x80140349[5:0]
PA[2]	GPIO	All functions[a]	SWM	0x8014034a[5:0]
PA[3]	GPIO	All functions	SWM	0x8014034b[5:0]
PA[4]	GPIO	All functions	SWM	0x8014034c[5:0]
PA[5]	GPIO	-	DM	0x8014034d[5:0]
PA[6]	GPIO	-	DP	0x8014034e[5:0]
PA[7]	SWS	-	SWS	0x8014034f[5:0]
PB[0]	GPIO	All functions	SWM	0x80140350[5:0]
PB[1]	GPIO	All functions	SWM	0x80140351[5:0]
PB[2]	GPIO	All functions	SWM	0x80140352[5:0]
PC[0]	SSPI_CN	All functions	SSPI_CN	0x80140358[5:0]
PC[1]	SSPI_CK	All functions	SSPI_CK	0x80140359[5:0]
PC[2]	SSPI_SI	All functions	SSPI_SI	0x8014035a[5:0]
PC[3]	SSPI_SO	All functions	SSPI_SO	0x8014035b[5:0]
PC[4]	TDI	All functions	TDI	0x8014035c[5:0]
PC[5]	TDO	All functions	TDO	0x8014035d[5:0]
PC[6]	TMS	All functions	TMS	0x8014035e[5:0]
PC[7]	TCK	All functions	TCK	0x8014035f[5:0]
PD[0]	GPIO	All functions	SWM	0x80140360[5:0]
PD[1]	GPIO	All functions	SWM	0x80140361[5:0]

Pad	Default	Register = [63:1]	Register = 0	Register
PD[2]	GPIO	All functions	SWM	0x80140362[5:0]
PE[0]	GPIO	-	LSPI_CN	0x80140368[5:0]
PE[1]	GPIO	-	LSPI_CK	0x80140369[5:0]
PE[2]	GPIO	-	LSPI_MOSI	0x8014036a[5:0]
PE[3]	GPIO	-	LSPI_MISO	0x8014036b[5:0]
PE[4]	GPIO	-	LSPI_IO2	0x8014036c[5:0]
PE[5]	GPIO	-	LSPI_IO3	0x8014036d[5:0]
PE[6]	GPIO	All functions	SWM	0x8014036e[5:0]
PE[7]	GPIO	All functions	SWM	0x8014036f[5:0]

a. "All functions" include 60 functions: GSPI_CN3, GSPI_CN2, GSPI_CN1, I2C1_SCL, I2C1_SDA, RX_CYC2LNA, ATSEL_3, ATSEL_2, ATSEL_1, ATSEL_0, BT_INBAND, BT_STATUS, BT_ACTIVITY, WIFI_DENY, TX_CYC2PA, DMIC1_DAT, DMIC1_CLK, DMIC0_DAT, DMIC0_CLK, I2S1_CLK, I2S1_DAT_IN, I2S1_LR_IN, I2S1_DAT_OUT, I2S1_LR_OUT, I2S1_BCK, I2S0_CLK, I2S0_DAT_IN, I2S0_LR_IN, I2S0_DAT_OUT, I2S0_LR_OUT, 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, and PWM0.

Table 10-2 GPIO Setting

Pad	Input	IE	OEN	Output	Polarity	DS	Act as GPIO
PA[0]	0x80140300 [0]	0x80140301 [0]	0x80140302 [0]	0x80140303 [0]	0x80140304 [0]	0x80140305 [0]	0x80140306 [0]
PA[1]	0x80140300 [1]	0x80140301 [1]	0x80140302 [1]	0x80140303 [1]	0x80140304 [1]	0x80140305 [1]	0x80140306 [1]
PA[2]	0x80140300 [2]	0x80140301 [2]	0x80140302 [2]	0x80140303 [2]	0x80140304 [2]	0x80140305 [2]	0x80140306 [2]
PA[3]	0x80140300 [3]	0x80140301 [3]	0x80140302 [3]	0x80140303 [3]	0x80140304 [3]	0x80140305 [3]	0x80140306 [3]
PA[4]	0x80140300 [4]	0x80140301 [4]	0x80140302 [4]	0x80140303 [4]	0x80140304 [4]	0x80140305 [4]	0x80140306 [4]
PA[5]	0x80140300 [5]	0x80140301 [5]	0x80140302 [5]	0x80140303 [5]	0x80140304 [5]	0x80140305 [5]	0x80140306 [5]
PA[6]	0x80140300 [6]	0x80140301 [6]	0x80140302 [6]	0x80140303 [6]	0x80140304 [6]	0x80140305 [6]	0x80140306 [6]

Pad	Input	IE	OEN	Output	Polarity	DS	Act as GPIO
PA[7]	0x80140300 [7]	0x80140301 [7]	0x80140302 [7]	0x80140303 [7]	0x80140304 [7]	0x80140305 [7]	0x80140306 [7]
PB[0]	0x80140308 [0]	0x80140309 [0]	0x8014030a [0]	0x8014030b [0]	0x8014030c [0]	0x8014030d [0]	0x8014030e [0]
PB[1]	0x80140308 [1]	0x80140309 [1]	0x8014030a [1]	0x8014030b [1]	0x8014030c [1]	0x8014030d [1]	0x8014030e [1]
PB[2]	0x80140308 [2]	0x80140309 [2]	0x8014030a [2]	0x8014030b [2]	0x8014030c [2]	0x8014030d [2]	0x8014030e [2]
PC[0]	0x80140310 [0]	ana_0xbd [0]	0x80140312 [0]	0x80140313 [0]	0x80140314 [0]	ana_0xbf [0]	0x80140316 [0]
PC[1]	0x80140310 [1]	ana_0xbd [1]	0x80140312 [1]	0x80140313 [1]	0x80140314 [1]	ana_0xbf [1]	0x80140316 [1]
PC[2]	0x80140310 [2]	ana_0xbd [2]	0x80140312 [2]	0x80140313 [2]	0x80140314 [2]	ana_0xbf [2]	0x80140316 [2]
PC[3]	0x80140310 [3]	ana_0xbd [3]	0x80140312 [3]	0x80140313 [3]	0x80140314 [3]	ana_0xbf [3]	0x80140316 [3]
PC[4]	0x80140310 [4]	ana_0xbd [4]	0x80140312 [4]	0x80140313 [4]	0x80140314 [4]	ana_0xbf [4]	0x80140316 [4]
PC[5]	0x80140310 [5]	ana_0xbd [5]	0x80140312 [5]	0x80140313 [5]	0x80140314 [5]	ana_0xbf [5]	0x80140316 [5]
PC[6]	0x80140310 [6]	ana_0xbd [6]	0x80140312 [6]	0x80140313 [6]	0x80140314 [6]	ana_0xbf [6]	0x80140316 [6]
PC[7]	0x80140310 [7]	ana_0xbd [7]	0x80140312 [7]	0x80140313 [7]	0x80140314 [7]	ana_0xbf [7]	0x80140316 [7]
PD[0]	0x80140318 [0]	ana_0xc0 [0]	0x8014031a [0]	0x8014031b [0]	0x8014031c [0]	ana_0xc2 [0]	0x8014031e [0]
PD[1]	0x80140318 [1]	ana_0xc0 [1]	0x8014031a [1]	0x8014031b [1]	0x8014031c [1]	ana_0xc2 [1]	0x8014031e [1]
PD[2]	0x80140318 [2]	ana_0xc0 [2]	0x8014031a [2]	0x8014031b [2]	0x8014031c [2]	ana_0xc2 [2]	0x8014031e [2]
PE[0]	0x80140320 [0]	0x80140321 [0]	0x80140322 [0]	0x80140323 [0]	0x80140324 [0]	0x80140325 [0]	0x80140326 [0]

Pad	Input	IE	OEN	Output	Polarity	DS	Act as GPIO
PE[1]	0x80140320 [1]	0x80140321 [1]	0x80140322 [1]	0x80140323 [1]	0x80140324 [1]	0x80140325 [1]	0x80140326 [1]
PE[2]	0x80140320 [2]	0x80140321 [2]	0x80140322 [2]	0x80140323 [2]	0x80140324 [2]	0x80140325 [2]	0x80140326 [2]
PE[3]	0x80140320 [3]	0x80140321 [3]	0x80140322 [3]	0x80140323 [3]	0x80140324 [3]	0x80140325 [3]	0x80140326 [3]
PE[4]	0x80140320 [4]	0x80140321 [4]	0x80140322 [4]	0x80140323 [4]	0x80140324 [4]	0x80140325 [4]	0x80140326 [4]
PE[5]	0x80140320 [5]	0x80140321 [5]	0x80140322 [5]	0x80140323 [5]	0x80140324 [5]	0x80140325 [5]	0x80140326 [5]
PE[6]	0x80140320 [6]	0x80140321 [6]	0x80140322 [6]	0x80140323 [6]	0x80140324 [6]	0x80140325 [6]	0x80140326 [6]
PE[7]	0x80140320 [7]	0x80140321 [7]	0x80140322 [7]	0x80140323 [7]	0x80140324 [7]	0x80140325 [7]	0x80140326 [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: 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: see section below

Table 10-3 GPIO Function Mux Configuration Registers

Address	Type	Description	Default Value
0x80140348	RW	[5:0]: function control bits of PA0_FS	0x00
0x80140349	RW	[5:0]: function control bits of PA1_FS	0x00
0x8014034a	RW	[5:0]: function control bits of PA2_FS	0x00
0x8014034b	RW	[5:0]: function control bits of PA3_FS	0x00
0x8014034c	RW	[5:0]: function control bits of PA4_FS	0x00
0x8014034d	R	[5:0]: function control bits of PA5_FS	0x00
0x8014034e	R	[5:0]: function control bits of PA6_FS	0x00

Address	Type	Description	Default Value
0x8014034f	R	[5:0]: function control bits of PA7_FS	0x00
0x80140350	RW	[5:0]: function control bits of PB0_FS	0x00
0x80140351	RW	[5:0]: function control bits of PB1_FS	0x00
0x80140352	RW	[5:0]: function control bits of PB2_FS	0x00
0x80140353	RW	[5:0]: function control bits of PB3_FS	0x00
0x80140354	RW	[5:0]: function control bits of PB4_FS	0x00
0x80140355	RW	[5:0]: function control bits of PB5_FS	0x00
0x80140356	RW	[5:0]: function control bits of PB6_FS	0x00
0x80140357	RW	[5:0]: function control bits of PB7_FS	0x00
0x80140358	RW	[5:0]: function control bits of PC0_FS	0x00
0x80140359	RW	[5:0]: function control bits of PC1_FS	0x00
0x8014035a	RW	[5:0]: function control bits of PC2_FS	0x00
0x8014035b	RW	[5:0]: function control bits of PC3_FS	0x00
0x8014035c	RW	[5:0]: function control bits of PC4_FS	0x00
0x8014035d	RW	[5:0]: function control bits of PC5_FS	0x00
0x8014035e	RW	[5:0]: function control bits of PC6_FS	0x00
0x8014035f	RW	[5:0]: function control bits of PC7_FS	0x00
0x80140360	RW	[5:0]: function control bits of PD0_FS	0x00
0x80140361	RW	[5:0]: function control bits of PD1_FS	0x00
0x80140362	RW	[5:0]: function control bits of PD2_FS	0x00
0x80140363	RW	[5:0]: function control bits of PD3_FS	0x00
0x80140364	RW	[5:0]: function control bits of PD4_FS	0x00
0x80140365	RW	[5:0]: function control bits of PD5_FS	0x00
0x80140366	RW	[5:0]: function control bits of PD6_FS	0x00
0x80140367	RW	[5:0]: function control bits of PD7_FS	0x00
0x80140368	R	[5:0]: function control bits of PE0_FS	0x00
0x80140369	R	[5:0]: function control bits of PE1_FS	0x00
0x8014036a	R	[5:0]: function control bits of PE2_FS	0x00

Address	Type	Description	Default Value
0x8014036b	R	[5:0]: function control bits of PE3_FS	0x00
0x8014036c	R	[5:0]: function control bits of PE4_FS	0x00
0x8014036d	R	[5:0]: function control bits of PE5_FS	0x00
0x8014036e	RW	[5:0]: function control bits of PE6_FS	0x00
0x8014036f	RW	[5:0]: function control bits of PE7_FS	0x00
0x80140370	RW	[5:0]: function control bits of PF0_FS	0x00
0x80140371	RW	[5:0]: function control bits of PF1_FS	0x00
0x80140372	RW	[5:0]: function control bits of PF2_FS	0x00
0x80140373	RW	[5:0]: function control bits of PF3_FS	0x00
0x80140374	RW	[5:0]: function control bits of PF4_FS	0x00
0x80140375	RW	[5:0]: function control bits of PF5_FS	0x00
0x80140376	RW	[5:0]: function control bits of PF6_FS	0x00
0x80140377	RW	[5:0]: function control bits of PF7_FS	0x00

10.1.2 Connection Relationship between GPIO and Related Modules

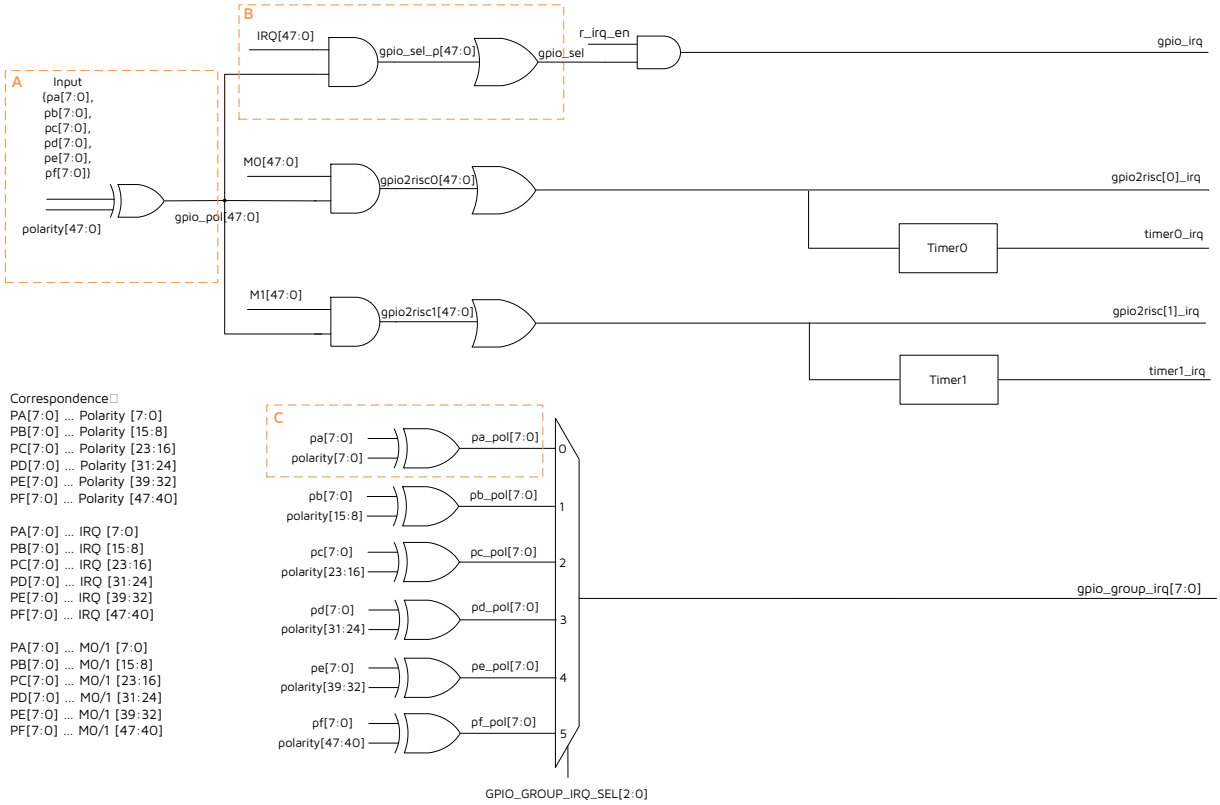
GPIO can be used to generate GPIO interrupt signal for interrupt system, counting or control signal for Timer/ Counter module, or gpio2risc interrupt signal for interrupt system.

For the "Exclusive Or (XOR)" operation result for input signal from any GPIO pin 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].

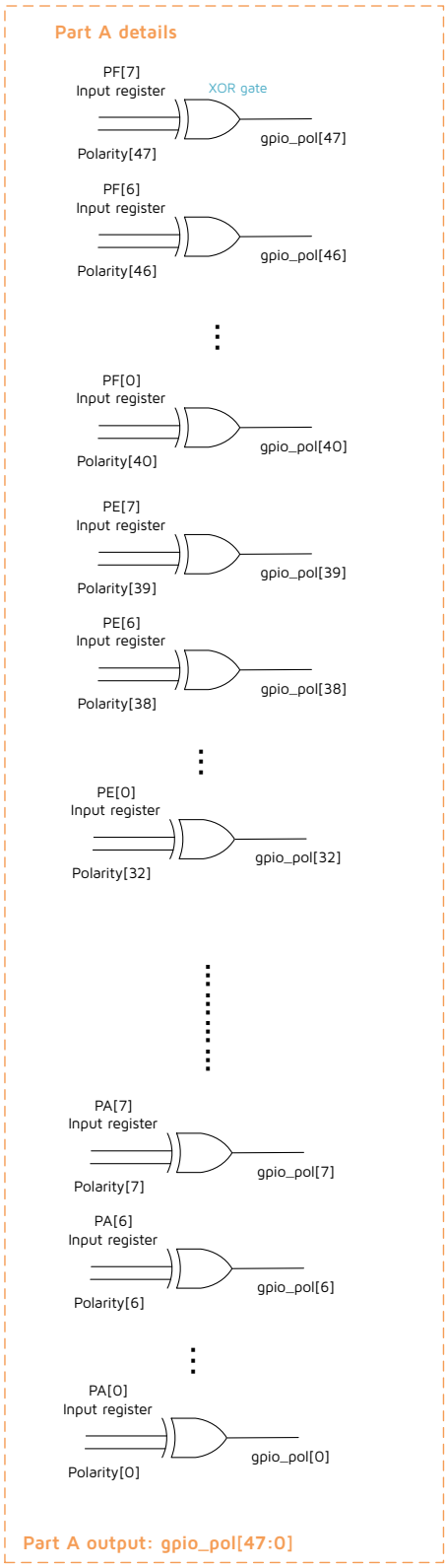
The logic relationship is shown in figure below.

Figure 10-1 Logic Relationship between GPIO and Related Modules



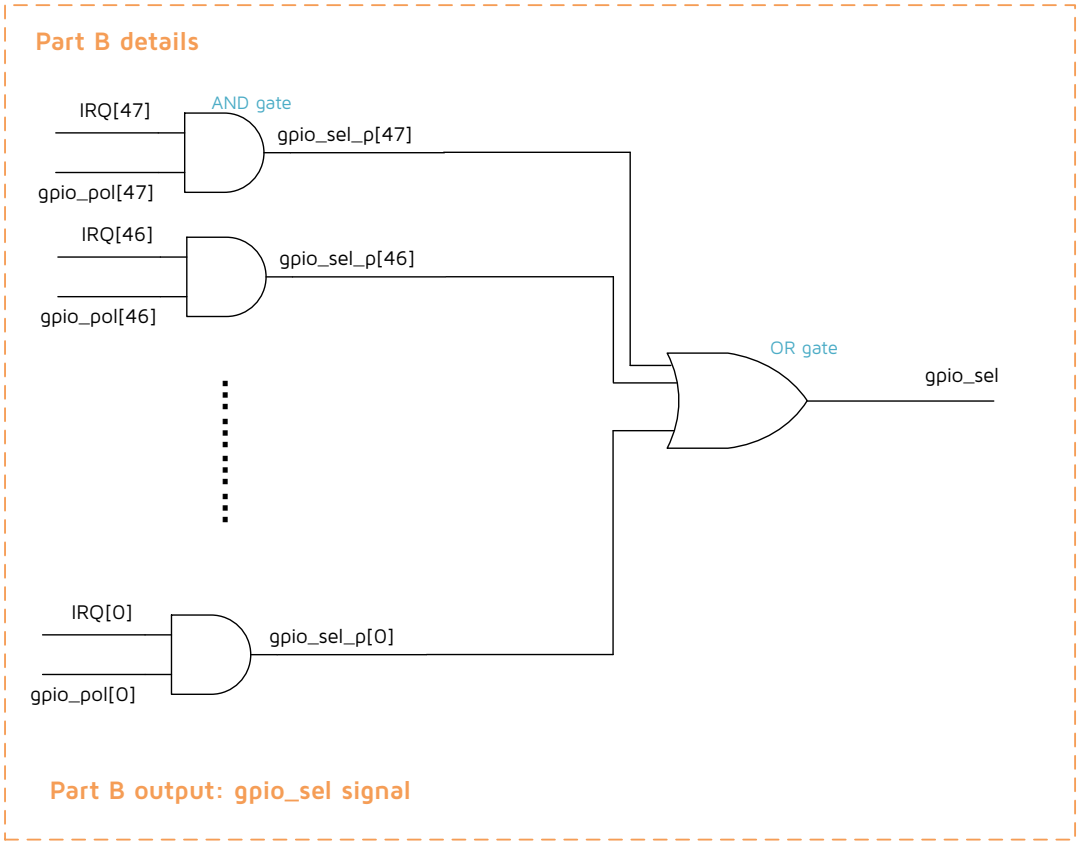
In the figure above, the detailed digital circuit for part A is shown as below. The input register and polarity register of each corresponding GPIO pin take XOR operation to get one gpio_pol signal. The output of this part is gpio_pol[47:0] which is a set of 48 bits signals.

Figure 10-2 Detailed Circuit for Part A



The detailed digital circuit for part B is shown as below. The IRQ register and gpio_pol 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_sel signal. The output of this part is gpio_sel which is the 1 bit signal.

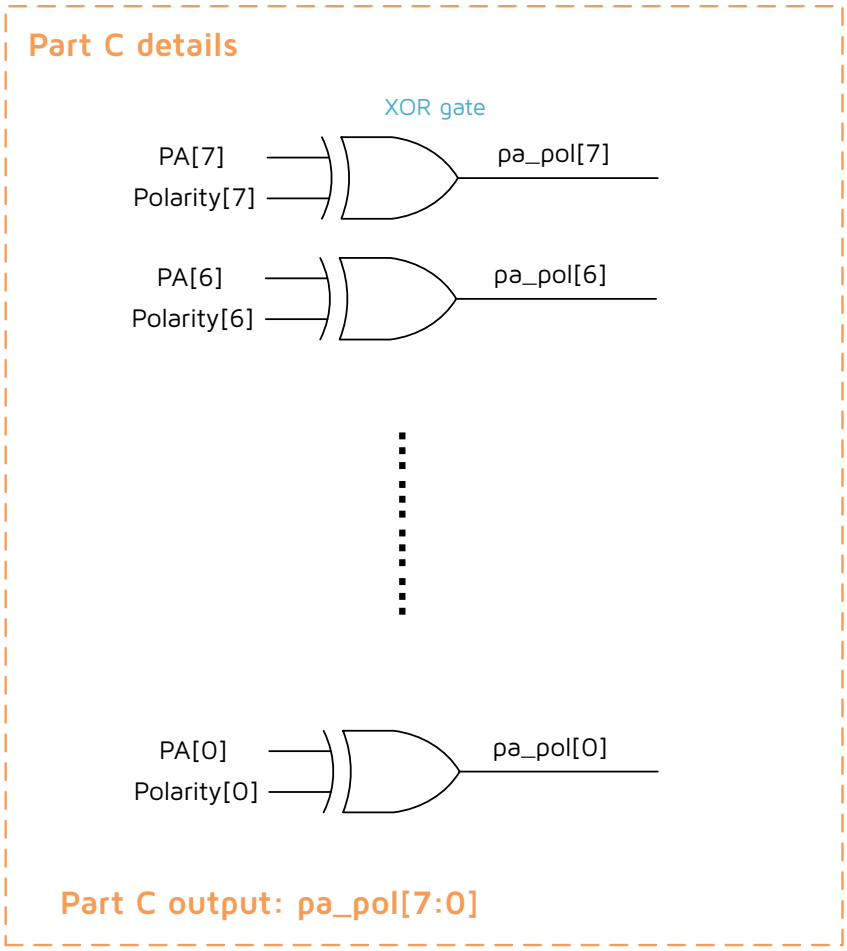
Figure 10-3 Detailed Circuit for Part B



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 is shown as below. The output of this part is pa_pol[7:0] which is a set of 8 bits signals.

Figure 10-4 Detailed Circuit of Part C



Similarly, the output of the circuits below part C are pb_pol[7:0], pc_pol[7:0], pd_pol[7:0],.pe_pol[7:0], and pf_pol[7:0]. The 6 groups of px_pol signals mux with GPIO_GROUP_IRQ_SEL[2:0] to get gpio_irq_group signals.

Please note that gpio_group_irq[7] is the result from pa_pol[7], pb_pol[7], pc_pol[7], pd_pol[7], pe_pol[7], pf_pol[7] and GPIO_GROUP_IRQ_SEL[2:0], and similarly, gpio_group_irq[6] is the result from pa_pol[6], pb_pol[6], pc_pol[6], pd_pol[6], pe_pol[6], pf_pol[6] and GPIO_GROUP_IRQ_SEL[2:0], and so on. This is different from the above interrupt request signals shown in Figure 10-1 Logic Relationship between GPIO and Related Modules, they are the operation results from all 48 GPIOs.

10.1.3 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 will take 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.

- High drive strength (suitable for GPIO pins PA[5:7], PC[0:7], PD[0:7], PE[0:1], PE[4:7])
  - "DS" = 1, maximum drive strength
  - "DS" = 0, minimum drive strength

- Standard drive strength (suitable for GPIO pins PA[0:4], PB[0:7], PE[2:3], PF[0:7] and PG[0:5])
  - "DS" = 1, maximum drive strength
  - "DS" = 0, minimum drive strength

The detailed current data of GPIO drive strength refers to Table 2-5 GPIO Drive Strength.

10.1.4 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.

10.1.5 GPIO IRQ Signal

Select GPIO interrupt trigger edge (positive edge or negative edge) via configuring "Polarity", and set corresponding GPIO interrupt enabling bit "Irq".

10.1.6 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 10-4 GPIO IRQ Table

Pad	Input	IRQ	M0	M1	Polarity
PA[0]	0x80140300[0]	0x80140307[0]	0x80140338[0]	0x80140340[0]	0x80140304[0]
PA[1]	0x80140300[1]	0x80140307[1]	0x80140338[1]	0x80140340[1]	0x80140304[1]
PA[2]	0x80140300[2]	0x80140307[2]	0x80140338[2]	0x80140340[2]	0x80140304[2]
PA[3]	0x80140300[3]	0x80140307[3]	0x80140338[3]	0x80140340[3]	0x80140304[3]
PA[4]	0x80140300[4]	0x80140307[4]	0x80140338[4]	0x80140340[4]	0x80140304[4]
PA[5]	0x80140300[5]	0x80140307[5]	0x80140338[5]	0x80140340[5]	0x80140304[5]
PA[6]	0x80140300[6]	0x80140307[6]	0x80140338[6]	0x80140340[6]	0x80140304[6]
PA[7]	0x80140300[7]	0x80140307[7]	0x80140338[7]	0x80140340[7]	0x80140304[7]
PB[0]	0x80140308[0]	0x8014030f[0]	0x80140339[0]	0x80140341[0]	0x8014030c[0]
PB[1]	0x80140308[1]	0x8014030f[1]	0x80140339[1]	0x80140341[1]	0x8014030c[1]
PB[2]	0x80140308[2]	0x8014030f[2]	0x80140339[2]	0x80140341[2]	0x8014030c[2]
PC[0]	0x80140310[0]	0x80140317[0]	0x8014033a[0]	0x80140342[0]	0x80140314[0]
PC[1]	0x80140310[1]	0x80140317[1]	0x8014033a[1]	0x80140342[1]	0x80140314[1]

Pad	Input	IRQ	M0	M1	Polarity
PC[2]	0x80140310[2]	0x80140317[2]	0x8014033a[2]	0x80140342[2]	0x80140314[2]
PC[3]	0x80140310[3]	0x80140317[3]	0x8014033a[3]	0x80140342[3]	0x80140314[3]
PC[4]	0x80140310[4]	0x80140317[4]	0x8014033a[4]	0x80140342[4]	0x80140314[4]
PC[5]	0x80140310[5]	0x80140317[5]	0x8014033a[5]	0x80140342[5]	0x80140314[5]
PC[6]	0x80140310[6]	0x80140317[6]	0x8014033a[6]	0x80140342[6]	0x80140314[6]
PC[7]	0x80140310[7]	0x80140317[7]	0x8014033a[7]	0x80140342[7]	0x80140314[7]
PD[0]	0x80140318[0]	0x8014031f[0]	0x8014033b[0]	0x80140343[0]	0x8014031c[0]
PD[1]	0x80140318[1]	0x8014031f[1]	0x8014033b[1]	0x80140343[1]	0x8014031c[1]
PD[2]	0x80140318[2]	0x8014031f[2]	0x8014033b[2]	0x80140343[2]	0x8014031c[2]
PE[0]	0x80140320[0]	0x80140327[0]	0x8014033c[0]	0x80140344[0]	0x80140324[0]
PE[1]	0x80140320[1]	0x80140327[1]	0x8014033c[1]	0x80140344[1]	0x80140324[1]
PE[2]	0x80140320[2]	0x80140327[2]	0x8014033c[2]	0x80140344[2]	0x80140324[2]
PE[3]	0x80140320[3]	0x80140327[3]	0x8014033c[3]	0x80140344[3]	0x80140324[3]
PE[4]	0x80140320[4]	0x80140327[4]	0x8014033c[4]	0x80140344[4]	0x80140324[4]
PE[5]	0x80140320[5]	0x80140327[5]	0x8014033c[5]	0x80140344[5]	0x80140324[5]
PE[6]	0x80140320[6]	0x80140327[6]	0x8014033c[6]	0x80140344[6]	0x80140324[6]
PE[7]	0x80140320[7]	0x80140327[7]	0x8014033c[7]	0x80140344[7]	0x80140324[7]

10.1.7 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".

10.1.8 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_0x0e<7:0> ~ afe_0x17<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 10-5 Analog Registers for Pull-up/Pull-down Resistor Control

Address	Type	Description	Default Value
0x0e	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
0x0f	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
0x10	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
0x11	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
0x12	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
0x13	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

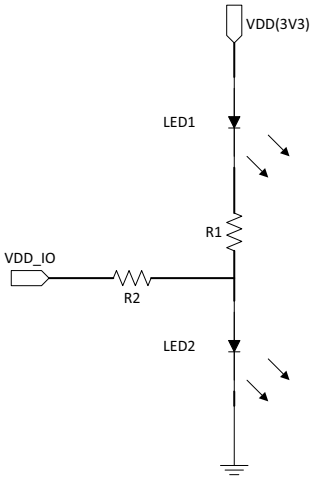
Address	Type	Description	Default Value
0x14	R/W	GPIO_D<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x15	R/W	GPIO_D<7:4> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x16	R/W	GPIO_E<3:0> pull up and down select: 00: Null 01: 1M pull up 10: 100K pull down 11: 10K pull up	00000000
0x17	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

10.1.9 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 10-5 One GPIO drives two LEDs



10.2 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 10-6 Swire Related Registers

Offset	Type	Description	Default Value
0x00	R	SWIRE_DATA [7:0] swire_data	0x00
0x01	RW	SWIRE_CTL [0]:swire_wr [1]:swire_rd [2]:swire_cmd [4]:swire_eop [6]:swire_usb_det [7]:swire_usb_en	0x80
0x02	RW	SWIRE_CTL2 [6:0]: swire_clk_div	0x05

Offset	Type	Description	Default Value
0x03	RW	SWIRE_ID [4:0] id_valid [7] fifo_mode	0x00

10.3 I2C

The SoC embeds I2C hardware module, which could act as Master mode or Slave mode. I2C is a popular inter-IC interface requiring only 2 bus lines, a serial data line (SDA) and a serial clock line (SCL).

There are two channels of I2C signals in this chip. When the multiplexed GPIO pins are configured as I2C_SDA and I2C_SCL, they can work in Master and Slave mode; when the multiplexed GPIO pins are configured as I2C1_SCL and I2C1_SDA, they can work in Master mode only.

10.3.1 Communication Protocol

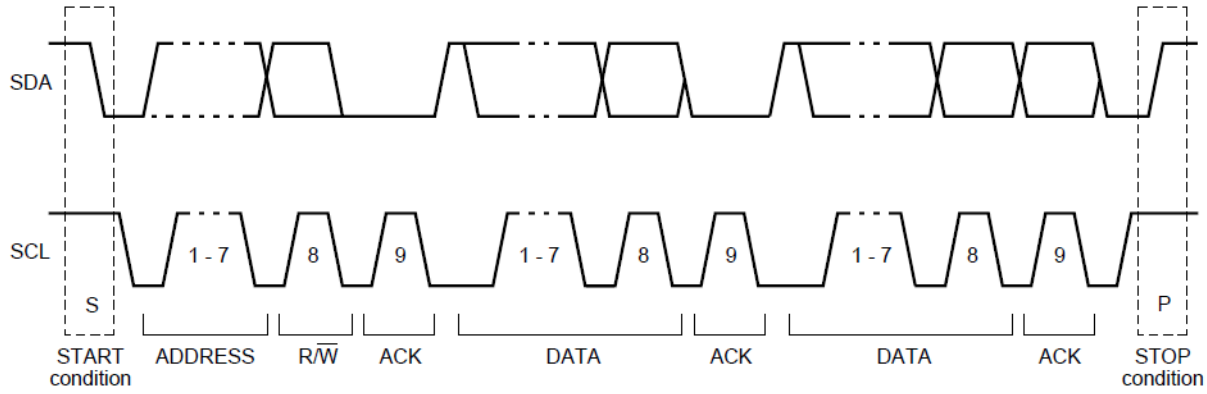
Telink I2C module supports standard mode (100kbps) and Fast-mode (400kbps) with restriction that system clock must be by at least 10x of data rate.

Two wires, SDA and SCL (SCK) carry information between Master device and Slave device connected to the bus. Each device is recognized by 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 by a Master.

Both SDA and SCL are bidirectional lines connected to a positive supply voltage via a pull-up resister. It's recommended to use external 3.3kOhm 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 free, both lines are HIGH. It's noted that data in SDA line must keep stable when clock signal in SCL line is at high level, and level state in SDA line is only allowed to change when clock signal in SCL line is at low level.

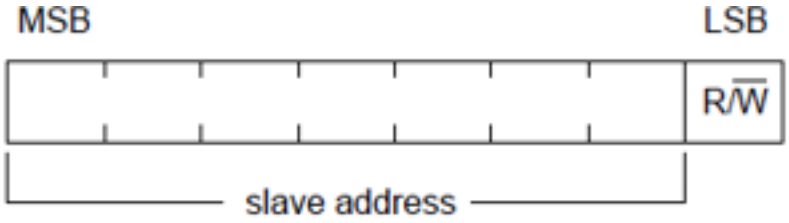
Figure 10-6 I2C Timing



10.3.2 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 10-7 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 will reply with ACK automatically. To monitor the start of I2C transaction, user could set interrupt from GPIO for SCA or SCL.

Read and write format of Slave modes are shown as below.

DMA and NDMA access buffer through dma and ahb, respectively.

Figure 10-8 Read Format in Slave Mode

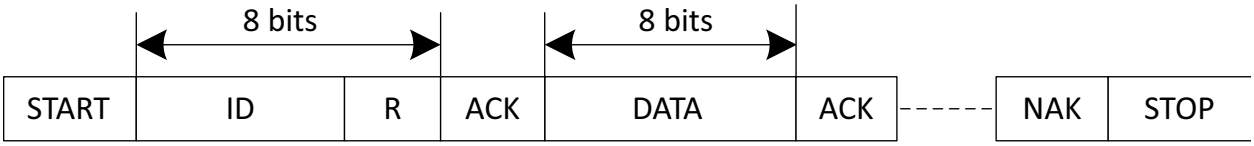
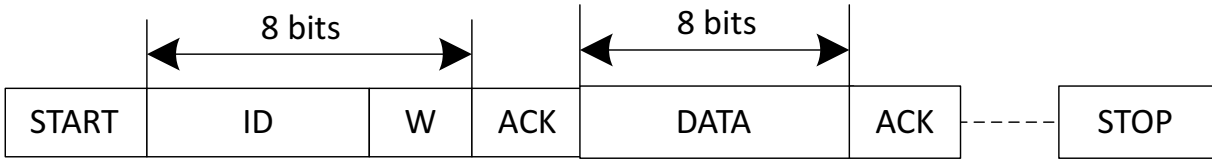


Figure 10-9 Write Format in Slave Mode



10.3.3 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} = (System Clock / (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 [7:1].

I2C Master could send START, Slave Address, R/W bit, data and STOP cycle by configuring SLAVE_STRECH_EN. I2C master will send enabled cycles in the correct sequence.

Register I2CMST serves to indicate whether Master/Master packet is busy, as well as Master received status. Bit[0] will be set to 1 when one byte is being sent, 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 signal is sent, and the bit will be automatically cleared after the stop signal is sent. Bit[2] indicates whether to succeed in sending acknowledgement signal.

10.3.3.1 I2C Master Write Transfer in NDMA Mode

I2C Master has 8-byte buffer for write data, which are I2C_data_buf0, I2C_data_buf1, I2C_data_buf2 and I 2C_data_buf3. Write transfer will be completed by I2C master module.

For example, to implement an I2C write transfer with 4-byte data, which contains START, Slave Address, 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], 1st byte address to buff. To start I2C write transfer, I2CSCT1 is configured to 0x13 (0001 0011). I2C Master will launch START, Slave address. 1 word data to buff; I2CSCT1 is configured to 0x24 (0000 0024).Write bit, load ACK to I2CMST[2], send buff data, load ACK to I2CMST[2] and then STOP sequentially.

I2C supports a single write of maximum length supported for a single DMA transfer.

10.3.3.2 I2C Master Read Transfer in NDMA Mode

I2C Master has 8 byte buffer for read data, which is fifo (0x08). Read transfer will be completed by I2C Master.

For example, to implement an I2C read transfer with 4 byte data, which contains START, Slave Address, Read bit, ACK from Slave, 4 byte from Slave, ACK by master and STOP, user needs to configure I2C slave address to I2C_ID[7:1]. To start I2C read transfer, I2CSCT1 is configured to 0x7b (0111 1011). I2C Master will launch START, Slave address, Read bit, load ACK to I2CMST[2], load data to I2CDR, reply ACK and then STOP sequentially.

I2C supports a single read of maximum length supported for a single DMA transfer.

10.3.3.3 I2C Master Writer Transfer in DMA Mode

The data to be sent is put into SRAM, set tx dma config, user needs to configure I2C slave address to I2C_ID [7:1], 1st byte address to buff. To start I2C write transfer, I2CSCT1 is configured to 0x13 (0001 0011). I2C Master will launch START, Slave address. 1 word data to buff, I2CSCT1 is configured to 0x24 (0000 0024). Write bit, load ACK to I2CMST[2], send buff data, load ACK to I2CMST[2] and then STOP sequentially.

I2C supports a single write of maximum length supported for a single DMA transfer.

10.3.3.4 I2C Master Read Transfer in DMA Mode

For example, set rx dma config, user needs to configure I2C slave address to I2C_ID[7:1]. To start I2C read transfer, I2CSCT1 is configured to 0x7b (0111 1011). I2C Master will launch START, Slave address, Read bit, load ACK to I2CMST[2], load data to I2CDR, reply ACK and then STOP sequentially.

I2C supports a single write of maximum length supported for a single DMA transfer.

10.3.4 Register Description

The I2C related registers are listed as following, the base address of the following registers is 0x80140280.

Table 10-7 I2C Related Registers

Address Offset	Type	Description	Default Value
0x00	RW	I2CSP I2C master clock speed: pclk*1000*1000/(4*I2C_CLK_SPEED)	0x1f

Address Offset	Type	Description	Default Value
0x01	RW	I2C_ID I2C id: [7:1] I2C slave address + [0] R/W flag bit	0x5c
0x02	VOLATILE/R	I2CMST [0]: master busy (volatile) [1]: master packet busy (volatile) [2]: master received status: 1 for nak; 0 for ack (volatile) [5:3]: 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]: slave state of the base:0-ss_id, 1-ss_dataw, 2-ss_datar (R)	0x30
0x03	RW	I2CSCT0 [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
0x04	RW	I2CSCT1 [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	RW	I2CTRIG [3:0]: rx_irq_trig level [7:4]: tx_irq_trig level	0x44

Address Offset	Type	Description	Default Value
0x06	RW	I2CCTRL2 [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]: hold0	0x00
0x07	RW	I2CCTRL3 [0]: r_clk_stretch_sen, slave auto stretch clk enable [1]: hold1 [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	RW	I2C_DATA_BUF0 Write/read buffer[7:0]	0x00
0x09	RW	I2C_DATA_BUF1 Write/read buffer[15:8]	0x00
0x0a	RW	I2C_DATA_BUF2 Write/read buffer[23:16]	0x00
0x0b	RW	I2C_DATA_BUF3 Write/read buffer[31:24]	0x00

Address Offset	Type	Description	Default Value
0x0c	VOLATILE	I2C_BUFCNT [3:0]: rx_buf_cnt [7:4]: tx_buf_cnt	0x00
0x0d	VOLATILE	I2C_STATUS [2:0]: rbcnt [0] W1C, manual_stretch_clr(manual clear slave stretch); [1] W1C, slave manul stretch clk trig; [2] W1C, manual_clr_mst_ack [3]: i2c_irq_o [6:4]: wbcnt [7]: rxdone	0x00
0x0e	VOLATILE	I2C_STATUS1 [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	W1C	I2C_CLR [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	RW	I2CLENL Config buffer send and receive byte number (low): default 1 byte	0x01
0x11	RW	I2CLENM Config buffer send and receive byte number (middle): default 0 byte	0x00

Address Offset	Type	Description	Default Value
0x12	RW	I2CLENH Config buffer send and receive byte number (high): default 0 byte	0x00
0x13	R	I2C_MST_STATUS [0]: id_busy, nak function id flag [1]: addr_busy, nak function address flag [2]: dataw_busy, nak function wdata flag [3]: datar_busy, nak function rdata flag	0x00

10.4 Memory SPI

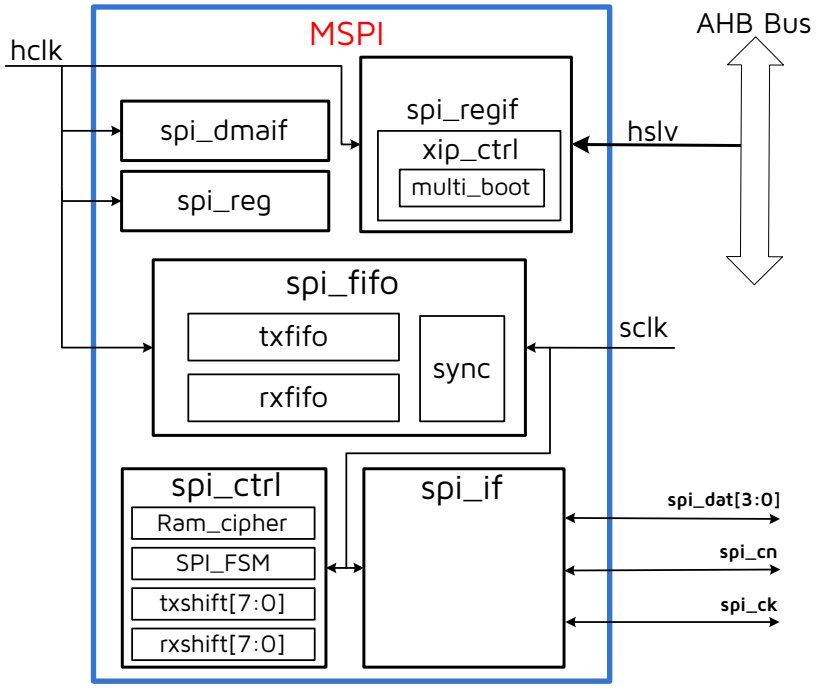
10.4.1 Memory SPI Diagram

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 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 10-10 Memory SPI Diagram



10.4.2 MSPI Register Description

Memory SPI related registers are listed as below. The base address of the following registers is 0xA3FFFF00.

Table 10-8 Memory SPI Related Registers

Offset	Type	Description	Default Value
0x00	RW	MSPI_WR_RD_DATA0 [7:0]: data[7:0] to transmit or received	0x00
0x01	RW	MSPI_WR_RD_DATA1 [7:0]: data[15:8] to transmit or received	0x00
0x02	RW	MSPI_WR_RD_DATA2 [7:0]: data[23:16] to transmit or received	0x00
0x03	RW	MSPI_WR_RD_DATA3 [7:0]: data[31:24] to transmit or received	0x00
0x04	RW	MSPI_CMD [7:0]: SPI Command	0x00

Offset	Type	Description	Default Value
0x05	RW	MSPI_CTRL0 [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
0x07	RW	MSPI_TIMING [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	0x09
0x08	RW	MSPI_CTRL1 [1:0] data_lane, data lane, MASTER ONLY 0: single, 1: dual, 2: quad, 3: quad [3:2] addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes, MASTER ONLY [4] addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase(Dual/Quad), MASTER ONLY [5] addr_en, 1:enabel addr phase, MASTER ONLY [6] cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase(Dual/Quad), MASTER ONLY [7] cmd_en, the spi commnd phase enable, MASTER ONLY	0xa9

Offset	Type	Description	Default Value
0x09	RW	MSPI_CTRL2 [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~0xf:reserved	0x77
0x0c	RW	MSPI_ADDR0 [7:0] spi_addr0, spi address byte0, MASTER ONLY	0x0
0x0d	RW	MSPI_ADDR1 [7:0] spi_addr1, spi address byte1, MASTER ONLY	0x0
0X0e	RW	MSPI_ADDR2 [7:0] spi_addr2, spi address byte2, MASTER ONLY	0x0
0x0f	RW	MSPI_ADDR3 [7:0] spi_addr3, spi address byte3, MASTER ONLY	0x0
0x10	RW	MSPI_TX_CNT0 [7:0] tx_cnt0, transfer count for write data, byte0	0x00
0x11	RW	MSPI_TX_CNT1 [7:0] tx_cnt1, transfer count for write data, byte1	0x00
0x12	RW	MSPI_TX_CNT2 [7:0] tx_cnt2, transfer count for write data, byte2	0x00
0x14	RW	MSPI_RX_CNT0 [7:0] rx_cnt0, transfer count for read data, byte0	0x00

Offset	Type	Description	Default Value
0x15	RW	MSPI_RX_CNT1 [7:0] rx_cnt1, transfer count for read data, byte1	0x00
0x16	RW	MSPI_RX_CNT2 [7:0] rx_cnt2, transfer count for read data, byte2	0x00
0x18	RW	MSPI_CTRL3 [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	RW	MSPI_TXFIFO_THRES [5:0] txfifo threshold	0x00
0x1a	RW	MSPI_RXFIFO_THRES [5:0] rxfifo threshold	0x00
0x1c	RW	MSPI_CTRL4 [0] xip_page_mode_en, xip page mode enable [1] xip_timeout_mode_en, 0:xip timeout disable 1:xip timeout enable [2] xip_stop, stop xip [3] xip_enable, enable xip [4] dummy_cnt_add	0x0a
0x1d	RW	MSPI_XIP_PAGE_SIZE [7:0] page_size, page boundary size = 2^page_size	0x00
0x1e	RW	MSPI_XIP_TIMEOUT_CNT [7:0] timeout_cnt, timeout time sel	0x20
0x20	RW	MSPI_SET_L [2:0] mspi_set_l, multiboot address offset option, 0:0k; 1:128k; 2:256k; 4:512k	0x00
0x21	RW	MSPI_SET_H [6:0] mspi_set_h, program space size = mspi_set_h*4k	0x00

Offset	Type	Description	Default Value
0x22	RW	MSPI_XIP_ADDR_OFFSET [7:0] xip_addr_offset, address offset = xip_addr_offset << 24	0x00
0x24	R	MSPI_TXFIFO_STATUS [6:0] txfifo_entries [7] txfifo_full	0x0
0x25	R	MSPI_RXFIFO_STATUS [6:0] rxfifo_entries [7] rxfifo_empty	0x0
0x28	RW/R	MSPI_STATUS [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	W1C	MSPI_INT_STATUS0 [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
0x90	RW	MSPI_XIP_RD_FMT [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	RW	MSPI_XIP_RD_TRANSMODE [3:0] xip0_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip0_rd_transmode	0x97

Offset	Type	Description	Default Value
0x93	RW	MSPI_XIP_RD_CMD [7:0] xip0_rd_cmd, read command used for xip	0x3b
0x94	RW	MSPI_XIP_WR_FMT [0] xip0_wr_data_dual, spi dual I/O mode, MASTER ONLY [1] xip0_wr_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	MSPI_XIP_WR_TRANSMODE [7:4] xip0_wr_transmode	0x10
0x97	RW	MSPI_XIP_WR_CMD [7:0] xip0_wr_cmd, write command used for xip	0x02

10.5 LSPI

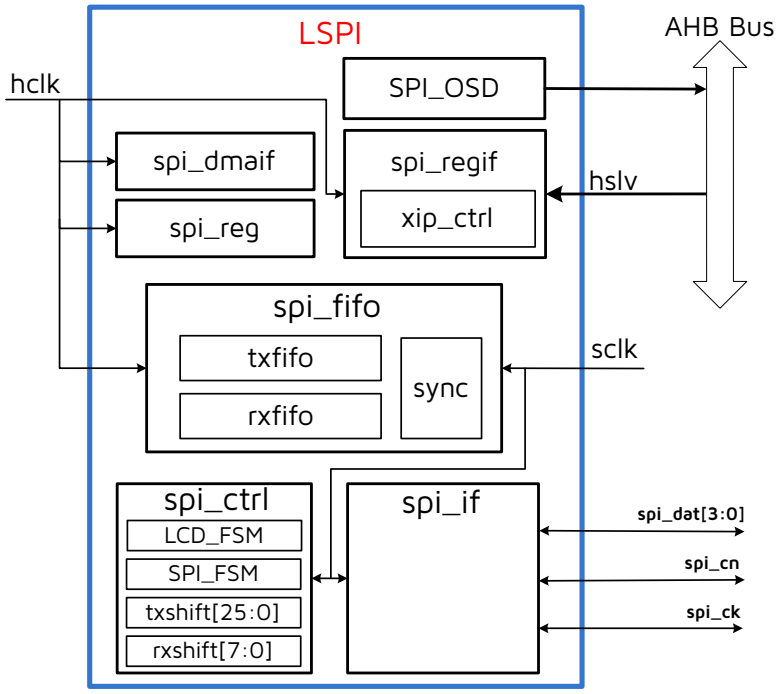
10.5.1 LSPI Diagram

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 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 10-11 LSPI Diagram



10.5.2 LSPI Register Description

The LSPI related registers are listed as below. The base address of the following registers is 0x87FFFF00.

Table 10-9 LSPI Related Registers

Offset	Type	Description	Default Value
0x00	RW	LSPI_WR_RD_DATA0 [7:0]: data[7:0] to transmit or received	0x00
0x01	RW	LSPI_WR_RD_DATA1 [7:0]: data[15:8] to transmit or received	0x00
0x02	RW	LSPI_WR_RD_DATA2 [7:0]: data[23:16] to transmit or received	0x00
0x03	RW	LSPI_WR_RD_DATA3 [7:0]: data[31:24] to transmit or received	0x00
0x04	RW	LSPI_CMD [7:0]: SPI Command	0x00

Offset	Type	Description	Default Value
0x05	RW	LSPI_CTRL0 [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
0x07	RW	LSPI_TIMING [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	0x09
0x08	RW	LSPI_CTRL1 [1:0] data_lane, data lane, MASTER ONLY 0: single, 1: dual, 2: quad, 3: quad [3:2] addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes, MASTER ONLY [4] addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase(Dual/Quad), MASTER ONLY [5] addr_en, 1:enabel addr phase, MASTER ONLY [6] cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase(Dual/Quad), MASTER ONLY [7] cmd_en, the spi commnd phase enable, MASTER ONLY	0xa9

Offset	Type	Description	Default Value
0x09	RW	LSPI_CTRL2 [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~0xf:reserved	0x77
0x0c	RW	LSPI_ADDR0 [7:0] spi_addr0, spi address byte0/lcd_porch_line_time[7:0]	0x0
0x0d	RW	LSPI_ADDR1 [7:0] spi_addr1, spi address byte1/lcd_porch_line_time[15:8]	0x0
0x0e	RW	LSPI_ADDR2 [7:0] spi_addr2, spi address byte2/lcd_display_line_time[7:0]	0x0
0x0f	RW	LSPI_ADDR3 [7:0] spi_addr3, spi address byte3/lcd_display_line_time[15:8]	0x0
0x10	RW	LSPI_TX_CNT0 [7:0] tx_cnt0, transfer count for write data, byte0/lcd_pixel_per_line[7:0]	0x00
0x11	RW	LSPI_TX_CNT1 [7:0] tx_cnt1, transfer count for write data, byte1/ {lcd_line_per_frame[5:0], lcd_pixel_per_line[9:8]}	0x00
0x12	RW	LSPI_TX_CNT2 [7:0] tx_cnt2, transfer count for write data, byte2/{4'h0, lcd_line_per_frame[9:6]}	0x00

Offset	Type	Description	Default Value
0x14	RW	LSPI_RX_CNT0 [7:0] rx_cnt0, transfer count for read data, byte0	0x00
0x15	RW	LSPI_RX_CNT1 [7:0] rx_cnt1, transfer count for read data, byte1	0x00
0x16	RW	LSPI_RX_CNT2 [7:0] rx_cnt2, transfer count for read data, byte2	0x00
0x18	RW	LSPI_CTRL3 [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	RW	LSPI_TXFIFO_THRES [5:0] txfifo threshold	0x00
0x1a	RW	LSPI_RXFIFO_THRES [5:0] rxfifo threshold	0x00
0x1c	RW	LSPI_CTRL4 [0] xip_page_mode_en, xip page mode enable [1] xip_timeout_mode_en, 0:xip timeout disable 1:xip timeout enable [2] xip_stop, stop xip [3] xip_enable, enable xip [4] dummy_cnt_add	0x0a
0x1d	RW	LSPI_XIP_PAGE_SIZE [7:0] page_size, page boundary size = 2^page_size	0x00
0x1e	RW	LSPI_XIP_TIMEOUT_CNT [7:0] timeout_cnt, timeout time sel	0x20
0x22	RW	LSPI_XIP_ADDR_OFFSET [7:0] xip_addr_offset, address offset = xip_addr_offset << 24	0x00

Offset	Type	Description	Default Value
0x24	R	LSPI_TXFIFO_STATUS [6:0] txfifo_entries [7] txfifo_full	0x0
0x25	R	LSPI_RXFIFO_STATUS [6:0] rxfifo_entries [7] rxfifo_empty	0x0
0x28	RW/R	LSPI_STATUS [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	0x0
0x2a	W1C	LSPI_INT_STATUS0 [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	W1C	LSPI_INT_STATUS1 [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
0x2f	RW	LSPI_LCD_CTRL2 [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	0x00

Offset	Type	Description	Default Value
0x30	RW	LSPI_LCD_CTRL [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	RW	LSPI_LCD_VBP_CNT [7:0] lcd_vbp_cnt, lcd vertical porch line number, actual num=lcd_vbp_cnt	0x00
0x32	RW	LSPI_LCD_VFP_CNT [7:0] lcd_vfp_cnt, lcd front porch line number, actual num=lcd_vbp_cnt	0x00
0x33	RW	LSPI_LCD_LINE_LVL [7:0] lcd_line_lvl, lcd line threshold to trig interrupt	0x00
0x34	RW	LSPI_LCD_BIMAGE_ADDR0 [7:5] lcd_bimage_start_addr0, background image data start address byte0	0x00
0x35	RW	LSPI_LCD_BIMAGE_ADDR1 [7:0] lcd_bimage_start_addr1, background image data start address byte1	0x00
0x36	RW	LSPI_LCD_BIMAGE_ADDR2 [7:0] lcd_bimage_start_addr2, background image data start address byte2	0x00
0x37	RW	LSPI_LCD_BIMAGE_ADDR3 [6:0] lcd_bimage_start_addr3, background image data start address byte3	0x00
0x38	RW	LSPI_LCD_FIMAGE_ADDR0 [7:4] lcd_fimage_start_addr0, front image data start address byte0	0x00
0x39	RW	LSPI_LCD_FIMAGE_ADDR1 [7:0] lcd_fimage_start_addr0, front image data start address byte1	0x00
0x3a	RW	LSPI_LCD_FIMAGE_ADDR2 [7:0] lcd_fimage_start_addr0, front image data start address byte2	0x00
0x3b	RW	LSPI_LCD_FIMAGE_ADDR2 [6:0] lcd_fimage_start_addr0, front image data start address byte3	0x00
0x3e	R	LSPI_LCD_LINE_CNT0 [7:0] lcd_line_cnt_l, lcd_line_cnt[7:0]	0x00

Offset	Type	Description	Default Value
0x3f	R	LSPI_LCD_LINE_CNT1 [1:0] lcd_line_cnt_l, lcd_line_cnt[9:8]	0x00
0x40	RW	LSPI_LCD_LUT_DATA0_BYTE0 [7:0] lcd_lut_data0_byte0, lcd lut address0 data byte0	0x00
0x41	RW	LSPI_LCD_LUT_DATA0_BYTE1 [7:0] lcd_lut_data0_byte1, lcd lut address0 data byte1	0x00
0x42	RW	LSPI_LCD_LUT_DATA0_BYTE2 [7:0] lcd_lut_data0_byte2, lcd lut address0 data byte2	0x00
0x44	RW	LSPI_LCD_LUT_DATA1_BYTE0 [7:0] lcd_lut_data1_byte0, lcd lut address1 data byte0	0x00
0x45	RW	LSPI_LCD_LUT_DATA1_BYTE1 [7:0] lcd_lut_data1_byte1, lcd lut address1 data byte1	0x00
0x46	RW	LSPI_LCD_LUT_DATA1_BYTE2 [7:0] lcd_lut_data1_byte2, lcd lut address1 data byte2	0x00
0x48	RW	LSPI_LCD_LUT_DATA2_BYTE0 [7:0] lcd_lut_data2_byte0, lcd lut address2 data byte0	0x00
0x49	RW	LSPI_LCD_LUT_DATA2_BYTE1 [7:0] lcd_lut_data2_byte0, lcd lut address2 data byte1	0x00
0x4a	RW	LSPI_LCD_LUT_DATA2_BYTE2 [7:0] lcd_lut_data2_byte0, lcd lut address2 data byte2	0x00
0x4c	RW	LSPI_LCD_LUT_DATA3_BYTE0 [7:0] lcd_lut_data3_byte0, lcd lut address3 data byte0	0x00
0x4d	RW	LSPI_LCD_LUT_DATA3_BYTE1 [7:0] lcd_lut_data3_byte0, lcd lut address3 data byte1	0x00
0x4e	RW	LSPI_LCD_LUT_DATA3_BYTE2 [7:0] lcd_lut_data3_byte0, lcd lut address3 data byte2	0x00
0x50	RW	LSPI_LCD_LUT_DATA4_BYTE0 [7:0] lcd_lut_data4_byte0, lcd lut address4 data byte0	0x00
0x51	RW	LSPI_LCD_LUT_DATA4_BYTE1 [7:0] lcd_lut_data4_byte0, lcd lut address4 data byte1	0x00

Offset	Type	Description	Default Value
0x52	RW	LSPI_LCD_LUT_DATA4_BYTE2 [7:0] lcd_lut_data4_byte0, lcd lut address4 data byte2	0x00
0x54	RW	LSPI_LCD_LUT_DATA5_BYTE0 [7:0] lcd_lut_data5_byte0, lcd lut address5 data byte0	0x00
0x55	RW	LSPI_LCD_LUT_DATA5_BYTE1 [7:0] lcd_lut_data5_byte0, lcd lut address5 data byte1	0x00
0x56	RW	LSPI_LCD_LUT_DATA5_BYTE2 [7:0] lcd_lut_data5_byte0, lcd lut address5 data byte2	0x00
0x58	RW	LSPI_LCD_LUT_DATA6_BYTE0 [7:0] lcd_lut_data6_byte0, lcd lut address6 data byte0	0x00
0x59	RW	LSPI_LCD_LUT_DATA6_BYTE1 [7:0] lcd_lut_data6_byte0, lcd lut address6 data byte1	0x00
0x5a	RW	LSPI_LCD_LUT_DATA6_BYTE2 [7:0] lcd_lut_data6_byte0, lcd lut address6 data byte2	0x00
0x5c	RW	LSPI_LCD_LUT_DATA7_BYTE0 [7:0] lcd_lut_data7_byte0, lcd lut address7 data byte0	0x00
0x5d	RW	LSPI_LCD_LUT_DATA7_BYTE1 [7:0] lcd_lut_data7_byte0, lcd lut address7 data byte1	0x00
0x5e	RW	LSPI_LCD_LUT_DATA7_BYTE2 [7:0] lcd_lut_data7_byte0, lcd lut address7 data byte2	0x00
0x60	RW	LSPI_LCD_LUT_DATA8_BYTE0 [7:0] lcd_lut_data8_byte0, lcd lut address8 data byte0	0x00
0x61	RW	LSPI_LCD_LUT_DATA8_BYTE1 [7:0] lcd_lut_data8_byte0, lcd lut address8 data byte1	0x00
0x62	RW	LSPI_LCD_LUT_DATA8_BYTE2 [7:0] lcd_lut_data8_byte0, lcd lut address8 data byte2	0x00
0x64	RW	LSPI_LCD_LUT_DATA9_BYTE0 [7:0] lcd_lut_data9_byte0, lcd lut address9 data byte0	0x00
0x65	RW	LSPI_LCD_LUT_DATA9_BYTE1 [7:0] lcd_lut_data9_byte0, lcd lut address9 data byte1	0x00

Offset	Type	Description	Default Value
0x66	RW	LSPI_LCD_LUT_DATA9_BYTE2 [7:0] lcd_lut_data9_byte0, lcd lut address9 data byte2	0x00
0x68	RW	LSPI_LCD_LUT_DATA10_BYTE0 [7:0] lcd_lut_data10_byte0, lcd lut address10 data byte0	0x00
0x69	RW	LSPI_LCD_LUT_DATA10_BYTE1 [7:0] lcd_lut_data10_byte0, lcd lut address10 data byte1	0x00
0x6a	RW	LSPI_LCD_LUT_DATA10_BYTE2 [7:0] lcd_lut_data10_byte0, lcd lut address10 data byte2	0x00
0x6c	RW	LSPI_LCD_LUT_DATA11_BYTE0 [7:0] lcd_lut_data11_byte0, lcd lut address11 data byte0	0x00
0x6d	RW	LSPI_LCD_LUT_DATA11_BYTE1 [7:0] lcd_lut_data11_byte0, lcd lut address11 data byte1	0x00
0x6e	RW	LSPI_LCD_LUT_DATA11_BYTE2 [7:0] lcd_lut_data11_byte0, lcd lut address11 data byte2	0x00
0x70	RW	LSPI_LCD_LUT_DATA12_BYTE0 [7:0] lcd_lut_data12_byte0, lcd lut address12 data byte0	0x00
0x71	RW	LSPI_LCD_LUT_DATA12_BYTE1 [7:0] lcd_lut_data12_byte0, lcd lut address12 data byte1	0x00
0x72	RW	LSPI_LCD_LUT_DATA12_BYTE2 [7:0] lcd_lut_data12_byte0, lcd lut address12 data byte2	0x00
0x74	RW	LSPI_LCD_LUT_DATA13_BYTE0 [7:0] lcd_lut_data13_byte0, lcd lut address13 data byte0	0x00
0x75	RW	LSPI_LCD_LUT_DATA13_BYTE1 [7:0] lcd_lut_data13_byte0, lcd lut address13 data byte1	0x00
0x76	RW	LSPI_LCD_LUT_DATA13_BYTE2 [7:0] lcd_lut_data13_byte0, lcd lut address13 data byte2	0x00
0x78	RW	LSPI_LCD_LUT_DATA14_BYTE0 [7:0] lcd_lut_data14_byte0, lcd lut address14 data byte0	0x00
0x79	RW	LSPI_LCD_LUT_DATA14_BYTE1 [7:0] lcd_lut_data14_byte0, lcd lut address14 data byte1	0x00

Offset	Type	Description	Default Value
0x7a	RW	LSPI_LCD_LUT_DATA14_BYTE2 [7:0] lcd_lut_data14_byte0, lcd lut address14 data byte2	0x00
0x7c	RW	LSPI_LCD_LUT_DATA15_BYTE0 [7:0] lcd_lut_data15_byte0, lcd lut address15 data byte0	0x00
0x7d	RW	LSPI_LCD_LUT_DATA15_BYTE1 [7:0] lcd_lut_data15_byte0, lcd lut address15 data byte1	0x00
0x7e	RW	LSPI_LCD_LUT_DATA15_BYTE2 [7:0] lcd_lut_data15_byte0, lcd lut address15 data byte2	0x00
0x90	RW	LSPI_XIP_RD_FMT [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	RW	LSPI_XIP_RD_TRANSMODE [3:0] xip0_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip0_rd_transmode, xip read transmode/lcd display transmode	0x97
0x93	RW	LSPI_XIP_RD_CMD [7:0] xip0_rd_cmd, read command used for xip	0x3b

Offset	Type	Description	Default Value
0x94	RW	LSPI_XIP_WR_FMT [0] xip0_wr_data_dual, spi dual I/O mode, MASTER ONLY [1] xip0_wr_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	LSPI_XIP_WR_TRANSMODE [7:4] xip0_wr_transmode, xip write transmode/lcd porch transmode	0x10
0x97	RW	LSPI_XIP_WR_CMD [7:0] xip0_wr_cmd, write command used for xip/lcd_cmd	0x02

10.6 GSPI

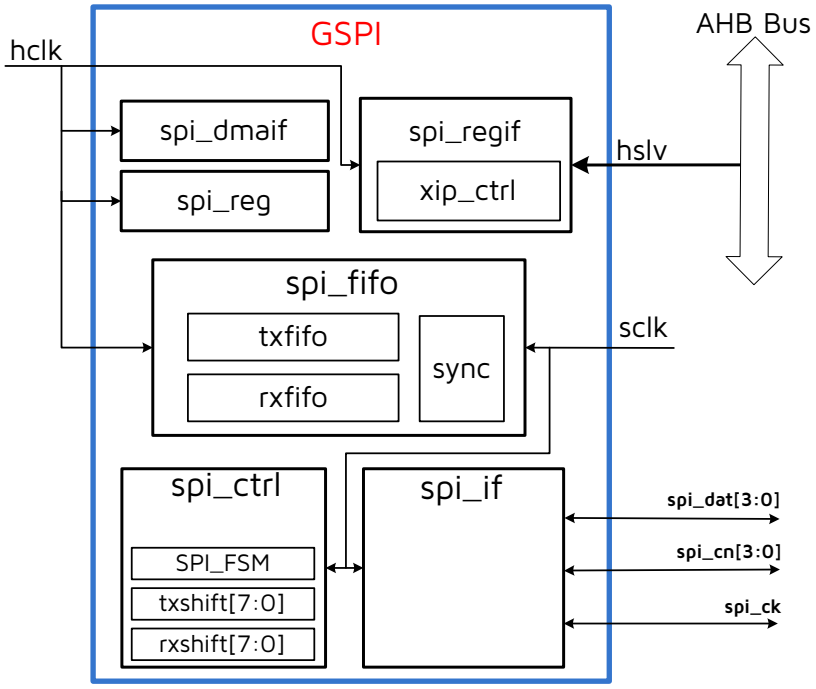
10.6.1 GSPI Diagram

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 Dual 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 10-12 GSPI Diagram



10.6.2 GSPI Register Description

The GSPI related registers are listed as below. The base address of the following registers is 0x8BFFFF00.

Table 10-10 GSPI Related Registers

Offset	Type	Description	Default Value
0x00	RW	GSPI_WR_RD_DATA0 [7:0] wr_rd_data0, data[7:0] to transmit or received	0x00
0x01	RW	GSPI_WR_RD_DATA1 [7:0] wr_rd_data1, data[15:8] to transmit or received	0x00
0x02	RW	GSPI_WR_RD_DATA2 [7:0] wr_rd_data2, data[23:16] to transmit or received	0x00
0x03	RW	GSPI_WR_RD_DATA3 [7:0] wr_rd_data3, data[31:24] to transmit or received	0x00
0x04	RW	GSPI_CMD [7:0]: SPI Command	0x00

Offset	Type	Description	Default Value
0x05	RW	GSPI_CTRL0 [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
0x07	RW	GSPI_TIMING [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	0x09
0x08	RW	GSPI_CTRL1 [1:0] data_lane, data lane, MASTER ONLY 0: single, 1: dual, 2: quad, 3: quad [3:2] addr_len, 2'b00:1bye 2'b01:2bytes 2'b10:3bytes 2'b11:4bytes, MASTER ONLY [4] addr_fmt, 0:single mode 1:the format of addr phase is the same as the data phase(Dual/Quad), MASTER ONLY [5] addr_en, 1:enabel addr phase, MASTER ONLY [6] cmd_fmt, 0: single mode 1: the format of the cmd phase is the same as the data phase(Dual/Quad), MASTER ONLY [7] cmd_en, the spi commnd phase enable, MASTER ONLY	0xa9

Offset	Type	Description	Default Value
0x09	RW	GSPI_CTRL2 [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~0xf:reserved	0x77
0x0c	RW	GSPI_ADDR0 [7:0] spi_addr0, spi address byte0, MASTER ONLY	0x0
0x0d	RW	GSPI_ADDR1 [7:0] spi_addr1, spi address byte1, MASTER ONLY	0x0
0x0e	RW	GSPI_ADDR2 [7:0] spi_addr2, spi address byte2, MASTER ONLY	0x0
0x0f	RW	GSPI_ADDR3 [7:0] spi_addr3, spi address byte3, MASTER ONLY	0x0
0x10	RW	GSPI_TX_CNT0 [7:0] tx_cnt0, transfer count for write data, byte0	0x00
0x11	RW	GSPI_TX_CNT1 [7:0] tx_cnt1, transfer count for write data, byte1	0x00
0x12	RW	GSPI_TX_CNT2 [7:0] tx_cnt2, transfer count for write data, byte2	0x00
0x14	RW	GSPI_RX_CNT0 [7:0] rx_cnt0, transfer count for read data, byte0	0x00

Offset	Type	Description	Default Value
0x15	RW	GSPI_RX_CNT1 [7:0] rx_cnt1, transfer count for read data, byte1	0x00
0x16	RW	GSPI_RX_CNT2 [7:0] rx_cnt2, transfer count for read data, byte2	0x00
0x18	RW	GSPI_CTRL3 [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	RW	GSPI_TXFIFO_THRES [5:0] txfifo threshold	0x00
0x1a	RW	GSPI_RXFIFO_THRES [5:0] rxfifo threshold	0x00
0x1c	RW	GSPI_CTRL4 [0] xip_page_mode_en, xip page mode enable [1] xip_timeout_mode_en, 0:xip timeout disable 1:xip timeout enable [2] xip_stop, stop xip [3] xip_enable, enable xip [4] dummy_cnt_add	0x0a
0x1d	RW	GSPI_XIP_PAGE_SIZE [7:0] page_size, page boundary size = 2^page_size	0x00
0x1e	RW	GSPI_XIP_TIMEOUT_CNT [7:0] timeout_cnt, timeout time sel	0x20
0x22	RW	GSPI_XIP_ADDR_OFFSET [7:0] xip_addr_offset, address offset = xip_addr_offset << 24	0x00
0x24	R	GSPI_TXFIFO_STATUS [6:0] txfifo_entries [7] txfifo_full	0x0

Offset	Type	Description	Default Value
0x25	R	GSPI_RXFIFO_STATUS [6:0] rxfifo_entries [7] rxfifo_empty	0x0
0x28	RW/R	GSPI_STATUS [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
0x2a	W1C	GSPI_INT_STATUS0 [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
0x90	RW	GSPI_XIP_RD_FMT [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	RW	GSPI_XIP_RD_TRANSMODE [3:0] xip0_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip0_rd_transmode	0x97

Offset	Type	Description	Default Value
0x93	RW	GSPI_XIP_RD_CMD [7:0] xip0_rd_cmd, read command used for xip	0x3b
0x94	RW	GSPI_XIP_WR_FMT [0] xip0_wr_data_dual, spi dual I/O mode, MASTER ONLY [1] xip0_wr_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	GSPI_XIP_WR_TRANSMODE [7:4] xip0_wr_transmode	0x10
0x97	RW	GSPI_XIP_WR_CMD [7:0] xip0_wr_cmd, write command used for xip	0x02
0x98	RW	GSPI_XIP1_RD_FMT [0] xip1_rd_data_dual, spi dual I/O mode, MASTER ONLY [1] xip1_rd_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	GSPI_XIP1_RD_TRANSMODE [3:0] xip1_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip1_rd_transmode	0x97

Offset	Type	Description	Default Value
0x9b	RW	GSPI_XIP1_RD_CMD [7:0] xip1_rd_cmd, read command used for xip	0x3b
0x9c	RW	GSPI_XIP1_WR_FMT [0] xip1_wr_data_dual, spi dual I/O mode, MASTER ONLY [1] xip1_wr_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	GSPI_XIP1_WR_TRANSMODE [7:4] xip1_wr_transmode	0x10
0x9f	RW	GSPI_XIP1_WR_CMD [7:0] xip1_wr_cmd, write command used for xip	0x02
0xa0	RW	GSPI_XIP2_RD_FMT [0] xip2_rd_data_dual, spi dual I/O mode, MASTER ONLY [1] xip2_rd_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	GSPI_XIP2_RD_TRANSMODE [3:0] xip2_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip2_rd_transmode	0x97

Offset	Type	Description	Default Value
0xa3	RW	GSPI_XIP2_RD_CMD [7:0] xip2_rd_cmd, read command used for xip	0x3b
0xa4	RW	GSPI_XIP2_WR_FMT [0] xip2_wr_data_dual, spi dual I/O mode, MASTER ONLY [1] xip2_wr_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	GSPI_XIP2_WR_TRANSMODE [7:4] xip2_wr_transmode	0x10
0xa7	RW	GSP2_XIP2_WR_CMD [7:0] xip2_wr_cmd, write command used for xip	0x02
0xa8	RW	GSPI_XIP3_RD_FMT [0] xip3_rd_data_dual, spi dual I/O mode, MASTER ONLY [1] xip3_rd_data_quad, spi quad I/O mode, MASTER ONLY [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
0xa9	RW	GSPI_XIP3_RD_TRANSMODE [3:0] xip3_rd_dummy_cnt, dummy number = dummy_cnt + 1 [7:4] xip3_rd_transmode	0x97

Offset	Type	Description	Default Value
0xab	RW	GSPI_XIP3_RD_CMD [7:0] xip3_rd_cmd, read command used for xip	0x3b
0xac	RW	GSPI_XIP3_WR_FMT [0] xip3_wr_data_dual, spi dual I/O mode, MASTER ONLY [1] xip3_wr_data_quad, spi quad I/O mode, MASTER ONLY [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	RW	GSPI_XIP3_WR_TRANSMODE [7:4] xip3_wr_transmode	0x10
0xaf	RW	GSP2_XIP3_WR_CMD [7:0] xip3_wr_cmd, write command used for xip	0x02
0xb0	RW	GSPI_XIP_SIZE_SET [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

10.7 SPI Slave (SPI_SLV)

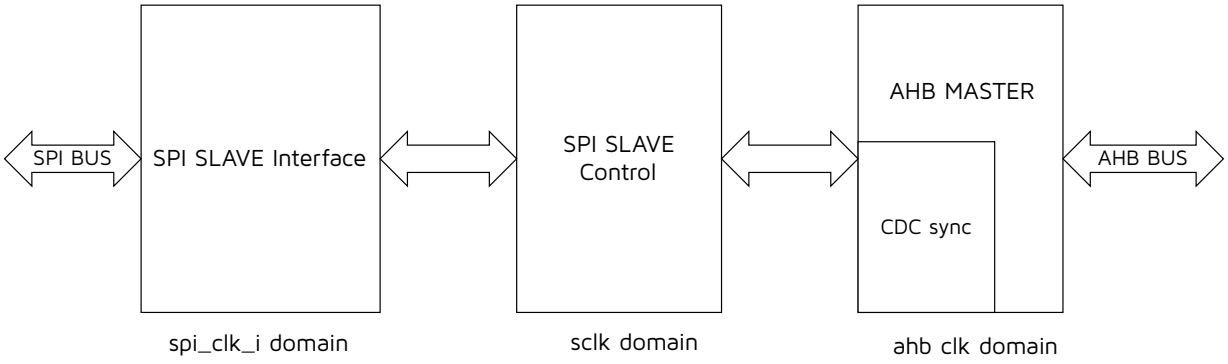
10.7.1 Diagram

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 10-13 SPI_SLV Diagram



10.7.2 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

10.7.3 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 10-14 SPI_SLV Write Format

Cmd(8bit)	Addr(32bit)	Data0(1byte)	Data1(1byte)	Data

Figure 10-15 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 10-11 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.

10.8 UART

10.8.1 Introduction

The SoC embeds UART (Universal Asynchronous Receiver/Transmitter) to implement full-duplex transmission and reception via UART TX and RX interface.

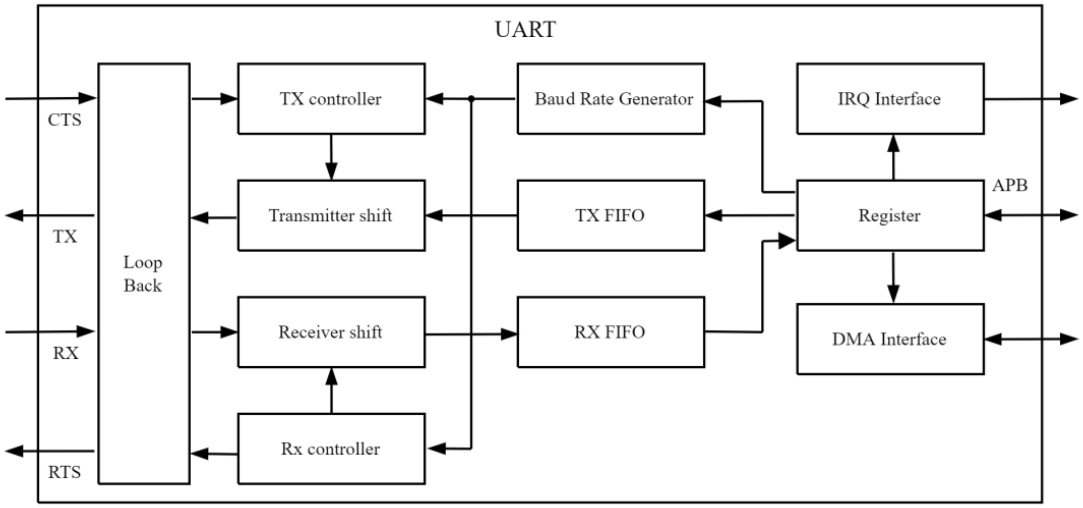
The UART module also supports ISO7816 protocol to enable communication with ISO/IEC 7816 integrated circuit card, especially smart card. In this mode, half-duplex communication (transmission or reception) is supported via the shared 7816_TRX interface.

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
- 2-channel UART (UART0, UART1)

10.8.2 Block Diagram

Figure 10-16 Block Diagram of UART



10.8.3 Function Description

10.8.3.1 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.

10.8.3.2 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 will assert 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 will assert dma_tx_req again unless the TX FIFO is full or the DMA transmission length is reached.

10.8.3.3 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 will assert 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 will assert dma_rx_req again unless the RX FIFO is empty. DMA relies on rxdone to read parts of the data below 4 characters.

10.8.3.4 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:

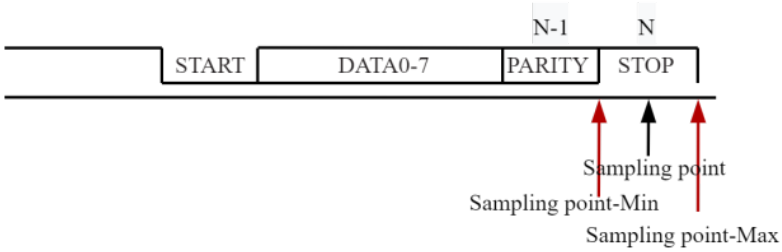
$$uart\_sclk = pclk/(uart\_clk\_div[14:0]+1)$$

$$Baudrate = uart\_sclk/(bpwc+1),\ bpwc > 2$$

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 10-17 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 10-12 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	

10.8.3.5 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.

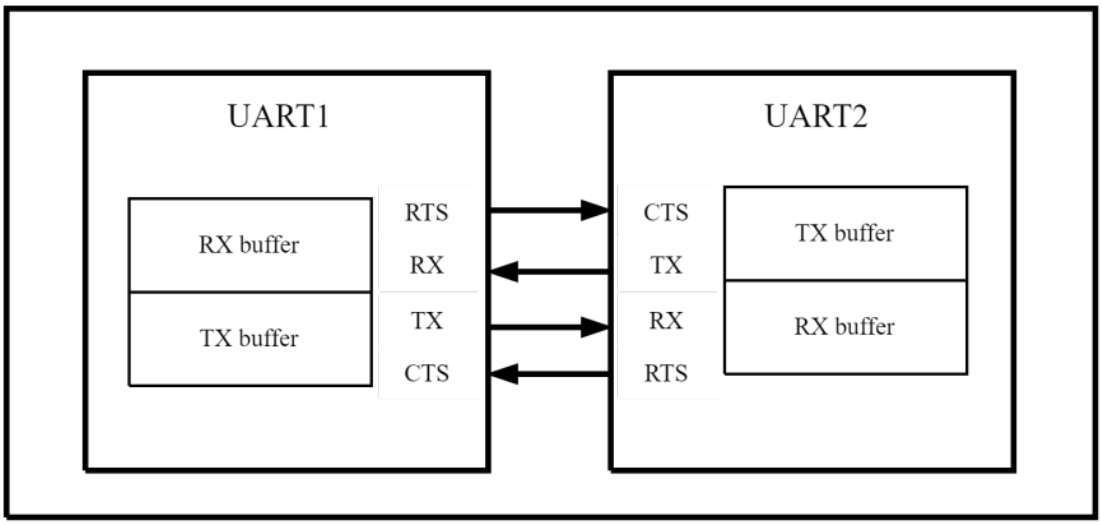
10.8.3.6 Error Conditions

An ERROR event, in the form of a framing error, will be generated if a valid stop bit is not detected in a frame. Second ERROR event, in the form of a break condition, will be 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.

10.8.3.7 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 10-18 Hardware Flow Control between 2 UARTs



If RX buffer of the UART is close to threshold, the UART will send 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 will behave as if the CTS and RTS lines are kept active all the time.

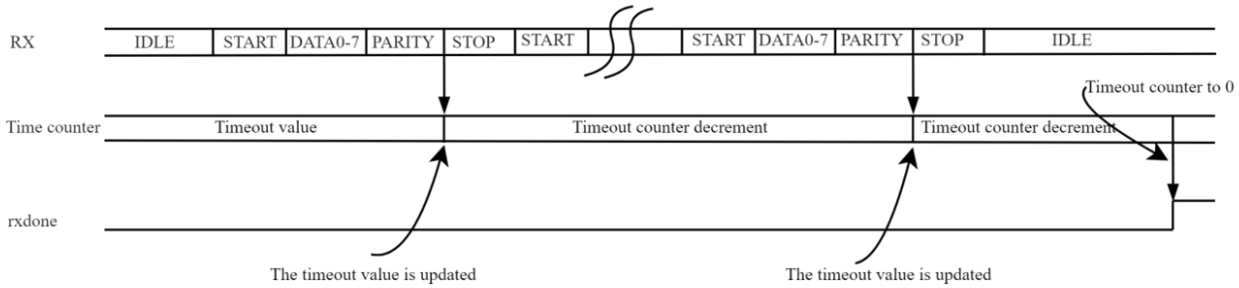
10.8.3.8 Receiver Timeout

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 10-19 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.

10.8.4 Register Description

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.

Table 10-13 UART Related Registers

Offset	Name	Type	Description	Default Value
0x00	uart_data_buf0	Volatile	Bit7-0 of Transmitter/Receiver Buffer Register (TX/RX FIFO)	0x00
0x01	uart_data_buf1	Volatile	Bit15-8 of Transmitter/Receiver Buffer Register (TX/RX FIFO)	0x00
0x02	uart_data_buf2	Volatile	Bit23-16 of Transmitter/Receiver Buffer Register (TX/RX FIFO)	0x00
0x03	uart_data_buf3	Volatile	Bit31-24 of Transmitter/Receiver Buffer Register (TX/RX FIFO)	0x00
0x04	uart_clk_div_l	RW	[7:0]: Least significant byte of the uart_clk_div register	0xff
0x05	uart_clk_div_h	RW	[6:0]: Most significant byte of the uart_clk_div register uart_sclk = pclk/(uart_clk_div[14:0]+1) [7]:enable clock divider 1:enable 0:disable	0x0f

Offset	Name	Type	Description	Default Value
0x06	uart_ctrl0	RW	[3:0] bpwc, bit width should be larger than 2 Baudrate = uart_sclk/(bpwc+1) [4] Auto clear function enable of DMA and NDMA mode; 1:enable,0:disable [5] rxdone (timeout) function enable in NDMA mode; 1:enable,0:disable; The DMA mode must disabled. [6] RTS controls timeout stop enabled; 1:enable,0:disable [7] 7816 enable; 1:enable,0:disable	0x7f
0x07	uart_ctrl1	RW	[0]: Polarity of CTS 0: Active low (0 - End of transmission) 1: Active high (1 - End of transmission) [1]: CTS enable, 1: enable, 0: disable [2]:Parity enable When this bit is set, a parity bit is generated in transmitted data before the first STOP bit and the parity bit would be checked for the received data. [3]: Even parity select, 1: old parity; 0: even parity (an even number of logic-1 is in the data and parity bits). [5:4]: stop bit, 00: STOP bit is 1 bit; 01: STOP bit is 1.5 bits; 1x: STOP bit is 2 bits [6]: TX and RX polarity selection, 0: Non-inverting; 1: Inverting [7]: Enable loopback mode, 1: enable; 0: disable	0x0e
0x08	uart_ctrl2	RW	[3:0]: RTS trig level. Trigger RTS when the RX FIFO reaches the threshold. [4]: Polarity of RTS 0: Active high (0-ready for receiving) 1: Active low (1-ready for receiving) [5]: RTS manual value [6]: RST manual enable [7]: RTS enable, 1: enable; 0: disable	0xa5

Offset	Name	Type	Description	Default Value
0x09	uart_ctrl3	RW	[3:0] rx_irq_trig level. Trigger rx_buf_irq interrupt when the RX FIFO reaches the threshold. [7:4] tx_irq_trig level. Trigger tx_buf_irq interrupt when the TX FIFO under the threshold.	0x44
0x0a	r_rxtimeout_l	RW	[7:0]: Least significant byte of the r_rxtimeout_o register: The setting is transfer one bytes need cycles base on uart_clk. For example, if transfer one byte (1start bit+8bits data+1 priority bit+2stop bits) total 12 bits, this register setting should be (bpwc+1)*12.	0xc0
0x0b	r_rxtimeout_h	RW	[1:0]: Most significant byte of the r_rxtimeout 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 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]: Enable rx_buf_irq interrupt [3]: Enable tx_buf_irq interrupt [4]: Enable rxdone_irq interrupt [5]: Enable txdone interrupt [6]: Enable rx_err interrupt [7]: Reserved	0x01
0x0c	buf_cnt	R	[3:0]: rx_buf_cnt This register is increased when there are incoming received data in the Receiver Buffer Register. When there is read data in the Receiver Buffer Register, this register is decremented. [7:4]: tx_buf_cnt This register is decremented when there are outgoing sent data in the Transmitter Buffer Register. When there is write data in the Transmitter Buffer Register, this register is increased.	0x00

Offset	Name	Type	Description	Default Value
0x0d	uart_sts	R	[2:0]: rcnt. When there is read data in the Receiver Buffer Register, this register is decremented. [3]: irq. Total interruption of UART. [6:4]: wbcnt. When there is write data in the Transmitter Buffer Register, this register is increased. [7]: rxdone. Similar to the rxdone irq interrupt, but rxdone is automatically cleared by the UART.	0x00

Offset	Name	Type	Description	Default Value
0x0e	irq_sts	Volatile	[1:0]: rx_rem_cnt_d. This register is increased when there are incoming received data in the Receiver Buffer Register. Gets incoming received data in the Receiver Buffer Register less than 1word. [2]: rx_buf_irq. When the RX FIFO reaches the threshold set by the rx_irq_trig Register, the UART controller will assert rx_buf_irq interrupt. W: write 1 to clear RX FIFO pointer, rx err, and so on. Note: When RX FIFO is below the threshold set by the rx_irq_trig Register, the rx_buf_irq interrupt clears automatically. [3]: tx_buf_irq When the TX FIFO under the threshold set by the tx_irq_trig register, the UART controller will assert tx_buf_irq interrupt. W: write 1 to clear TX FIFO pointer, and so on. Note: When TX FIFO is greater than the threshold set by the tx_irq_trig register, the tx_buf_irq interrupt clears automatically. [4]: rxdone_irq When the receiver ends (the timeout counter decays to 0), the UART controller will assert rxdone_irq interrupt. W: write 1 to clear rxdone_irq [5]: txdone. When the transmitter ends, the UART controller will assert txdone interrupt. W: write 1 to clear txdone [6]: rx_err. The error status is asserted when the following error events: parity errors framing errors, line breaks [7]: timeout The flag bit that the timeout counter decays to 0.	0x00

Offset	Name	Type	Description	Default Value
0x0f	state_sts	R	[2:0]: tx state machine 0-idle 1-Start 2-Byte 3-Parity 4-Stop 5-pop byte [3]: reserved [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: rxdone enables the RTS 0: rxdone disables the RTS [1]: Reserved	0x01

10.9 USB

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).

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.

NOTE: *Since 1.8V IO voltage does not comply with USB electrical layer regulations, the IO voltage of GPIO cannot be configured to 1.8V when using USB.*

The USB related registers are listed in table below, the base address is 0x80100800.

Table 10-14 USB Related Registers

Offset	Type	Description	Default Value
0x00	Volatile	EDP0PTR [3:0]: reg_ptr, Endpoint 0 buffer point	0x00
0x01	Volatile	EDP0DAT [7:0]: buff, Endpoint 0 buffer data access address	0x00
0x02	Volatile	EDP0CT [0]: ack_data, Ack data [1]: stall_data, Stall data [2]: ack_status, Ack status [3]: stall_status, Stall status [7:4]: udc_cnt, number of data transferred	0x00
0x03	R	EDP0ST [0]: irq_reset, W: Clear usb reset edge interrupted [1]: irq_250us_sof, W:Clear usb 250us or sof edge interrupted [2]: suspend_i, USB suspend status read only: suspend [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

Offset	Type	Description	Default Value
0x04	RW	EDP0MODE [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	RW	USBCT [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	R	CALCYCL [7:0]: r_clk_div_il, r_clk_div_i[7:0]	0x00
0x07	R	CALCYCH [2:0]: r_clk_div_ih, r_clk_div_i[10:8]	0x00
0x0a	RW	MDEV [0]: r_mdev, self power, 1: self power, 0: bus power [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	R	EDP0SIE [6:0]: sie_adr_i, usb_address [7]: r_config, config_now	0x00
0x0c	RW	SUSPENDCYC [4:0]: r_suspend_cnt, suspend_cnt [5]: r_edp0_stall, Avoid bugs with 8 multiples of in transfers [7:6]: hold0	0x18

Offset	Type	Description	Default Value
0x0d	R	INFALT [7:0]: r_infalt, Interface and alternate setting number in last SET_INTERFACE command	0x00
0x0e	RW	EDPS_EN [7:0]:edps_en	0xff
0x0f	RW	IRQ_MASK [2:0]: r_mask, mask[0]: irq_reset; mask[1]: irq_250us; mask[2]: irq_suspend; [4:3]: r_lvl, lvl[0]:0-->irq_reset_edge; 1-->usb_reset_i; lvl[1]:0-->irq_250us_edge; 1-->usb_250us_i; [5]: r_250us_sof_sel, 1'b0 sel 250us;1'b1 sel sof [7:6]: hold1	0x04
0x10	Volatile	EDPSPTR [7:0]: rd_ptrl	0x00
0x11	Volatile	EDPS1PTR [7:0]: rd_ptrl	0x00
0x12	Volatile	EDPS2PTR [7:0]: rd_ptrl	0x00
0x13	Volatile	EDPS3PTR [7:0]: rd_ptrl	0x00
0x14	Volatile	EDPS4PTR [7:0]: rd_ptrl	0x00
0x15	Volatile	EDPS5PTR [7:0]: rd_ptrl	0x00
0x16	Volatile	EDPS6PTR [7:0]: rd_ptrl	0x00
0x17	Volatile	EDPS7PTR [7:0]: rd_ptrl	0x60
0x18	Volatile	EDPSPTRH [1:0]: rd_ptrh	0x00
0x19	Volatile	EDPS1PTRH [1:0]: rd_ptrh	0x00

Offset	Type	Description	Default Value
0x1a	Volatile	EDPS2PTRH [1:0]: rd_ptrh	0x00
0x1b	Volatile	EDPS3PTRH [1:0]: rd_ptrh	0x00
0x1c	Volatile	EDPS4PTRH [1:0]: rd_ptrh	0x00
0x1d	Volatile	EDPS5PTRH [1:0]: rd_ptrh	0x00
0x1e	Volatile	EDPS6PTRH [1:0]: rd_ptrh	0x00
0x1f	Volatile	EDPS7PTRH [1:0]: rd_ptrh	0x00
0x20	Volatile	EDPSDATA [7:0]: sr_q	0x00
0x21	Volatile	EDPS1DATA [7:0]: sr_q	0x00
0x22	Volatile	EDPS2DATA [7:0]: sr_q	0x00
0x23	Volatile	EDPS3DATA [7:0]: sr_q	0x00
0x24	Volatile	EDPS4DATA [7:0]: sr_q	0x00
0x25	Volatile	EDPS5DATA [7:0]: sr_q	0x00
0x26	Volatile	EDPS6DATA [7:0]: sr_q	0x00
0x27	Volatile	EDPS7DATA [7:0]: sr_q	0x00

Offset	Type	Description	Default Value
0x28	Volatile	EDPSCT [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	Volatile	EDP1SCT [0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2a	Volatile	EDP2SCT [0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2b	Volatile	EDP3SCT [0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2c	Volatile	EDP4SCT [0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00
0x2d	Volatile	EDP5SCT [0]: rd_ack, ACK [1]: rd_stall, Stall [2]: set_data0, Set Data0 [3]: set_data1, Set Data1	0x00

Offset	Type	Description	Default Value
0x2e	Volatile	EDP6SCT [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	Volatile	EDP7SCT [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	RW	EDPSADR, Endpoint 8(0) buffer address low	0x80
0x31	RW	EDPS1ADR, Endpoint 1 buffer address low	0x00
0x32	RW	EDPS2ADR, Endpoint 2 buffer address low	0x08
0x33	RW	EDPS3ADR, Endpoint 3 buffer address low	0x10
0x34	RW	EDPS4ADR, Endpoint 4 buffer address low	0x40
0x35	RW	EDPS5ADR, Endpoint 5 buffer address low	0xc0
0x36	RW	EDPS6ADR, Endpoint 6 buffer address low	0x20
0x37	RW	EDPS7ADR, Endpoint 7 buffer address low	0x30
0x38	RW	EDPSADRH, [1:0] Endpoint 8(0) buffer address high	0x00
0x39	RW	EDPS1ADRH, [1:0] Endpoint 1 buffer address high	0x00
0x3a	RW	EDPS2ADRH, [1:0] Endpoint 2 buffer address high	0x00
0x3b	RW	EDPS3ADRH, [1:0] Endpoint 3 buffer address high	0x00
0x3c	RW	EDPS4ADRH, [1:0] Endpoint 4 buffer address high	0x00
0x3d	RW	EDPS5ADRH, [1:0] Endpoint 5 buffer address high	0x00
0x3e	RW	EDPS6ADRH, [1:0] Endpoint 6 buffer address high	0x00
0x3f	RW	EDPS7ADRH, [1:0] Endpoint 7 buffer address high	0x00

Offset	Type	Description	Default Value
0x40	RW	USBSO, Enable endpoint ISO mode	0xc0
0x41	RW	USBIRQ [7:0]: r_irq, Endpoint data transfer interrupt	0x00
0x42	RW	USBMASK, Endpoint interrupt mask	0xff
0x43	RW	USBMAX0, Maximum endpoint 8 transfer number: max_size = {USBMAX0[7:0],5'h0}	0x10
0x44	RW	USBMIN0, Minimum threshold to ACK endpoint 8 transfer (the buffer must have USBMIN8 data to ack to IN YOKEN)	0x40
0x45	RW	USBFIFO [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]: r_mode05	-
0x46	RW	USBMAX [6:0]: max_in, Max data in for endpoint buffer (except 7): Max_data_size =USBMAX*8	0x08
0x47	Volatile	USBTICK [7:0] r_tick, Just a tick that increase on posedge of the sclk_usb	0x00
0x48	RW	USBRAM [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	RW	USBMIN1 [1:0] usb_blk1, Minimum threshold to ACK endpoint 8 transfer [2] hold2	0x00

11 PWM

The SoC supports 6-channel PWM (Pulse-Width-Modulation) output. Each PWM#n (n=0~5) has its corresponding inverted output at PWM#n_N pin.

11.1 Enable PWM

Register PWM_EN[5:1] and PWM_EN0[0] serves to enable PWM5~PWM0 respectively via writing "1" for the corresponding bits.

11.2 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:

$$F_{PWM} = F_{System\_clock} / (PWM\_CLKDIV+1)$$

11.3 PWM Waveform, Polarity and Output Inversion

Each PWM channel has independent counter and 2 status including "Count" and "Remaining". Count and Remaining status form a signal frame.

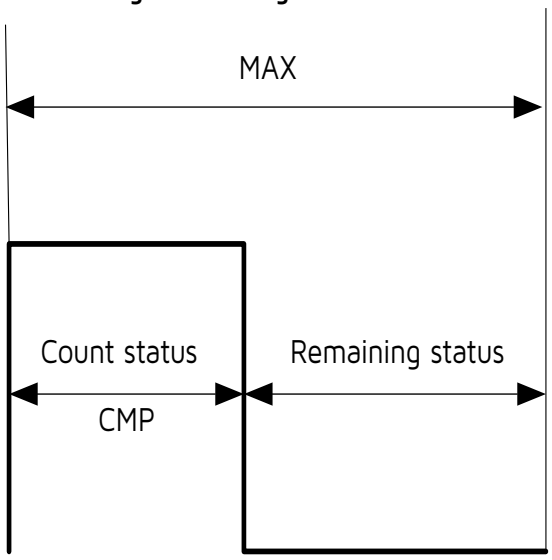
11.3.1 Waveform of Signal Frame

When PWM#n is enabled, first PWM#n enters Count status and outputs High level signal by default. When PWM#n counter reaches cycles set in register PWM_TCMP#n / PWM_TCMP_FSK_L/PWM_TCMP_FSK_H PWM#n enters Remaining status and outputs Low level till PWM#n cycle time configured in register PWM_TMAX#n / PWM_TMAX_FSK_L/ PWM_TMAX_FSK_H expires.

An interruption will be generated at the end of each signal frame if enabled via register PWM_MASK.

Signal frame is shown as following:

Figure 11-1 Signal Frame



11.3.2 Invert PWM Output

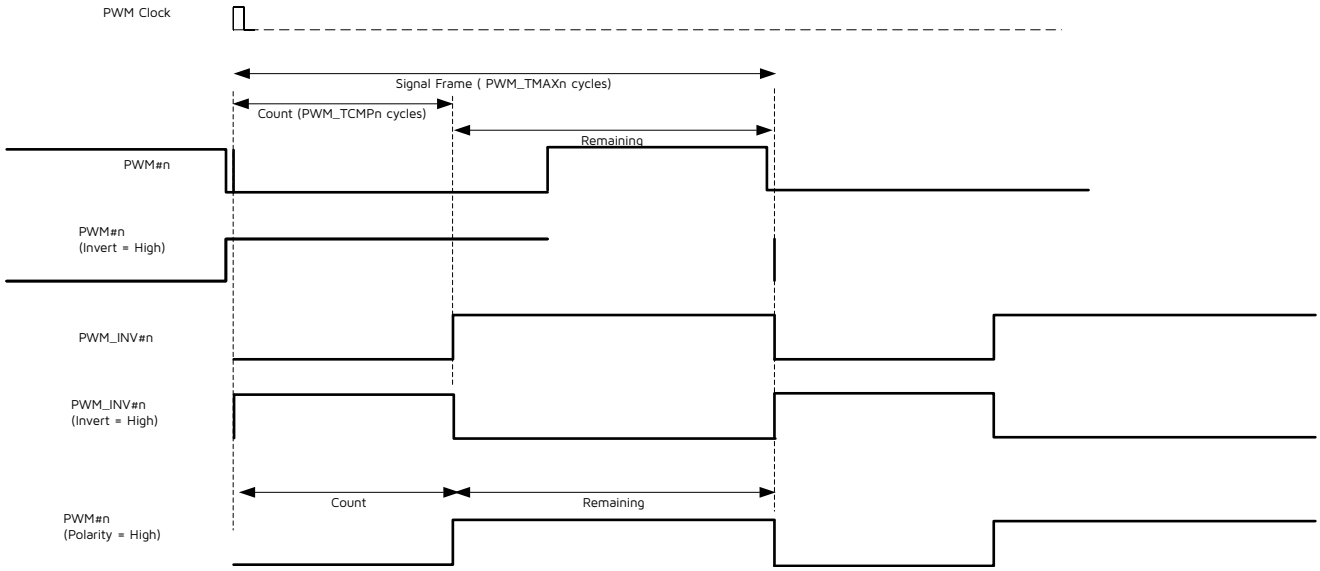
PWM#n and PWM#n_N output could be inverted independently via register PWM_CC0 and PWM_CC1. When the inversion bit is enabled, waveform of the corresponding PWM channel will be inverted completely.

11.3.3 Polarity for Signal Frame

By default, PWM#n outputs High level at Count status and Low level at Remaining status. When the corresponding polarity bit is enabled via register PWM_CC2[5:0], PWM#n will output Low level at Count status and High level at Remaining status.

PWM output waveform is shown as below.

Figure 11-2 PWM Output Waveform Chart



11.4 PWM Mode

11.4.1 Select PWM Modes

PWM0 supports five modes, including Continuous mode (normal mode, default), Counting mode, IR mode, IR FIFO mode, IR DMA FIFO mode.

PWM1~PWM5 only support Continuous mode.

Register PWM_MODE serves to select PWM0 mode.

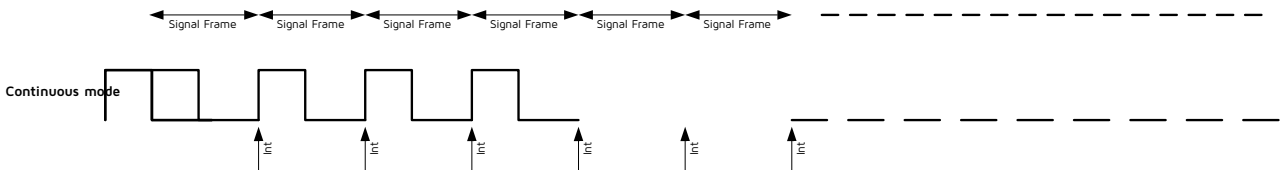
11.4.2 Continuous Mode

PWM0~PWM5 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 will turn low immediately.

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 will take effect in the next signal frame.

After each signal frame is finished, corresponding PWM cycle done interrupt flag bit (PWM_INT[2:7]) will be automatically set to 1'b1. If the interrupt is enabled by setting PWM_MASK0[2:7] as 1'b1, a frame interruption will be generated. User needs to write 1'b1 to the flag bit to manually clear it.

Figure 11-3 Continuous Mode



11.4.3 Counting Mode

Only PWM0 supports Counting mode. PWM_MODE [2:0] should be set as 4b'0001 to select PWM0 counting mode.

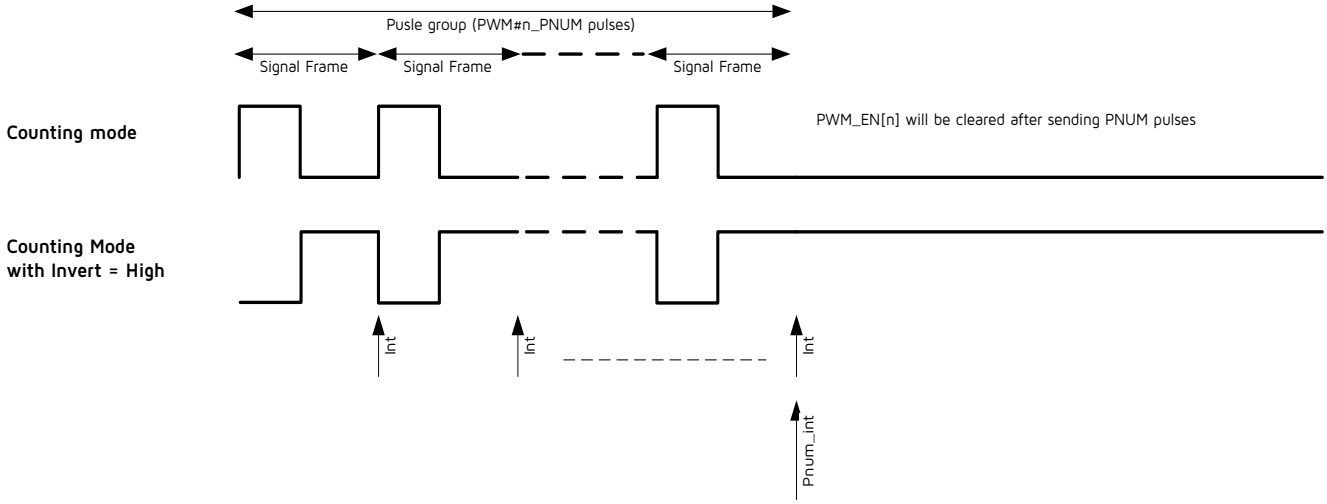
In this mode, PWM0 sends out specified number of signal frames which is defined as a pulse group. The number is configured via register PWM_PNUM.

After each signal frame is finished, PWM0 cycle done interrupt flag bit (PWM_INT[2]) will be automatically set to 1'b1. If the interrupt is enabled by setting PWM_MASK0 [2] as 1'b1, a frame interruption will be generated. User needs to write 1'b1 to the flag bit to manually clear it.

After a pulse group is finished, PWM0 will be disabled automatically, and PWM0 pnum interrupt flag bit (PWM_INT [0]) will be automatically set to 1'b1. If the interrupt is enabled by setting PWM_MASK0 as 1'b1, a Pnum interruption will be generated. User needs to write 1'b1 to the flag bit to manually clear it.

Counting mode also serves to stop IR mode gracefully.

Figure 11-4 Counting Mode (n=0)



11.4.4 IR Mode

Only PWM0 supports IR mode. PWM_MODE[2:0] should be set as 4b'0011 to select PWM0 IR mode.

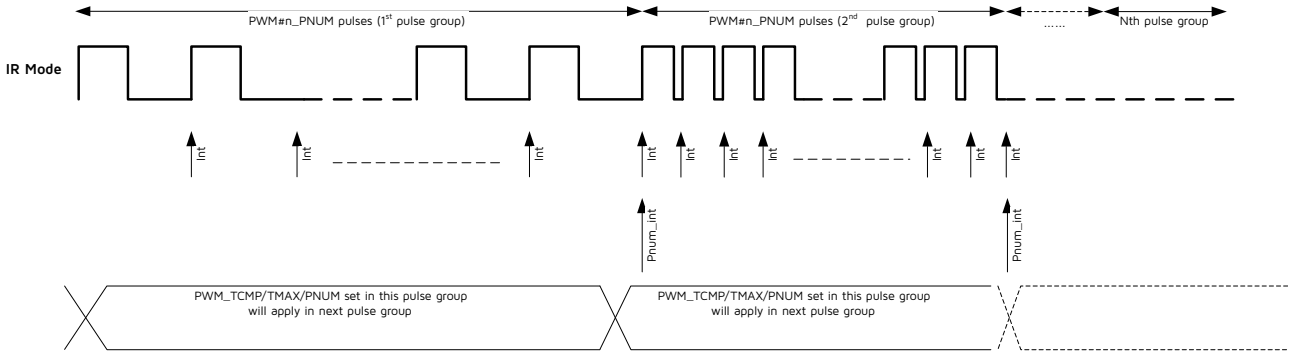
In this mode, specified number of frames is defined as one pulse group. In contrast to Counting mode where PWM0 stops after first pulse group is finished, PWM0 will constantly send pulse groups in IR mode.

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 will 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 will stop after current pulse group is finished. If PWM0 is disabled directly via PWM_EN0[0], PWM0 output will turn Low immediately despite of current pulse group.

After each signal frame/pulse group is finished, PWM0 cycle done interrupt flag bit (PWM_INT[2])/PWM0 pnum interrupt flag bit (PWM_INT[0]) will be automatically set to 1'b1. A frame interruption/Pnum interruption will be generated.

Figure 11-5 IR Mode (n=0)



11.4.5 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[2:0] 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:

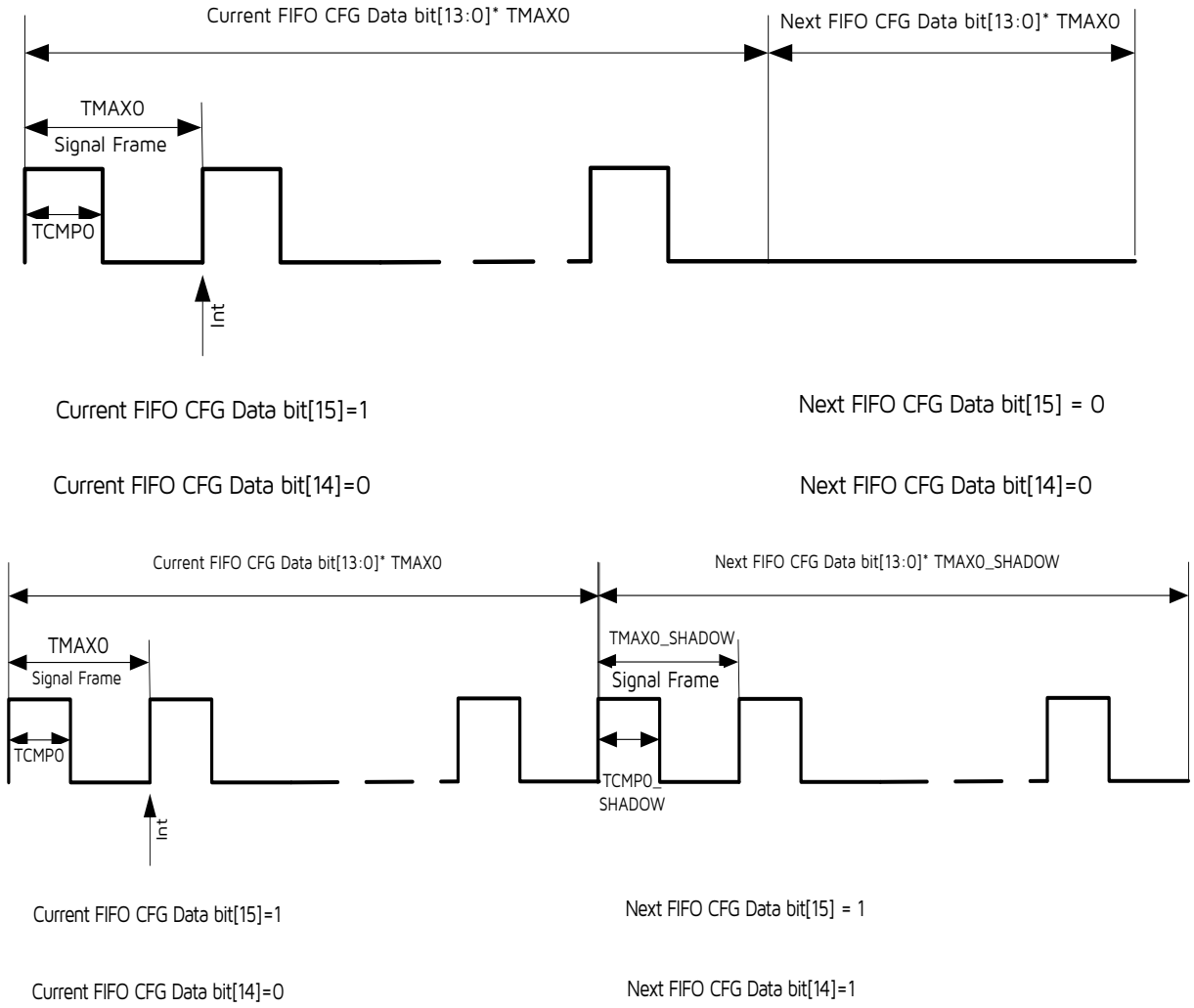
- bit[13:0] defines PWM pulse number of current group.
- bit[14] determines duty cycle and period for current PWM pulse group.
- 0: use configuration of TCMP0 and TMAX0;
- 1: use configuration of PWM_TCMP_FSK_L/PWM_TCMP_FSK_H and PWM_TMAX_FSK_L/ PWM_TMAX_FSK_H.
- bit[15] determines whether current PWM pulse group is used as carrier, i.e. whether PWM will output pulse (1) or low level (0).

User should use PWM_RDAT_L0, PWM_RDAT_H0, PWM_RDAT_L1, PWM_RDAT_H1 in 0x7c8~0x7cb 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 0x7c8, 0x7c9, 0x7ca and 0x7cb.
- To write by half word, user should successively write 0x7c8 and 0x7ca.
- To write by word, user should write 0x7c8.

FIFO depth is 8 bytes. User can read the register FIFO_SR in 0x7cd to view FIFO empty/full status and check FIFO data number.

Figure 11-6 IR Format Examples



When "FIFO CFG Data" is configured in FIFO and PWM0 is enabled via PWM_EN0[0], the configured waveforms will be 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 will be disabled automatically.

The FIFO_CLR register serves to clear data in FIFO. Writing 1'b1 to this register will clear all data in the FIFO. Note that the FIFO can only be cleared when not in active transmission.

11.4.6 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 will automatically write the configuration into FIFO.

NOTE: *In this mode, when DMA channel 5 is enabled, PWM will automatically output configured waveform, without the need to manually enable PWM0 via PWM_EN0 (i.e. PWM_EN0[0] will be set as 1'b1 automatically).*

11.5 PWM Interrupt

There are 9 interrupt sources from PWM function.

After each signal frame, PWM#n (n = 0 ~ 5) will generate a frame-done IRQ (Interrupt Request) signal.

In Counting mode and IR mode, PWM0 will generate a Pnum IRQ signal after completing a pulse group.

In IR FIFO mode, PWM0 will generate a FIFO mode count IRQ signal when the FIFO_NUM value is less than the FIFO_NUM_LVL, and will generate a FIFO mode stop IRQ signal after FIFO becomes empty.

In IR DMA FIFO mode, PWM0 will generate an IR waveform send done IRQ signal, after DMA has sent all configuration data, FIFO becomes empty and final waveform is sent.

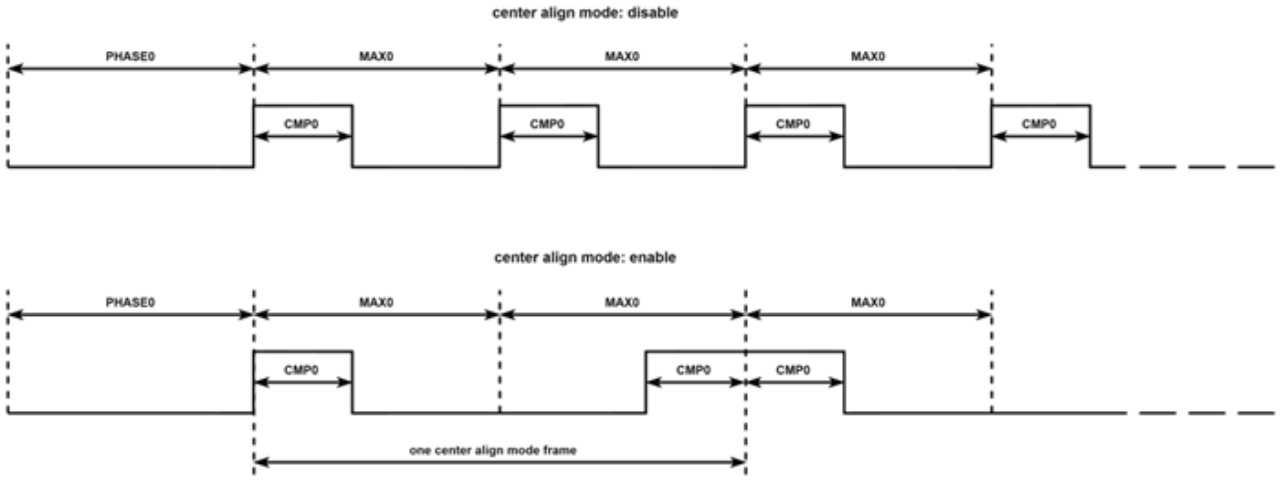
To enable PWM interrupt, the total enabling bit "irq_pwm" should be set as 1'b1. To enable various PWM interrupt sources, PWM_MASK0 and PWM_MASK1 should be set as 1'b1 correspondingly.

Interrupt status can be cleared via register PWM_INT0 and PWM_INT1.

11.6 PWM Center Align Mode

PWM#n (n = 0 ~ 5) support center align mode. For example, PWM0's center align mode is enabled by setting PWM_CENTER[5:0] as 6'b000001. PWM0 output waveform is shown as below.

Figure 11-7 Center Align Mode



11.7 Register Description

PWM related registers are listed as following. The base address for below registers is 0x80140400.

Table 11-1 PWM Registers

Offset	Type	Description	Default Value
0x00	W	PWM_EN pwm[5:1] enable	0x00
0x01	W	PWM_EN0 pwm0 enable	0x00
0x02	RW	PWM_CLKDIV	0x00
0x03	RW	PWM_MODE [0]:crun_o [1]:catch_o [2]:fifio_mode_en	0x00
0x04	RW	PWM_CC0 invert PWM output	0x00
0x05	RW	PWM_CC1 invert PWM_INV output	0x00
0x06	RW	PWM_CC2 PWM pola	0x00
0x07	RW	MODE32K	0x00
0x14	RW	PWM_TCMP0_L tcmpb0[7:0] bits 7-0 of PWM0's high time or low time	0x00
0x15	RW	PWM_TCMP0_H tcmpb0[15:8] bits 15-8 of PWM0's high time or low time	0x00
0x16	RW	PWM_TMAX0_L tmaxb0[7:0] bits 7-0 of PWM0's cycle time	0x00
0x17	RW	PWM_TMAX0_H tmaxb0[15:8] bits 15-8 of PWM0's cycle time	0x00
0x18	RW	PWM_TCMP1_L tcmpb1_o[7:0] bits 7-0 of PWM1's high time or low time	0x00

Offset	Type	Description	Default Value
0x19	RW	PWM_TCMP1_H tcmpb1_o[15:8] bits 15-8 of PWM1's high time or low time	0x00
0x1a	RW	PWM_TMAX1_L tmaxb1_o[7:0] bits 7-0 of PWM1's cycle time	0x00
0x1b	RW	PWM_TMAX1_H tmaxb1_o[15:8] bits 15-8 of PWM1's cycle time	0x00
0x1c	RW	PWM_TCMP2_L tcmpb2_o[7:0] bits 7-0 of PWM2's high time or low time	0x00
0x1d	RW	PWM_TCMP2_H tcmpb2_o[15:8] bits 15-8 of PWM2's high time or low time	0x00
0x1e	RW	PWM_TMAX2_L tmaxb2_o[7:0] bits 7-0 of PWM2's cycle time	0x00
0x1f	RW	PWM_TMAX2_H tmaxb2_o[15:8] bits 15-8 of PWM2's cycle time	0x00
0x20	RW	PWM_TCMP3_L tcmpb3_o[7:0] bits 7-0 of PWM3's high time or low time	0x00
0x21	RW	PWM_TCMP3_H tcmpb3_o[15:8] bits 15-8 of PWM3's high time or low time	0x00
0x22	RW	PWM_TMAX3_L tmaxb3_o[7:0] bits 7-0 of PWM3's cycle time	0x00
0x23	RW	PWM_TMAX3_H tmaxb3_o[15:8] bits 15-8 of PWM3's cycle time	0x00
0x24	RW	PWM_TCMP4_L tcmpb4_o[7:0] bits 7-0 of PWM4's high time or low time	0x00
0x25	RW	PWM_TCMP4_H tcmpb4_o[15:8] bits 15-8 of PWM4's high time or low time	0x00

Offset	Type	Description	Default Value
0x26	RW	PWM_TMAX4_L tmaxb4_o[7:0] bits 7-0 of PWM4's cycle time	0x00
0x27	RW	PWM_TMAXB4_H tmaxb4_o[15:8] bits 15-8 of PWM4's cycle time	0x00
0x28	RW	PWM_TCMP5_L tcmpb5_o[7:0] bits 7-0 of PWM5's high time or low time	0x00
0x29	RW	PWM_TCMP5_H tcmpb5_o[15:8] bits 15-8 of PWM5's high time or low time	0x00
0x2a	RW	PWM_TMAX5_L tmaxb5_o[7:0] bits 7-0 of PWM5's cycle time	0x00
0x2b	RW	PWM_TMAX5_H tmaxb5_o[15:8] bits 15-8 of PWM5's cycle time	0x00
0x2c	RW	PWM_PNUM_L pnumb[7:0]	0x00
0x2d	RW	PWM_PNUM_H pnumb[13:8]	0x00
0x2e	RW	PWM_CENTER [5:0]:center_align [6]:auto_txclr_off [7]:txf_nempty_en	0x00
0x30	RW	PWM_MASK [0]:mask_pwm [1]:mask_fifo [7:2]:mask	0x00
0x31	Volatile	PWM_INT [0]:int_pwm,count model interrupt flag [1]:int_fifo_done,int_fifo_done [7:2]:int_flag,w1c too, int_flag[5:0]	0x00
0x32	RW	PWM_MASK_LVL	0x00
0x33	Volatile	PWM_INT_LVL	0x00

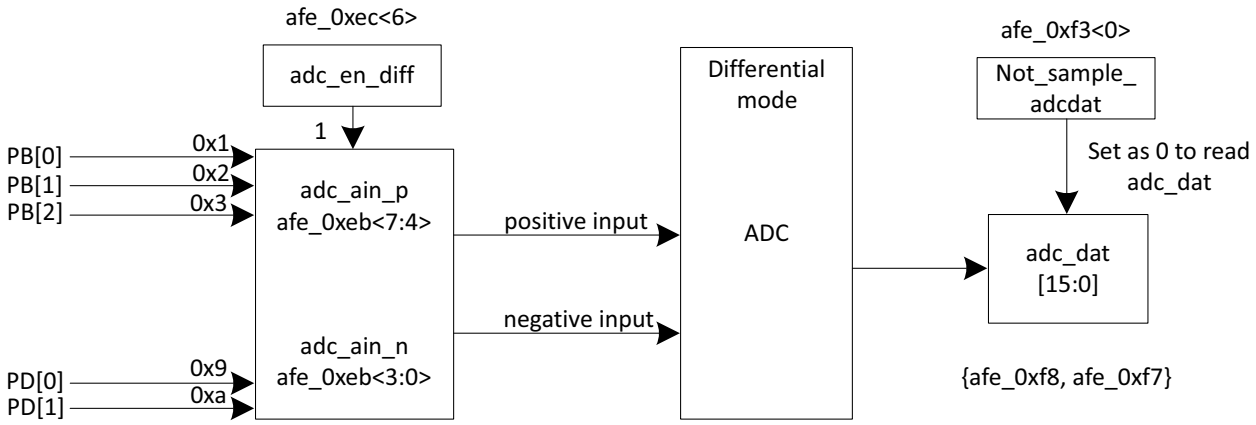
Offset	Type	Description	Default Value
0x34	Volatile	PWM_CNT0_L	0x00
0x35	Volatile	PWM_CNT0_H	0x00
0x36	Volatile	PWM_CNT1_L	0x00
0x37	Volatile	PWM_CNT1_H	0x00
0x38	Volatile	PWM_CNT2_L	0x00
0x39	Volatile	PWM_CNT2_H	0x00
0x3a	Volatile	PWM_CNT3_L	0x00
0x3b	Volatile	PWM_CNT3_H	0x00
0x3c	Volatile	PWM_CNT4_L	0x00
0x3d	Volatile	PWM_CNT4_H	0x00
0x3e	Volatile	PWM_CNT5_L	0x00
0x3f	Volatile	PWM_CNT5_H	0x00
0x40	R	PWM_NCNT_L	0x00
0x41	R	PWM_NCNT_H	0x00
0x44	RW	PWM_TCMP_FSK_L	0x00
0x45	RW	PWM_TCMP_FSK_H	0x00
0x46	RW	PWM_TMAX_FSK_L	0x00
0x47	RW	PWM_TMAX_FSK_H	0x00
0x48	R	PWM_RDAT_L0	0x00
0x49	R	PWM_RDAT_H0	0x00
0x4a	R	PWM_RDAT_L1	0x00
0x4b	R	PWM_RDAT_H1	0x00
0x4c	RW	PWM_FIFO_LVL	0x00
0x4d	Volatile	PWM_TX_CTRL	0x10
0x4e	W	CLR_TXFIFO	0x00

12 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 12-1 Diagram of ADC



NOTE:

- If the IO voltage of GPIO is configured as 3.3V, the sampling range of GPIO signal is 0 ~ 3.3 V.

12.1 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.

12.2 ADC Clock

The ADC clock is derived from external 24 MHz crystal source, with frequency dividing factor configured 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.

12.3 ADC Control in Auto Mode

12.3.1 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>).

12.3.2 "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.

12.3.3 "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>) will be set as 1'b1 at the end of "Capture" state to indicate the ADC data is valid, and this flag bit will be 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}*

12.3.4 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 12-1 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
Set resolution	afe_0xec<1:0> = 3 resolution: 14 bits

Function	Register Setting
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

12.4 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 3.3V, the sampling range of VBAT is 1.8 ~ 4.3 V.

12.5 Register Table

Table 12-2 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
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

Address	Default Value	Description
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: vbatdiv
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
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

Address	Default Value	Description
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 will be set as 1 at the end of capture state to indicate the ADC data is valid, and will be cleared when set state starts.)
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

Address	Default Value	Description
afe_0xf9<3:2>	0	Vbat divider select sel_vbatdiv[1:0] Vbatdiv 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%
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

Address	Default Value	Description
afe_0xfc<5>	1	Power down ADC 1: Power down 0: Power up

13 Temperature Sensor

The SoC integrates a temperature sensor and it's used in combination with the SAR ADC to detect real-time temperature.

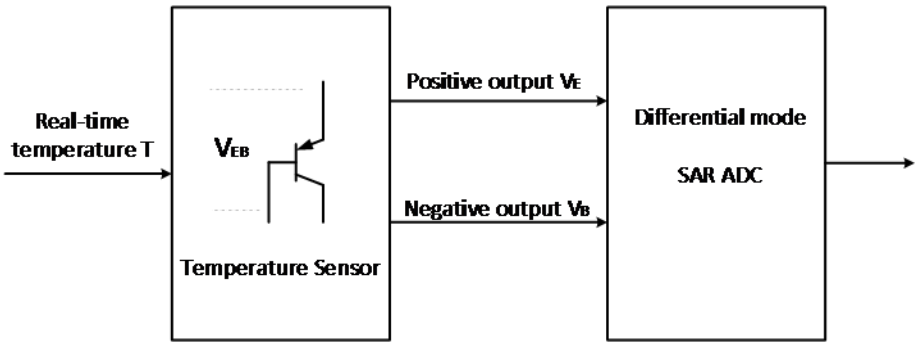
The temperature sensor is disabled by default. The analog register afe_0x06<2> should be set as 1'b0 to enable the temperature sensor.

Table 13-1 Analog Register for Temperature Sensor

Address	Type	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 13-1 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 will convert 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.

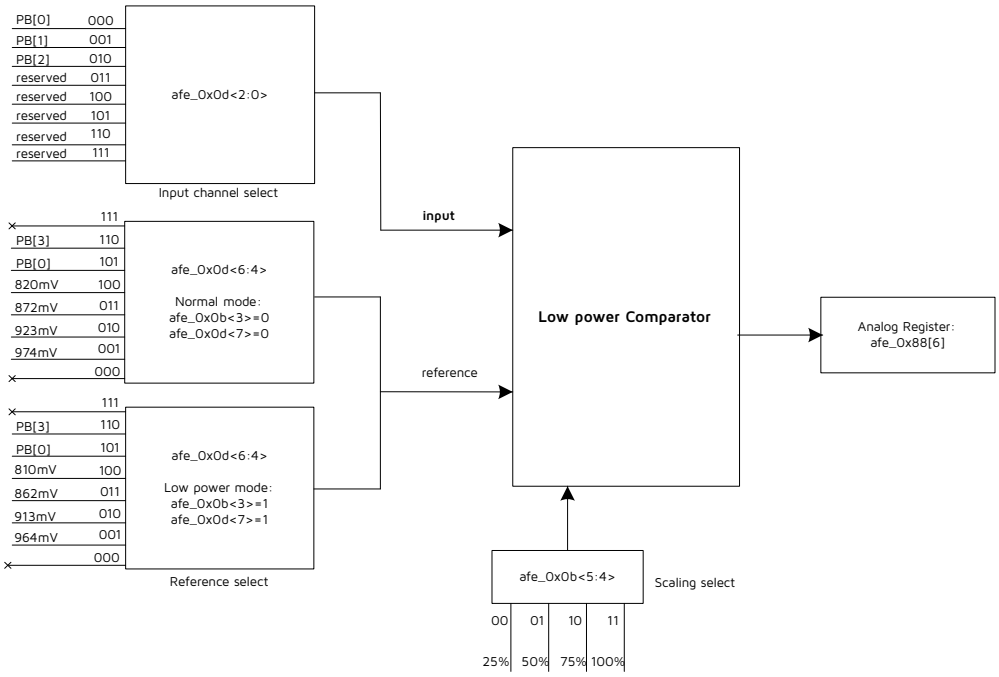
Then the real-time temperature T can be calculated according to the V_{EB}.

14 Low Power Comparator

The SoC embeds a low power comparator. This comparator takes two inputs: input derived from external PortB (PB[1]~PB[2]), and reference input derived from internal reference, PB[0] or float.

By comparing the input voltage multiplied by selected scaling coefficient with reference input voltage, the low power comparator will output high or low level accordingly.

Figure 14-1 Block Diagram of Low Power Comparator



14.1 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 will be powered on; by setting this bit to 1'b1, this comparator will be 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.

14.2 Select Input Channel

Input channel is selectable from the PortB (PB[1]~PB[2]) via the analog register afe_0x0d<2:0>.

14.3 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 UVLO 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] 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 Bandgap 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] and float.

14.4 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%.

14.5 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 will be low ("0").
- If the value of [input voltage *scaling] is lower than reference voltage input, the output will be high ("1").
- If the value of [input voltage *scaling] equals reference voltage input, or input channel is selected as float, the output will be 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.

14.6 Register Description

Table 14-1 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; 1: 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[1] / 0x110 -> B[1]	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

Address	Description	Default Value
afe_0x4d<0>	pd_lc_comp auto 1: auto power down low power comparator	0x0

15 Secure Boot, Firmware Encryption and Debug Lock

The TLSR9228 supports secure boot function. Secure boot ensures that the chip boots from the secure code in unmodifiable Boot-ROM at the time of booting, and that the code is verified, encrypted and decrypted. The secure boot process allows the user to form a trusted security chain based on a secure root key, ensuring that all code executed is trusted and secure.

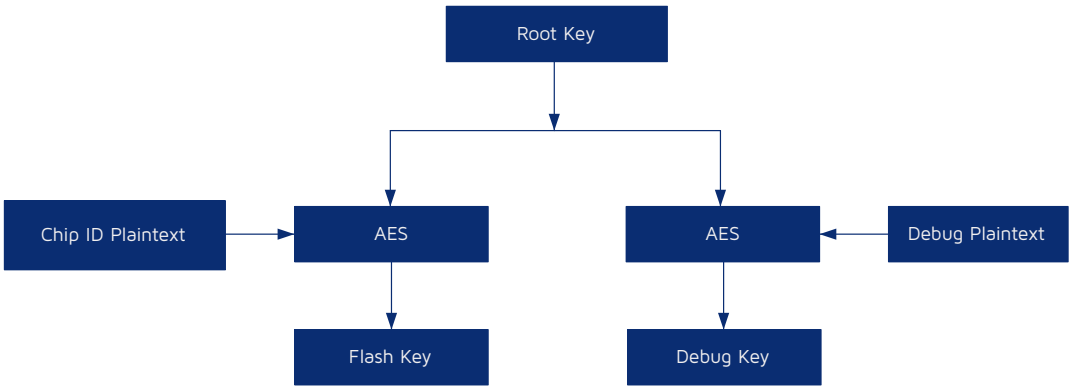
An important function of secure boot is public key verification. Public key verification means that the user signs the firmware with the private key. The chip side uses the public key to verify the signature before running the firmware. The firmware will run only if the verification passes. The signature verification uses the Elliptic Curve Digital Signature Algorithm (ECDSA). The public key verification ensures that only the code that passes the verification can be executed.

In order to protect the code from being cloned or being stored in plaintext, code encryption can be used. The firmware is encrypted by hardware when it is downloaded to flash. The encryption adopts an AES-128-like Light Crypto encryption algorithm. When the chip is running, it will decrypt the firmware in real time.

The TLSR9228 also supports option for disabling the debug port so that during mass production, the debug port is disabled and as a result, hackers will not have means to access any registers or memories within the chip through debug interfaces.

15.1 Key Management

Figure 15-1 Key Management

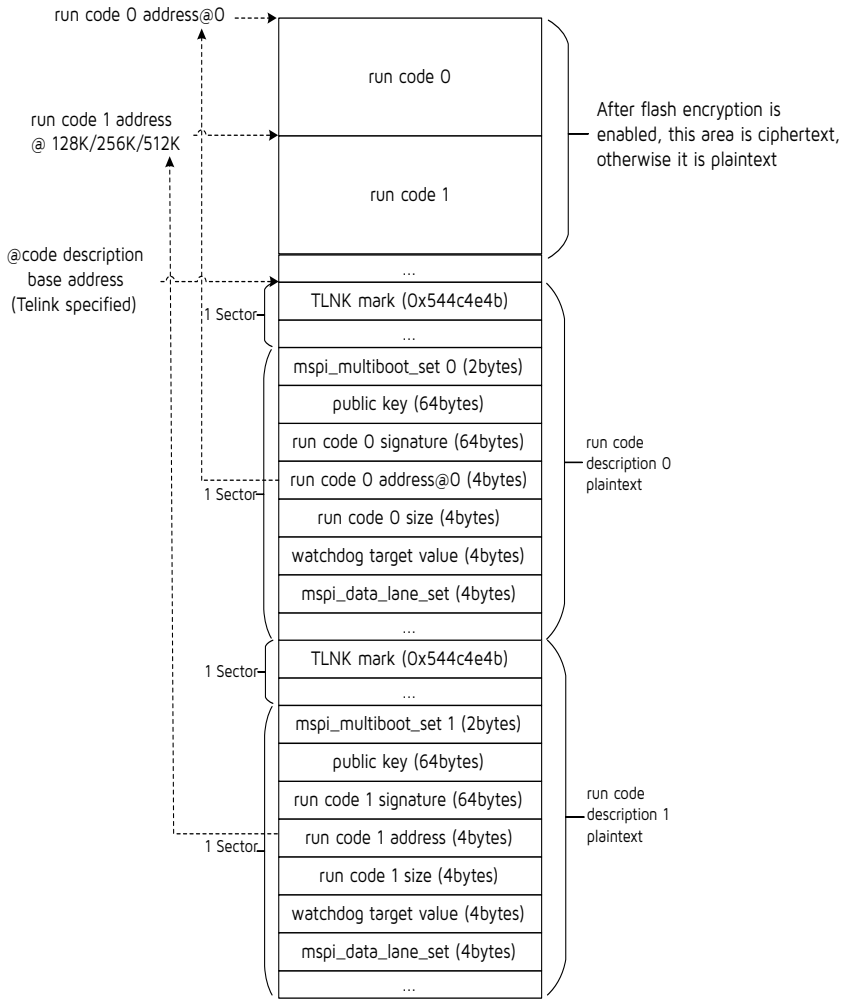


The keys used for real-time firmware encryption (Flash key) and for debug port lock (Debug key), are generated from the root key by the AES module through the above key derivation structure. The root key can be set by the customer to control its own product security derivation structures. The root key is a customer provisioned and controlled key that ensures that the customer has full control of the encryption and debug lock process. The flash key is derived from the root key and the chip ID plaintext through an AES-based derivation process and is unique per chip; the debug key is derived from the root key and the debug plaintext through AES-based derivation process, the debug plaintext is provided by customers as a means to identify their unique debug key and is unique per customer or per customer project. The flash key is the key for real-time encryption and decryption of the firmware, and the debug key is the key used to enable the debug interfaces when the debug interfaces are disabled.

15.2 Flash Space and eFuse Definition

15.2.1 Flash Space in Secure Boot Mode

Figure 15-2 Flash Space for Secure Boot



The flash may contain several segments of data including code 0 & 1, and their corresponding code description 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 description block is only present if secure boot with firmware signature verification is enabled.

The code, which has a fixed starting address of 0, is the firmware code to be executed. 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 description 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 mspi_multiboot_set is used to set parameters related to run code address.
- 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.
- The mspi_data_lane_set is to set the desired MSPI width for the running data. The data lane mode can be selected from 2-lane and 4-lane, please use the default value provided by Telink.

15.2.2 eFuse Definition in Secure Boot Mode

The eFuse definition in secure boot mode is shown in the table below. Please also check 4.1.3 eFuse for details on eFuse.

Table 15-1 eFuse Data for Secure Boot

Definition	Size (Bit)	Description
root_key	128	Root key, a customer provided key for Flash key and Debug key derivation
chip_ID_plaintext	128	Unique chip ID
Debug_plaintext	128	Debug plaintext, is a text string provided by the customer to control the debug lock feature for a customer project
public_key_hash	256	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 Boot-ROM to verify that the correct public key for firmware signature verification is used.

15.2.3 Use of Debug Key

If the user wants to re-enable debug after the debug port is locked, he needs to send the 129-bit debug key sequence (128-bit debug key with one more bit of 0) according to the single wire protocol by high byte first to enable debug port.

15.3 Usage

The chip has two modes: normal mode and secure boot mode. The two modes are configured by compatible identifier in eFuse Security_Feature field. If bit [29] is 1, it is secure boot mode. If bit [29] is 0, it is normal mode.

Normal Mode

In this mode, the firmware plaintext runs from address offset 0K/128K/256K/512K bytes of Flash;

Flash encryption function can be enabled in normal mode by setting Security_Feature bit[31].

Secure Boot Mode

There are following four configurations in secure boot mode:

1. No encryption or signature verification: bit[31] Flash encryption disabled, bit[29] secure boot mode set to secure mode (1).

In this mode, neither firmware encryption nor firmware signature verification is enabled. This configuration is not recommended in Secure Boot mode.

2. Encryption only without signature verification: bit[31] Flash encryption enabled, bit[29] secure boot mode set to secure mode (1).

In this mode, the firmware stored on the Flash is encrypted. It is decrypted in real-time during running. This configuration is also not recommended in Secure Boot mode. For Flash encryption only, user is recommended to use the Normal Mode with Flash encryption option.

3. Signature verification only without encryption: bit[31] Flash encryption disabled, bit[29] secure boot mode set to secure mode (1).

In this mode, firmware is stored in plaintext on the Flash, but the code description information is used to verify its signature.

4. Encryption and signature verification: bit[31] Flash encryption enabled, bit[29] secure boot mode set to secure mode (1).

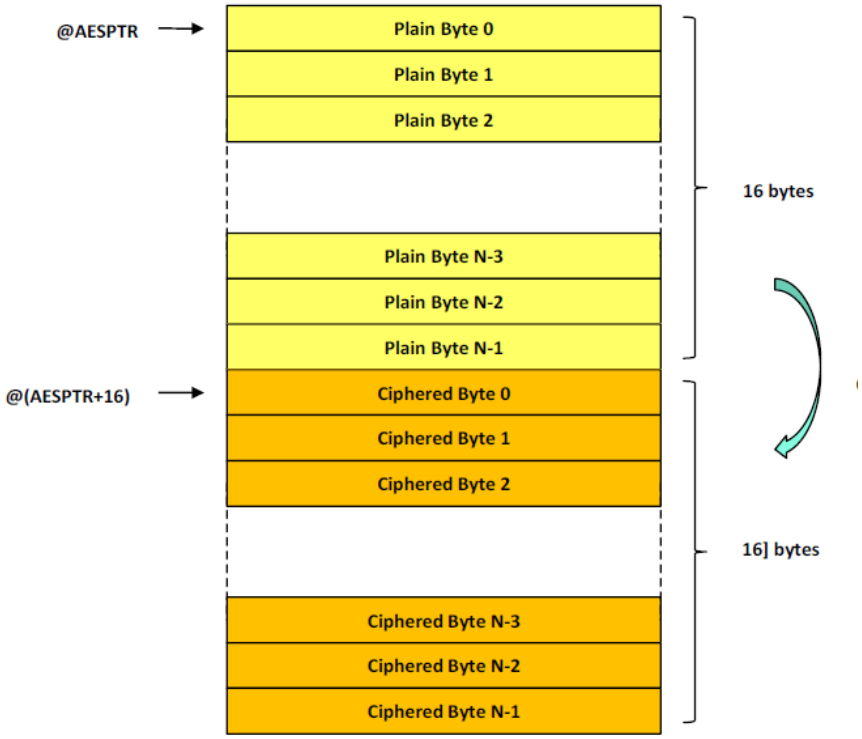
In this mode, firmware is stored in ciphertext on the Flash, and the code description information is used to verify its signature on the original plaintext.

16 AES

The SoC embeds AES module with encryption and decryption function. The input 128-bit plaintext in combination of key is converted into the final output ciphertext via encryption; the 128-bit ciphertext in combination of key can also be converted into 128-bit plaintext via decryption.

Software stores the block to encrypt, at AESPTR address in the SRAM. The block size to encrypt is always considered to equal 128-bit. Software defines the input key setting its value in AESKEY<> registers. Once the settings done, Software starts AES-128 use by writing a 1 in AES_START. On normal termination, the Software receives a crypt_irq. When the process ends the Software can find encrypted data at AESPTR+16 address as shown in figure below (considering byte address memory).

Figure 16-1 AES Address



The AES related registers are listed as following. The base address for the registers below is 0x80160000.

Table 16-1 AES Related Registers

Address Offset	Type	Description	Reset Value
0xb0	R/W	AESCNTL [0] AES_START [1] AES_MODE	0x00
0xb4	R/W	[7:0] AESKEY31_00, AES encryption 128-bit key. Bit 7 down to 0	0x00
0xb5	R/W	[7:0] AESKEY31_01, AES encryption 128-bit key. Bit 15 down to 8	0x00
0xb6	R/W	[7:0] AESKEY31_02, AES encryption 128-bit key. Bit 23 down to 16	0x00

Address Offset	Type	Description	Reset Value
0xb7	R/W	[7:0] AESKEY31_03, AES encryption 128-bit key. Bit 31 down to 24	0x00
0xb8	R/W	[7:0] AESKEY63_32_0, AES encryption 128-bit key. Bit 39 down to 32	0x00
0xb9	R/W	[7:0] AESKEY63_32_1, AES encryption 128-bit key. Bit 47 down to 40	0x00
0xba	R/W	[7:0] AESKEY63_32_2, AES encryption 128-bit key. Bit 55 down to 48	0x00
0xbb	R/W	[7:0] AESKEY63_32_3, AES encryption 128-bit key. Bit 63 down to 56	0x00
0xbc	R/W	[7:0] AESKEY95_64_0, AES encryption 128-bit key. Bit 71 down to 64	0x00
0xbd	R/W	[7:0] AESKEY95_64_1, AES encryption 128-bit key. Bit 79 down to 72	0x00
0xbe	R/W	[7:0] AESKEY95_64_2, AES encryption 128-bit key. Bit 87 down to 80	0x00
0xbf	R/W	[7:0] AESKEY95_64_3, AES encryption 128-bit key. Bit 95 down to 88	0x00
0xc0	R/W	[7:0] AESKEY127_96_0, AES encryption 128-bit key. Bit 103 down to 96	0x00
0xc1	R/W	[7:0] AESKEY127_96_1, AES encryption 128-bit key. Bit 111 down to 104	0x00
0xc2	R/W	[7:0] AESKEY127_96_2, AES encryption 128-bit key. Bit 117 down to 112	0x00
0xc3	R/W	[7:0] AESKEY127_96_3, AES encryption 128-bit key. Bit 127 down to 118	0x00
0xc4	R/W	[7:0] AESPTR0, Pointer to the memory zone where the block to cipher using AES-128 is stored.	0x00
0xc5	R/W	[7:0] AESPTR1	0x00

17 Public Key Engine (PKE)

The TLSR9228 embeds Public Key Engine (PKE) Standard Performance acceleration module and this section describes its function and use.

17.1 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 ~ 1024 bits, 32-bit step
- ECC (supports ECDH and ECDSA, prime field): 192, 224, 256 and 384 bits
- Ed25519/X25519
- SM2

17.2 Function Description

17.2.1 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.

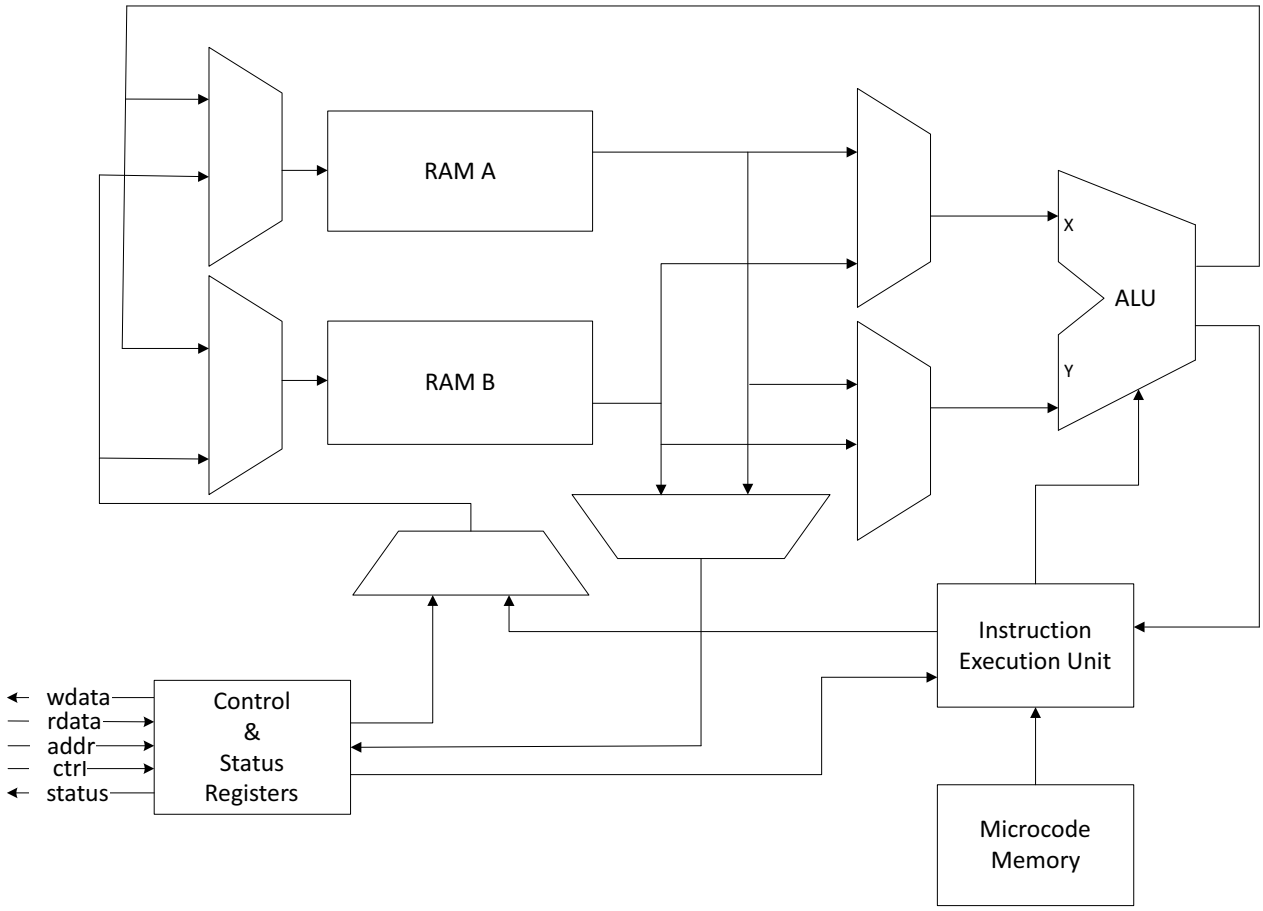
According to different register configurations, the PKE can complete the following operations of different precisions:

- RSA: length 512 ~ 1024 bits, 32-bit step
- ECC (prime field): length 192, 224, 256, and 384 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 17-1 Block Diagram of PKE Module



17.2.2 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 will encounter 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 17-1 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 will be larger than the space listed in the table and some intermediate variable storage is not open to the CPU.

Data will be stored in the mode of little-endian in RAM.

17.3 Register Description

The base address for the following PKE related registers is 0x80110000.

Table 17-2 PKE Related Registers

Offset	Type	Description	Default Value
0x00	W1S	PKE_CTRL [0]: Start Trigger PKE to start operation 0: No effect 1: PKE will start operation in the next clock cycle. The operations performed by PKE are decided by PKE_CFG and PKE_MC_PTR.	0x00
0x04	WR	PKE_CFG0 [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	RW	PKE_CFG1 [4:0] partial_radix_hi	0x01

Offset	Type	Description	Default Value
0x06	RW	PKE_CFG2 [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
0x08	RW	PKE_MC_PTR0 [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 will be ignored. During the operation of PKE, this field will update 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	RW	PKE_MC_PTR1 [3:0] addr_hi	0x00
0x0c	W0C	PKE_RISR [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

Offset	Type	Description	Default Value
0x10	RW	PKE_IMCR [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	RO	PKE_MISR [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	RO	PKE_RT_CODE [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
0x50	RW	PKE_EXE_CFG [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 will be converted to the Jacobian coordinate system before the operation 1: The input point is a point on the affine coordinate system	0x15

Offset	Type	Description	Default Value
		[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 will be 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 will be converted to the Jacobian coordinate system before the operation 1: The input point is a point on the affine coordinate system	
		[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 will be 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	

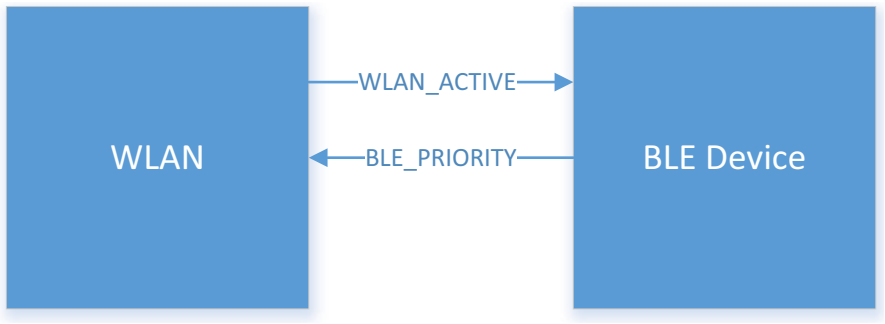
Offset	Type	Description	Default Value
0xfc	R	PKE_RBG_VERSION0 [3:0] mir: Sub version number. [7:4] mar: Main version number	0x10
0xfe	R	PKE_RBG_VERSION2 [7:0] project_lo {project_hi, project_lo} Project number	0x06
0xff	R	PKE_RBG_VERSION3 [7:0] project_hi	0xef

18 PTA Interface

The TLSR9228 supports a Packet Traffic Arbitration (PTA) interface to facilitate co-existence with 802.11 WLAN. The TLSR9228 supports a 2/3/4-wire BLE PTA 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 10-1 GPIO Pad Function Mux.

18.1 BLE Two-Wire Signaling

Figure 18-1 BLE Two-Wire Signaling



WLAN_ACTIVE:

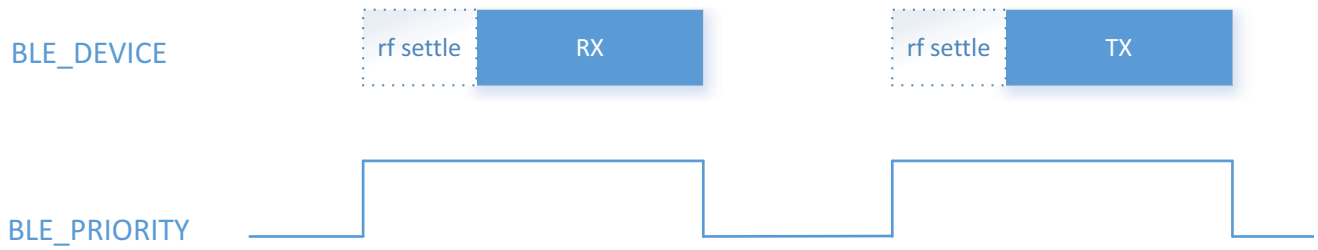
The WLAN_AVTIVE signal is asserted by WLAN controller when 802.11b/g packets are actively being transmitted or received. The BLE device avoids transmitting low-priority packets that are likely to cause interference with the 802.11b/g activity.

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.

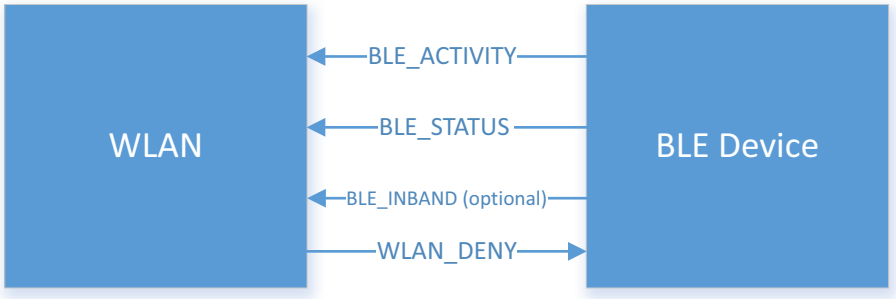
The WLAN_ACTIVE signal is judged by the software.

Figure 18-2 Example of BLE Two-Wire PTA Timing Diagram



18.2 BLE Three-Wire or Four-Wire Signaling

Figure 18-3 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.

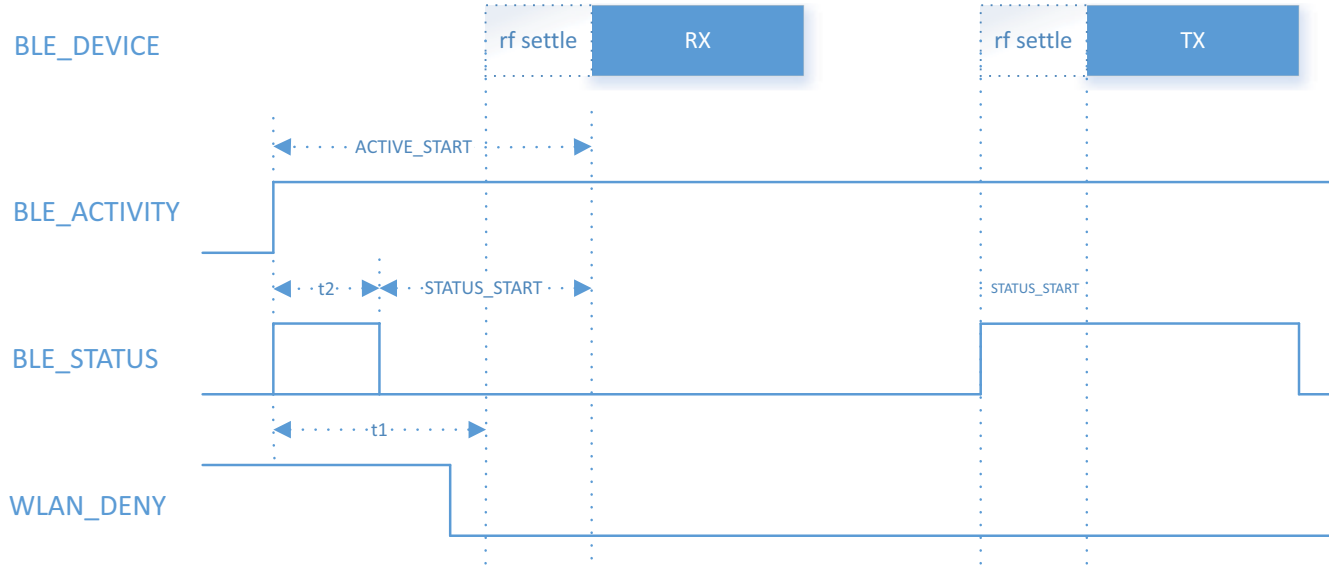
BLE_INBAND (optional):

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 18-4 Example of BLE Four-Wire PTA Timing Diagram



The two registers below are used to configure t1/t2:

Table 18-1 Register Configuration for t1/t2

Address	Name	Type	Description	Default Value
0xf12	r_t_coex_t1	RW	[7:0]: Corresponds to t1 in Figure 18-4 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 18-4 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

19 Quadrature Decoder

The TLSR9228 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.

19.1 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 0x80140242 (for channel a) / 0x80140243 (for channel b).

Table 19-1 Input Pin Selection

Address 0x80140242/0x80140243	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.*

19.2 Common Mode and Double Accuracy Mode

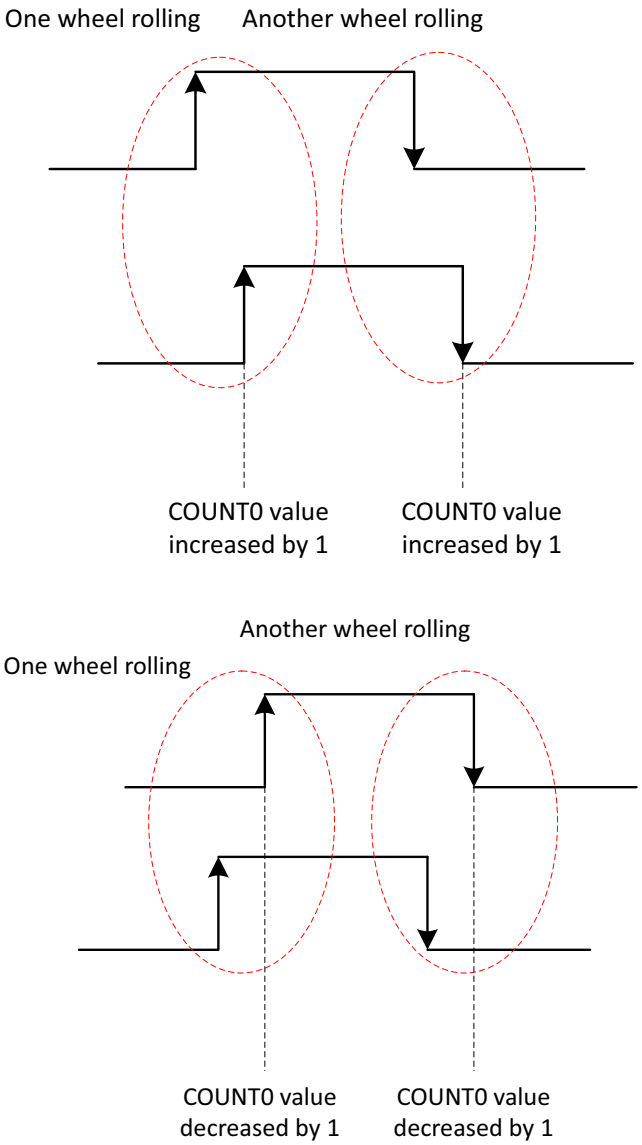
The QDEC embeds an internal hardware counter, which is not connected with bus.

Address 0x80140247 serves to select common mode or double accuracy mode, 0: common mode; 1: double accuracy mode.

For each wheel rolling step, two pulse edges (rising edge or falling edge) are generated.

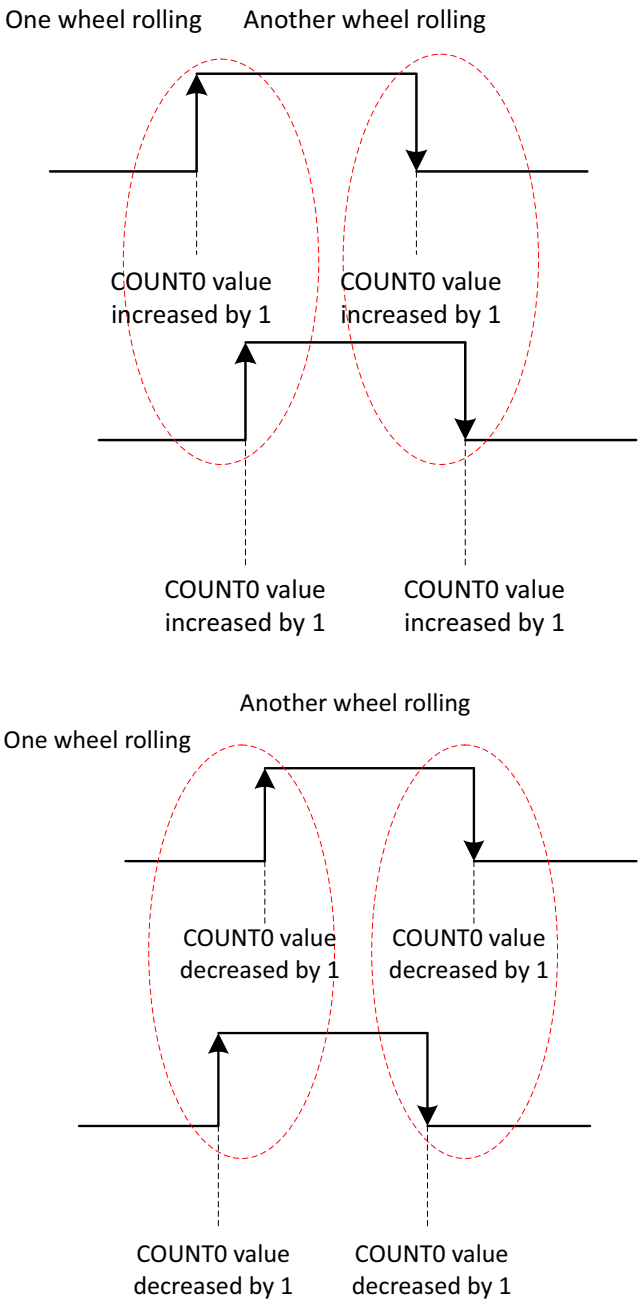
If address 0x80140247 is set to 0 to select common mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 only when the same rising/falling edges are detected from the two phase signals.

Figure 19-1 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 will be increased/decreased by 2 for one wheel rolling.

Figure 19-2 Double Accuracy Mode

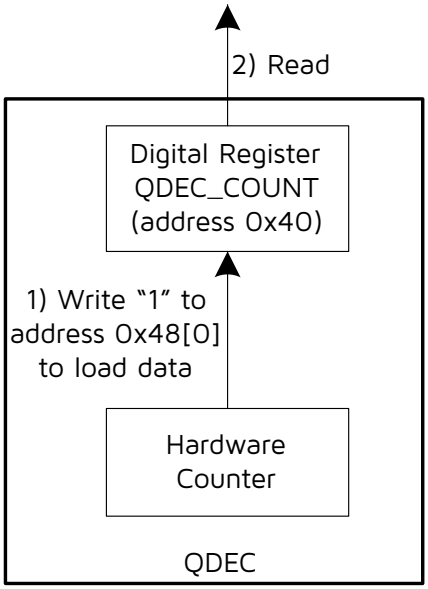


19.3 Read Real Time Counting Value

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 19-3 Read Real Time Counting Value



19.4 QDEC Reset

Address 0x801401f3[5] serves to reset the QDEC. The QDEC Counter value is cleared to zero.

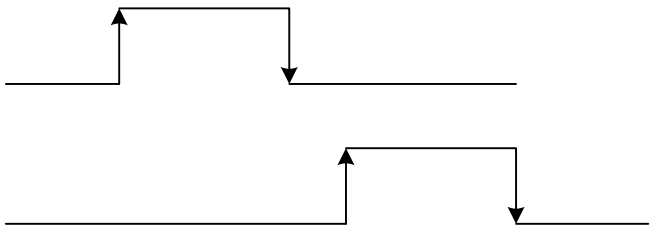
19.5 Other Configuration

The QDEC supports hardware debouncing. Address 0x41[2:0] serves to set filtering window duration. All jitter with period less than the value will be 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 19-4 Shuttle Mode



19.6 Timing Sequence

Figure 19-5 Timing Sequence Chart

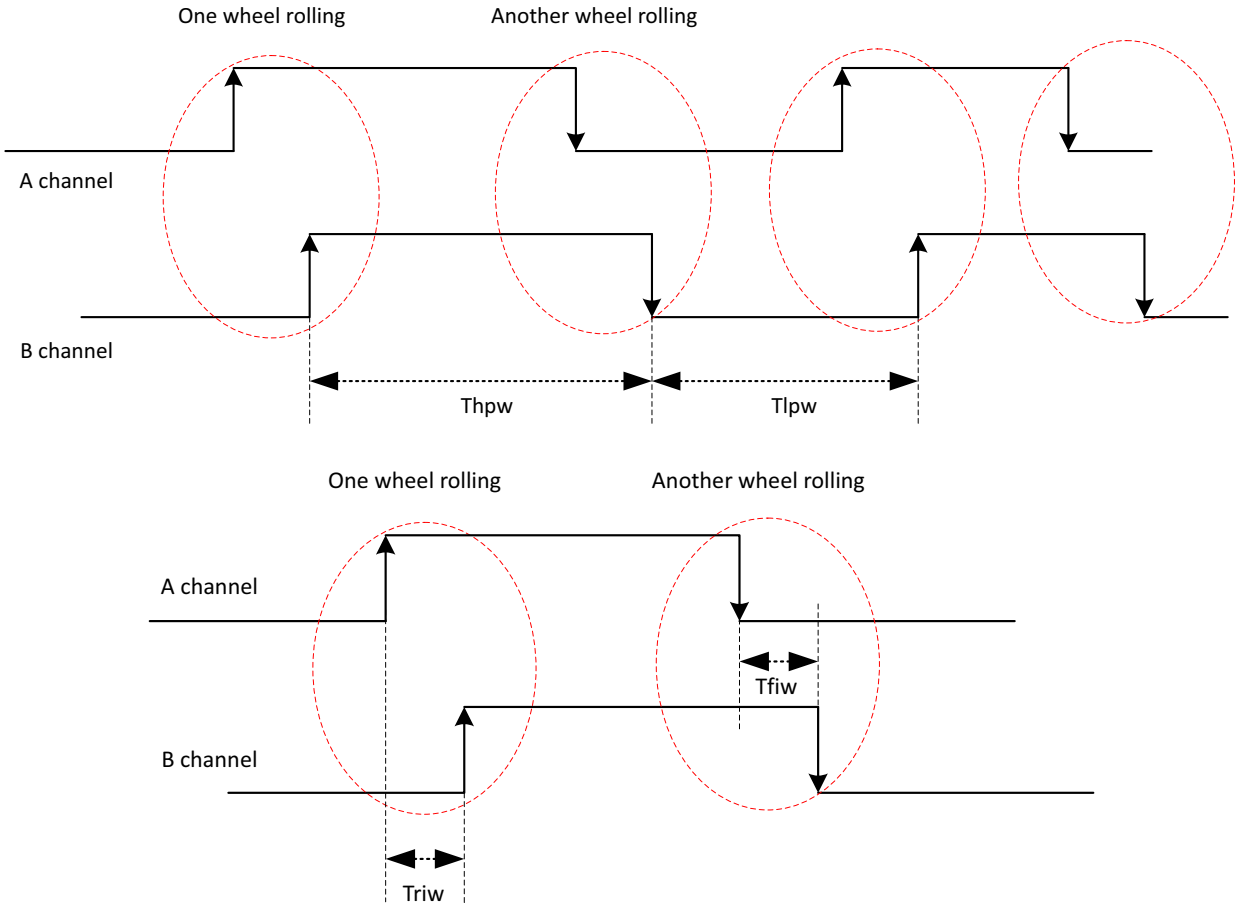


Table 19-2 Timing

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])

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])) will be 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.

19.7 Register Table

QDEC related registers are listed in the following table. The base address for the following registers is 0x80140240.

Table 19-3 Register Table for QDEC

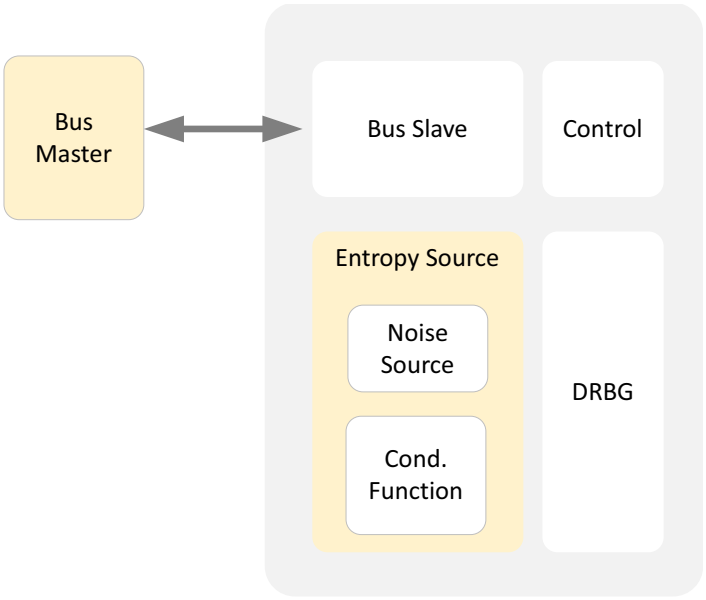
Address offset	Name	Type	Description	Reset Value
0x00	QDEC_COUNT0	R	QDEC Counting value (read to clear): Pulse edge number	0x00
0x01	QDEC_DBNTIME	RW	[5]: shuttle mode 1 - enable shuttle mode [4]: pola, input signal pola 0 - no signal is low, 1 - no signal is high [2:0]: filter time (can filter 2^n *clk_32k*2 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: {}	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: {}	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 will be 0.	0x00

20 True Random Number Generator (TRNG)

20.1 Model Overview

The True Random Number Generator (TRNG) module contains entropy source and post processing (Deterministic Random Bit Generators, DRBG). The entropy source is designed using Ring Oscillator (RO). The top block diagram of the random number generator is shown below.

Figure 20-1 Module Structure



20.2 Interrupt Description

The Random Bit Generator (RBG) module has the following interrupt sources:

- CPU reads RBG_DR without data
- Data valid

The above interrupts can be set by RBG_CR. By default, the data valid interrupt is enabled.

When the RBGEN of RBG_CR is low, the interrupt signal will not be cleared. Therefore, before enabling RBGEN, it is necessary to ensure that there is no previous interrupt signal, otherwise it will affect the next interrupt.

20.2.1 CPU Reads RBG_DR without Data

In order to prevent the CPU from reading the invalid data, the RBG can remind the CPU to read in such a situation when there is no valid random number. In order to avoid the CPU reading the empty data, it is recommended to read the RBG_FIFO_SR first every time to get the random number before the CPU gets data in the current FIFO to avoid invalid data.

The CPU can clear the interrupt by writing 1 to ERERR in RBG_SR. If the write is successful, the interrupt will be cleared. When the above situation occurs again, the interrupt will be valid again.

20.2.2 Data Valid

RBG provides two ways to output data. When the interrupt is enabled, the random number can be read by the way of interrupting. In this design, the data in the corresponding FIFO will only be pulled up after the threshold is reached, thus the CPU can obtain multiple data at once. The threshold can be set by RBG_FIFO_CR. The CPU can clear the interrupt by writing 1 to DRDY of RBG_SR. If the write is successful, the interrupt will be pulled down. The interrupt is pulled high again when the data in the FIFO reaches the threshold again.

It is important to note that the interrupt will only be pulled up when the amount of data in the FIFO reaches the threshold. Therefore, the data in the FIFO exceeds the threshold firstly and then RBG module pulls up the interrupt. When the CPU doesn't obtain data or have obtained data but the amount of data remaining in the FIFO is still larger than the threshold, then clear the interrupt. Although the data in the FIFO is still larger than the threshold, it will not be interrupted.

In addition, the CPU can use the RBG_FIFO_SR register to view the remaining data in the FIFO. It can also use this method to obtain a random number. Check the RBG_FIFO_SR register when the random number is needed and the number of random numbers indicated by the register can be fetched at one time. If the rate at which the CPU handles random numbers is slower than the rate at which RBG random numbers are generated, it is generally not recommended to use interrupt to obtain random numbers.

20.3 Usage Procedure

20.3.1 Normal Operation

Turn off the RBG module first after the CPU works normally, that is to set RBGEN of the RBG_CR to 0. Then it can be configured and write 1 to RBGEN after the configuration is complete to make it work normally.

The CPU can configure RBG module by configuring RBG_CR, RBG_FIFO_CR and other optional configuration registers.

When writing 1 to RBGEN in RBG_CR, the modification of the value of the above register will not affect the RBG. Therefore, when configuring, set the RBGEN in the RBG_CR register after configuring other registers to enable the OSR_RBG module.

TRBG and DRBG can be switched by modifying RBG_RTCR during the operation to meet different usage environments.

20.3.2 Entropy Source

In this design, the random number generator module uses RO RNG as the entropy source. RO RNG contains modules such as random source and post-processing. RO RNG has four independent RO entropy sources. Each entropy source can choose to use its own RO CLK as the sampling clock or select the system clock as the sampling clock. The selection is determined by the input of I_rbg_sclk_sel, which is high for the system clock and low for the internal RO CLK. All RO enable signals are open at the same time and some of the ROs can be turned on or off by controlling the register.

20.4 Register Description

TRNG related registers are listed in the following table. The base address for the following registers is 0x80101800.

Table 20-1 TRNG Related Register

Offset	Type	Description	Reset Value
0x00	RW	TRNG_CR0 [0]: Random bit generator enable. [1]: Each bit states enable for one RO SOURCE0 [2]: Each bit states enable for one RO SOURCE1 [3]: Each bit states enable for one RO SOURCE2 [4]: Each bit states enable for one RO SOURCE3	0x07
0x04	RW	TRNG_RTCR [0]: Mode select. 0: TRBG without post-processing; 1: TRBG with post-processing	0x00
0x08	R	RBG_SR [0]: Data ready. Data valid indicating bit, 0: random data not ready; 1: random data ready.	0x00
0x0c	R	RBG_DR0, rbg data	0x00
0x0d	R	RBG_DR1, rbg data	0x00
0x0e	R	RBG_DR2, rbg data	0x00
0x0f	R	RBG_DR3, rbg data	0x00
0x10	R	RBG_VERSION0 [3:0]: Sub version number [7:4]: Main version number	0x01
0x12	RO	RBG_VERSION2, PROJECT number low	0x3a
0x13	RO	RBG_VERSION3, PROJECT number high	0xef
0x80	RW	RO_CR1_0 RO enable of RO SOURCE1. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 1.	0xff
0x81	RW	RO_CR1_1 RO enable of RO SOURCE1. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 1.	0xff
0x82	RW	RO_CR0_0 RO enable of RO SOURCE0. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 0.	0xff

Offset	Type	Description	Reset Value
0x83	RW	RO_CR0_1 RO enable of RO SOURCE0. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 0.	0xff
0x84	RW	RO_CR3_0 RO enable of RO SOURCE3. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 3.	0x00
0x85	RW	RO_CR3_1 RO enable of RO SOURCE3. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 3.	0x00
0x86	RW	RO_CR2_0 RO enable of RO SOURCE2. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 2.	0x00
0x87	RW	RO_CR2_1 RO enable of RO SOURCE2. Each bit controls one RO. In total, there are 16 ROs in RW RO SOURCE 2.	0x00
0x88	RW	FSEL [1:0]: RO sampling clock frequency division selection. 00: 4 frequency division, 01: 8 frequency division, 10: 16 frequency division, 11: 32 frequency division.	0x03