TRIDENT IoT

T32CZ20

Sub-GHz Z-Wave RF SoC

Description

The T32CZ20 is an ultra-low power, high performance Sub-GHz wireless SoC supporting Z-Wave to facilitate sensor network, building automation, smart locks and smart home applications.

Harnessing the power of ARM Cortex®-M33 and TrustZone technology, the T32CZ20 SoC establishes secure enclaves to safeguard critical data and processes. The integration of Physical Unclonable Function (PUF) adds an extra layer of security by leveraging unique physical variations within the chip to generate device-specific cryptographic keys, enhancing resistance to cloning. Complementing these features, the True Random Number Generator (TRNG) ensures the generation of unpredictable and truly random numbers, vital for cryptographic operations and secure communications.

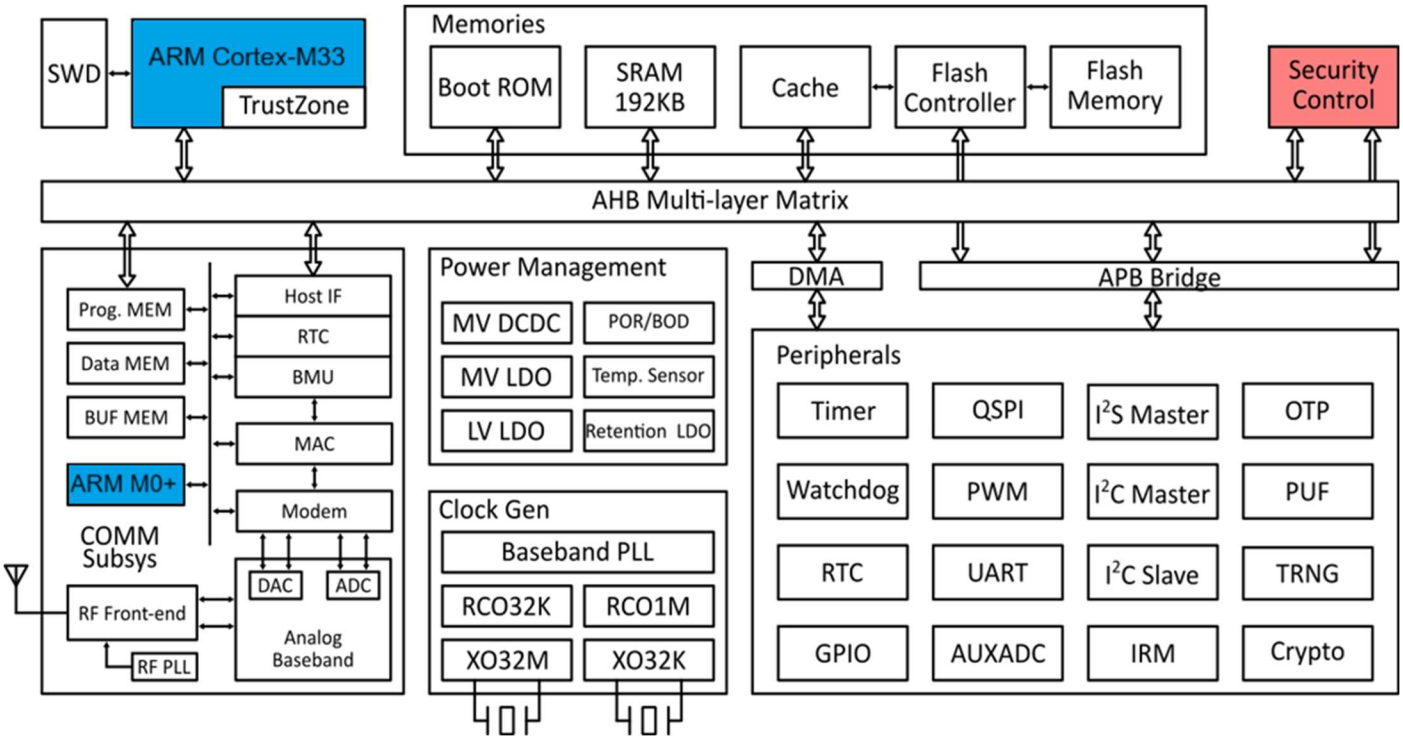


Figure 1 T32CZ20 Block Diagram

Table of Contents

1. Introduction

The T32CZ20 is an ultra-low power, high performance Sub-GHz wireless SoC for the Z-Wave Long Range wireless communication protocol network for automation, smart locks and smart home, and security applications. It features an integrated ARM Cortex®-M33 core with a rich set of peripherals and an integrated wireless transceiver with its own dedicated ARM Cortex®-M0+ core. It is efficient enough for low-power end-devices and powerful enough for integrating Z-Wave Controllers into base stations, hubs, bridges, security panels etc. This makes the T32CZ20 ideal for battery-powered devices and applications where both performance and energy efficiency are essential.

The Z-Wave Long Range communication protocol propagates better through walls and has much less interference than the crowded 2.4GHz band using spread spectrum technology to transmit at much higher power and still be well within RF regulatory limits dynamically adapting RF power when needed to conserve power and has much better latency and overall downstream performance and capabilities than LoRa.

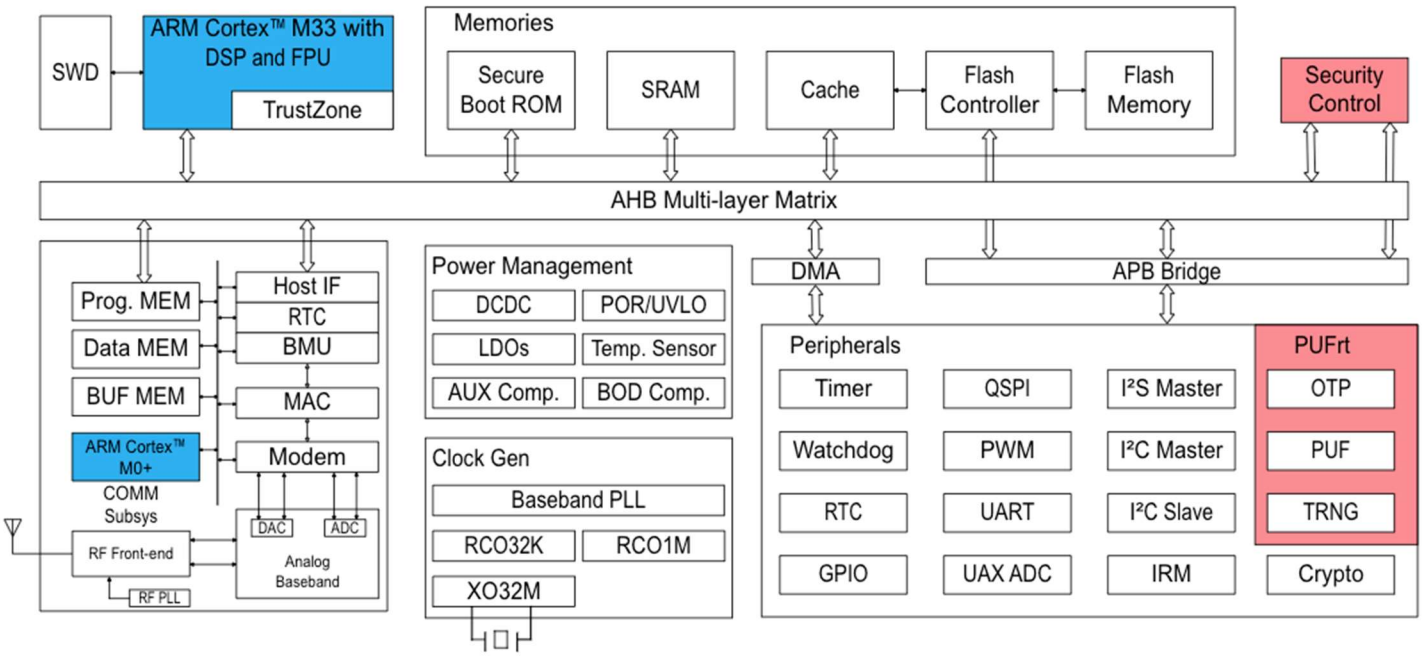


Figure 1-1 CZ20 Block Diagram

There are two subsystems inside the CZ20 with 288KB total SRAM:

- host subsystem with high to 64 MHz 32-bit ARM Cortex-M33, 192KB SRAM and up to 2MB Flash for application requirements
- communication subsystem (COMM Subsys) with ARM Cortex-M0+ MCU and a HW MAC accelerator utilizing 96KB program SRAM for dynamic switching between standard protocols to support multiple simultaneous unique links.

Through the whole document, Host CPU indicates Cortex-M33 and MCU indicates Cortex-M0+.

1.1. Features

- Host CPU
  - ARM Cortex-M33
  - 32 KB Boot ROM with Secure Boot function
  - 192 KB SRAM
  - Clock frequencies support up to 64 MHz
  - XIP (eXecute In Place) Flash with Cache
- Peripherals
  - DMA x2
  - General Timer x3
  - Slow clock Timer x2
  - Watchdog Timer
  - RTC Timer
  - GPIO up to 22
  - (Q)SPI x2
  - PWM x5
  - UART x3
  - I^2S Master
  - I^2C Master x2
  - I^2C Slave
  - IRM (Infrared Modulator)
  - BOD Comparator
  - AUX Comparator
  - AUX ADC up to 4 channels
- Security Peripherals
  - Secure OTP
  - PUF (Physical Unclonable Function)
  - TRNG
  - Cryptographic accelerator
    - AES
    - SHA256
    - ECC/ECDSA/ECDH
- Clocks
  - Baseband PLL up to 72MHz
  - 32 MHz crystal oscillator
  - 1 MHz RC oscillator
  - 32 KHz RC oscillator
  - 32 KHz crystal oscillator (supported only in certain packages)
- Low-power modes
  - CPU Wait-for-interrupt (WFI)
  - Sleep mode
  - Deep sleep mode
  - Deep power-down mode
- Power Managements
  - POR
  - BOD
  - Buck DCDC
  - LDOs
- COMM Subsys
  - ARM Cortex-M0+
  - 96 KB SRAM for program memory (can be shared with Cortex-M33)
  - 8 KB SRAM for data memory
  - 8 KB SRAM for TX/RX data buffers
  - Multi-protocol Modem and Link control
  - Independent AES/CCM engine for on-the-fly encryption/decryption

2. Host CPU

The Cortex-M33 CPU is a high performance 32-bit processor designed for the microcontroller market. It offers significant benefits to developers, including:

- outstanding processing performance combined with fast interrupt handling
- efficient processor core, system and memories
- ultra-low power consumption with integrated sleep modes
- platform security robustness with TrustZone

2.1. CPU Configuration Supported

Table 2-1 Configurable options of Cortex-M33 in CZ20

	Description	Implementation
FPU	Floating Point Unit	YES
DSP	DSP Extension	YES
SECEXT	Security Extension	YES
SAU	Security Attribution Unit	YES (Use platform-defined security control unit instead)
NUMIRQ	Number of IRQ inputs	48
IRQLVL	Control bits of IRQ priority level	3 (8 levels)
DBGLVL	Debug level	2 (Full set of debug resources)
ITM	Instrumentation Trace Macrocell	No
ETM	Embedded Trace Macrocell	No
MTB	Micro Trace Buffer	No
WIC	Wake-up Interrupt Controller	Yes
Endianness	Memory system endianness	Little endian
JTAG/SWD	Debug Interface	SWD only

2.2. Memory Map

Table 2-2 Memory Map

Modules	Size	Non-secure Address	Secure Address
Flash	128MB	0x00000000~0x07FFFFFF	0x10000000~0x17FFFFFF
Boot ROM	32KB	N/A	0x18000000~0x18007FFF
Modules	**Size**	**Non-secure Address**	**Secure Address**

Reserved		0x08008000~0x0FFFFFFF	0x18008000~0x1FFFFFFF
RAM 0	32KB	0x20000000~0x20007FFF	0x30000000~0x30007FFF
RAM 1	32KB	0x20008000~0x2000FFFF	0x30008000~0x3000FFFF
RAM 2	32KB	0x20010000~0x20017FFF	0x30010000~0x30017FFF
RAM 3	32KB	0x20018000~0x2001FFFF	0x30018000~0x3001FFFF
RAM 4	32KB	0x20020000~0x20027FFF	0x30020000~0x30027FFF
RAM 5	16KB	0x20028000~0x2002BFFF	0x30028000~0x3002BFFF
RAM 6	16KB	0x2002C000~0x2002FFFF	0x3002C000~0x3002FFFF
Reserved		0x20030000~0x2003FFFF	0x30030000~0x3003FFFF
COMM Subsys RAM	96KB	0x20040000~0x20057FFF	0x30040000~0x30057FFF
Reserved		0x20030000~0x2FFFFFFF	0x30030000~0x3FFFFFFF
System Control	4KB	0x40000000~0x40000FFF	0x50000000~0x50000FFF
GPIO Control	4KB	0x40001000~0x40001FFF	0x50001000~0x50001FFF
Reserved		0x40002000~0x40002FFF	0x50002000~0x50002FFF
Security Control	4KB	N/A	0x50003000~0x50003FFF
RTC Timer	4KB	0x40004000~0x40004FFF	0x50004000~0x50004FFF
Deep Power-down Control	4KB	0x40005000~0x40005FFF	0x50005000~0x50005FFF
Power Management	4KB	0x40006000~0x40006FFF	0x50006000~0x50006FFF
Reserved		0x40007000~0x40007FFF	0x50007000~0x50007FFF
Reserved		0x40008000~0x40008FFF	0x50008000~0x50008FFF
Flash Control	4KB	0x40009000~0x40009FFF	0x50009000~0x50009FFF
Timer 0	4KB	0x4000A000~0x4000AFFF	0x5000A000~0x5000AFFF
Timer 1	4KB	0x4000B000~0x4000BFFF	0x5000B000~0x5000BFFF
Timer 2	4KB	0x4000C000~0x4000CFFF	0x5000C000~0x5000CFFF
Slow-clock Timer 0	4KB	0x4000D000~0x4000DFFF	0x5000D000~0x5000DFFF
Slow-clock Timer 1	4KB	0x4000E000~0x4000EFFF	0x5000E000~0x5000EFFF
Reserved		0x4000F000~0x4000FFFF	0x5000F000~0x5000FFFF
Watchdog Timer	4KB	0x40010000~0x40010FFF	0x50010000~0x50010FFF
Reserved		0x40011000~0x40011FFF	0x50011000~0x50011FFF
UART 0	4KB	0x40012000~0x40012FFF	0x50012000~0x50012FFF
UART 1	4KB	0x40013000~0x40013FFF	0x50013000~0x50013FFF
Reserved		0x40014000~0x40014FFF	0x50014000~0x50014FFF
Reserved		0x40015000~0x40015FFF	0x50015000~0x50015FFF
Modules	**Size**	**Non-secure Address**	**Secure Address**
Reserved		0x40016000~0x40016FFF	0x50016000~0x50016FFF
Reserved		0x40017000~0x40017FFF	0x50017000~0x50017FFF

I²C Slave	4KB	0x40018000~0x40018FFF	0x50018000~0x50018FFF
Reserved		0x40019000~0x40019FFF	0x50019000~0x50019FFF
COMM Subsys	4KB	0x4001A000~0x4001AFFF	0x5001A000~0x5001AFFF
Reserved		0x4001B000~0x4001BFFF	0x5001B000~0x5001BFFF
Reserved		0x4001C000~0x4001CFFF	0x5001C000~0x5001CFFF
BOD Comparator	4KB	0x4001D000~0x4001DFFF	0x5001D000~0x5001DFFF
AUX Comparator	4KB	0x4001E000~0x4001EFFF	0x5001E000~0x5001EFFF
Reserved		0x4001F000~0x4001FFFF	0x5001F000~0x5001FFFF
QSPI 0	4KB	0x40020000~0x40020FFF	0x50020000~0x50020FFF
QSPI 1	4KB	0x40021000~0x40021FFF	0x50021000~0x50021FFF
Reserved		0x40022000~0x40022FFF	0x50022000~0x50022FFF
Reserved		0x40023000~0x40023FFF	0x50023000~0x50023FFF
IRM	4KB	0x40024000~0x40024FFF	0x50024000~0x50024FFF
UART 2	4KB	0x40025000~0x40025FFF	0x50025000~0x50025FFF
PWM	4KB	0x40026000~0x40026FFF	0x50026000~0x50026FFF
Reserved		0x40027000~0x40027FFF	0x50027000~0x50027FFF
xDMA	4KB	0x40028000~0x40028FFF	0x50028000~0x50028FFF
DMA 0	4KB	0x40029000~0x40029FFF	0x50029000~0x50029FFF
DMA 1	4KB	0x4002A000~0x4002AFFF	0x5002A000~0x5002AFFF
I²C Master 0	4KB	0x4002B000~0x4002BFFF	0x5002B000~0x5002BFFF
I²C Master 1	4KB	0x4002C000~0x4002CFFF	0x5002C000~0x5002CFFF
I²S	4KB	0x4002D000~0x4002DFFF	0x5002D000~0x5002DFFF
Reserved		0x4002E000~0x4002EFFF	0x5002E000~0x5002EFFF
AUX ADC	4KB	0x4002F000~0x4002FFFF	0x5002F000~0x5002FFFF
Software IRQ 0	4KB	0x40030000~0x40030FFF	0x50030000~0x50030FFF
Software IRQ 1	4KB	0x40031000~0x40031FFF	0x50031000~0x50031FFF
Reserved		0x40032000~0x4043FFFF	0x50032000~0x5043FFFF
PUFrt	32KB	0x40044000~0x40047FFF	0x50044000~0x50047FFF
Crypto Engine	16KB	0x60000000~0x60003FFF	0x70000000~0x70003FFF

2.3. Memory Banks

Memories are divided into four banks that are listed in Table 2-3. Memory accesses in different banks can be occurred concurrently. This can significantly boost the performance of the memory system.

Table 2-3 Memory Banks

Banks	Size	Memory
0	64KB	RAM 0 RAM 1
1	64KB	RAM 2 RAM 3
2	32KB	RAM 4
3	32KB	RAM 5 RAM 6

2.4. Interrupts

Table 2-4 Interrupt Table

IRQ Index	Source	Active State
0	GPIO	programmable
1	Timer 0	level high
2	Timer 1	level high
3	Timer 2	level high
4	Slow-clock Timer 0	level high
5	Slow-clock Timer 1	level high
6	Watchdog Timer	level high
7	RTC Timer	level high
8	N/A	
9	Software IRQ 0	level high
10	Software IRQ 1	level high
11	N/A	
12	DMA 0	level high
13	DMA 1	level high
14	N/A	
15	N/A	
16	UART 0	level high
17	UART 1	level high
18	UART 2	level high
19	N/A	
20	IRM	level high
21	I^2C Master 0	level high
22	I^2C Master 1	level high
23	I^2C Slave	level high

24	N/A	
25	QSPI 0	level high
26	QSPI 1	level high
27	N/A	
28	N/A	
29	I^2S	level high
30	N/A	
31	N/A	
32	PWM 0	level high
33	PWM 1	level high
34	PWM 2	level high
35	PWM 3	level high
36	PWM 4	level high
37	N/A	
38	N/A	
39	Flash Control	level high
40	PUFrt	level high
41	Crypto Engine	level high
42	BOD Comparator	programmable
43	N/A	
44	Security Control	level high
45	COMM Subsys	level high
46	AUX ADC	level high
47	AUX Comparator	programmable

3. Clocks

There are several clock sources in CZ20 and they are summarized in Table 3-1.

Table 3-1 Clock Sources of CZ20

Clock Name	Type	Typical Frequency	Comment
xtal_clk	Crystal oscillator	32 MHz	The default clock for Host CPU, memories and peripherals.
pll_clk	PLL	36/40/48/64 MHz	A higher frequency clock for Host CPU and memories.
rco1m	RC oscillator	921.6 KHz	An optional clock for UARTs operating in Sleep mode.
rco32k	RC oscillator	32 KHz	The default clock for slow-clock timers.
xo32k	Crystal oscillator	32.768 KHz	A high accuracy clock for replacing rco32k.

3.1. The 32 MHz Crystal Oscillator

Since CZ20 is a SoC chip for wireless communications, a crystal oscillator of 32 MHz (xtal_clk) is required for establishing communication links correctly. It is also the default clock for Host CPU, memories and peripherals after power on.

During low-power modes (refer to Chapter 5 Low Power Modes for details), the xtal_clk is turned off to reduce the current consumption. After waking up from low-power modes, it will be turned on again. Those on/off procedures are controlled by Hardware automatically and is transparent to Software.

3.2. The Baseband PLL

The Baseband PLL (pll_clk) provides a higher clock frequency than 32MHz that can boost the performance of Host CPU and memories. Available clock options of the Baseband PLL are 36, 40, 48, 64, and 72 MHz.

Like the xtal_clk, if pll_clk is enabled, it will be turned off in low-power modes and turned on after waking up from low-power modes automatically by Hardware.

It is very important to follow below procedures to enable the Baseband PLL or to change its frequency, otherwise the unstable clock may result in malfunction.

- Switch the CPU clock to xtal_clk
- Disable Baseband PLL
- Set new frequency of Baseband PLL
- Enable Baseband PLL
- Wait 100u
- Switch the CPU clock to pll_clk

The frequency of pll_clk is selected by the register, cfg_bbpll_freq, which is summarized in Table 3-2.

Table 3-2 Frequency Selections of Baseband PLL

cfg_bbpll_freq[2:0]	Baseband PLL Frequency (MHz)
000b	48
001b	64
010b	Reserved
011b	Reserved
100b	Reserved
101b	Reserved
110b	36
111b	40

3.3. The 1 MHz RC Oscillator

This 1 MHz RC Oscillator (rco1m) is primarily used for UARTs that are required to operate normally in low-power modes.

By default, all UARTs are clocked by the xtal_clk after power-on. Since the xtal_clk is turned off in low-power modes, the UART interface can't receive any data in low-power modes. This may not be a problem because for most applications CZ20 can decide when to communicate through the UART interface.

For applications that require UART communications at any time, we can let CZ20 not enter low-power modes with the cost of higher power consumptions. Alternatively, we can change the UART clock to rco1m so that the UART interface can still receive data in low-power modes. For this situation, we must first calibrate the frequency of rco1m to 921.6 KHz and the maximum baud rate of UART is limited to 115200.

The rco1m is not restricted to the UART use and the Host CPU can use this clock also.

3.4. The 32 KHz RC Oscillator

The 32 KHz RC oscillator (rco32k) is used for slow-clock timers, like RTC timer and slow-clock Timer 0 and 1. The rco32k is kept power-on in Sleep Mode but is powered off in Deep Sleep Mode by default. For those applications required RTC Timer running in Deep Sleep Mode, the control register, cfg_ds_rco32k_off, shall be set to 0b.

3.5. The 32 KHz Crystal Oscillator

CZ20 also supports the optional 32 KHz crystal oscillator (xo32k) that can replace the rco32k if a high-accuracy slow clock is needed.

Because the xo32k requires external pins to connect to the crystal, the support of xo32k is only available in certain types of packages.

3.6. Internal Clocks

There are three major internal clocks that can be selected from above clock sources and they are hclk, per_clk and slow_clk. Some peripherals have their dedicated clocks derived from them. Table 3-3 summarizes those internal clocks and their usages. The structure of internal clocks is illustrated in Figure 3-1.

Table 3-3 Summary of Internal Clocks

Internal Clock	Description	Usage
hclk	AHB clock	Host CPU AHB/APB bus Memories DMAx Flash I2S Crypto Engine
slow_clk	Slow clock	RTC Timer Slow-clock Timers
per_clk	Peripheral clock	Watchdog Timer I2C Master QSPIx IRM
pwmx_clk	PWM clocks, where x=0,1,2,3,4.	PWMx
timerx_clk	Timer clocks, where x=0,1,2.	Timerx
uartx_clk	UART clocks, where x=0,1,2.	UARTx

Although hclk and per_clk can be selected independently, some combinations are undesired. It is very important that the frequency of hclk must be always great or equal than that of per_clk.

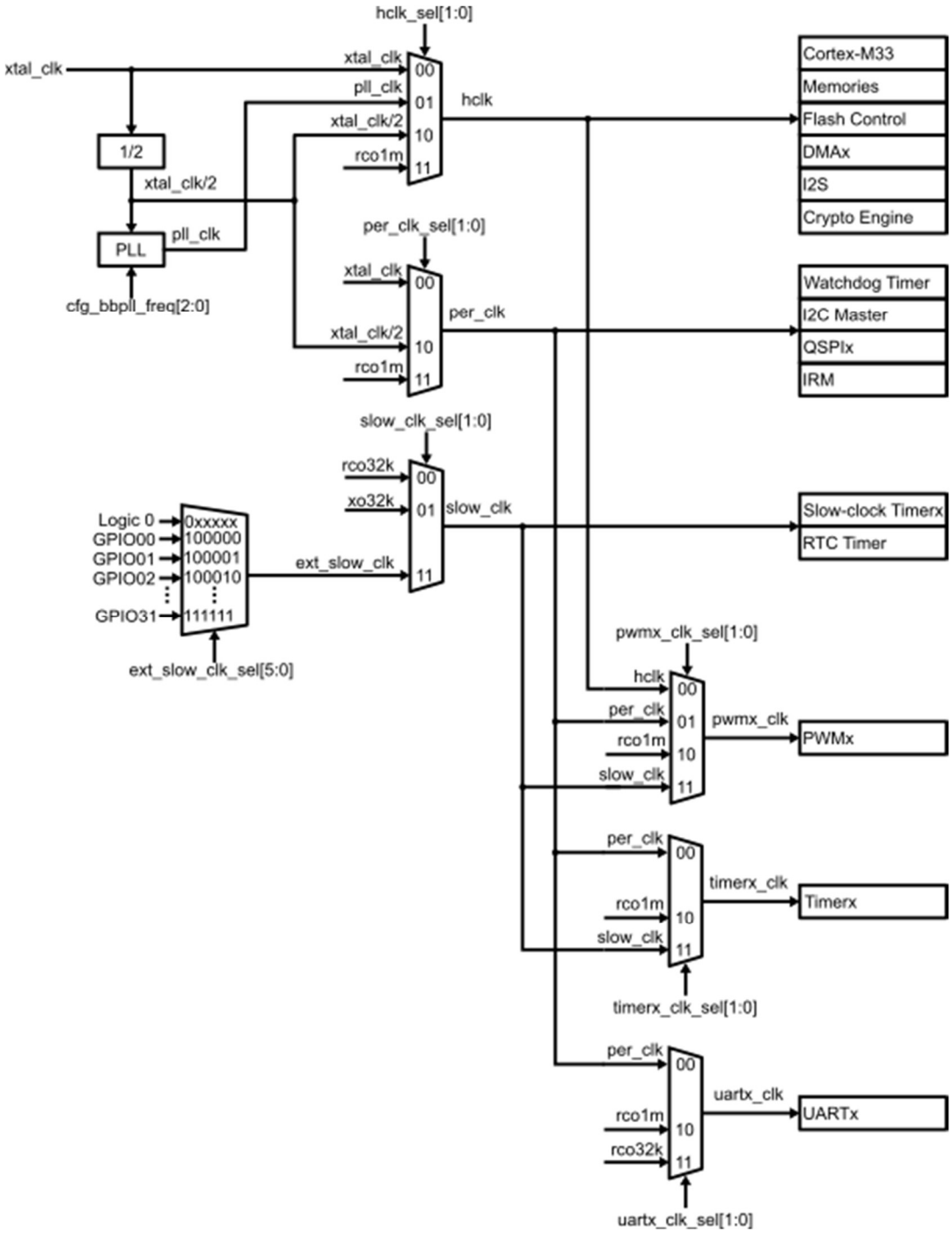


Figure 3-1 The Structure of Internal Clocks

3.6.1. Selections of hclk

The hclk is used for those that require performance, such like the Host CPU, the bus system, the memories, and etc. The selection of hclk is controlled by the register, hclk_sel, which is listed in Table 3-4.

Table 3-4 Selections of hclk

hclk_sel[1:0]	hclk source
00b	xtal_clk
01b	pll_clk
10b	xtal_clk/2
11b	rco1m

3.6.2. Selections of slow_clk

The slow clock is used for slow-clock timers that is controlled by the register, slow_clk_sel, and is listed in Table 3-5.

Table 3-5 Selections of slow_clk

slow_clk_sel[1:0]	slow_clk source
00b	rco32k
01b	xo32k
10b	Reserved
11b	External slow clock (from GPIO)

Besides the rco32k and xo32k, the slow_clk can be come from the external clock source through the GPIO input. The register, ext_slow_clk_sel, controls which GPIO is used that is summarized in Table 3-6.

Table 3-6 Select the GPIO for External Slow Clock

Register	Descriptions
ext_slow_clk_sel[5]	External slow clock input enable. 0b Disable 1b Enable
ext_slow_clk_sel[4:0]	Select which GPIOxx is used as the external slow clock input. The value of 0x00 selects GPIO00 , the value 0x01 selects GPIO01 and so on.

3.6.3. Selections of per_clk

The per_clk is used for Watchdog Timer, Timer 0/1/2, I2C Master 0/1, QSPI 0/1 and IRM. The control register, per_clk_sel, selects the clock source of per_clk that is summarized in Table 3-7.

Table 3-7 Selections of per_clk

per_clk_sel[1:0]	per_clk source
00b	xtal_clk
01b	Reserved
10b	xtal_clk/2
11b	rco1m

3.6.4. Selections of pwmx_clk

There are five PWM modules and each of them has the dedicated clock selection. The control register, pwmx_clk_sel, selects the clock of the PWMx module.

Table 3-8 Selections of pwmx_clk_sel

pwmx_clk_sel[1:0]	pwmx_clk source
00b	hclk
01b	per_clk
10b	rco1m
11b	slow_clk

3.6.5. Selections of timerx_clk

There are three Timer modules and each of them has the dedicated clock selection. The control register, timerx_clk_sel, selects the clock of the Timerx module.

Table 3-9 Selections of timerx_clk_sel

timerx_clk_sel[1:0]	timerx_clk source
00b	per_clk
01b	Reserved
10b	rco1m
11b	slow_clk

3.6.6. Selections of uartx_clk

There are three UART modules and each of them has the dedicated clock selection. The control register, uartx_clk_sel, selects the clock of the UARTx module.

Table 3-10 Selections of uartx_clk_sel

uartx_clk_sel[1:0]	uartx_clk source

00b	per_clk
01b	Reserved
10b	rco1m
11b	rco32k

If the rco1m is selected for the UART clock, the frequency of rco1m shall be calibrated to 921.6 KHz and the maximum baud rate is limited to 115200. If the rco32k is selected for the UART clock, the frequency of rco32k shall be calibrated to 38.4 KHz and the maximum baud rate is limited to 9600.

Because either rco1m or rco32k is still alive in low-power modes, the UART can receive data correctly and wake up the Host CPU in low-power modes when these clocks are used.

3.7. Registers of Clock Control

The clock control registers are resident in the System Control space whose base address is 0x50000000 for the secure partition or 0x40000000 for the non-secure partition.

- Offset 0x04: SYS_CLK_CTRL0

Signal	Bits	Default	R/W	Description
Reserved	[31:16]			
cfg_bbpll_en	[15]	0b	R/W	Baseband PLL enable. 0b disable 1b enable
reserved	[14:11]			
cfg_bbpll_freq	[10:8]	000b	R/W	The frequency selection of Baseband PLL 000b 48 MHz 001b 64 MHz 110b 36 MHz 111b 40 MHz Others Reserved
slow_clk_sel	[7:6]	00b	R/W	The selection of slow_clk. 00b rco32k 01b xo32k 10b reserved 11b external clock source from GPIOs
reserved	[5:4]			
per_clk_sel	[3:2]	00b	R/W	The selection of per_clk. 00b xtal_clk 01b reserved 10b xtal_clk/2 11b rco1m
hclk_sel	[1:0]	00b	R/W	The selection of hclk.

				00b xtal_clk 01b pll_clk 10b xtal_clk/2 11b rco1m

- Offset 0x08: SYS_CLK_CTRL1

Signal	Bits	Default	R/W	Description
timer2_clk_sel	[31:30]	00b	R/W	The clock selection of Timer 0. 00b per_clk 01b reserved 10b rco1m 11b slow_clk
timer1_clk_sel	[29:28]	00b	R/W	The clock selection of Timer 1. 00b per_clk 01b reserved 10b rco1m 11b slow_clk
timer0_clk_sel	[27:26]	00b	R/W	The clock selection of Timer 0 00b per_clk 01b reserved 10b rco1m 11b slow_clk
pwm4_clk_sel	[25:24]	00b	R/W	The clock selection of PWM4 00b hclk 01b per_clk 10b rco1m 11b slow_clk
pwm3_clk_sel	[23:22]	00b	R/W	The clock selection of PWM3 00b hclk 01b per_clk 10b rco1m 11b slow_clk
pwm2_clk_sel	[21:20]	00b	R/W	The clock selection of PWM2 00b hclk 01b per_clk 10b rco1m 11b slow_clk
pwm1_clk_sel	[19:18]	00b	R/W	The clock selection of PWM1 00b hclk 01b per_clk 10b rco1m 11b slow_clk
pwm0_clk_sel	[17:16]	00b	R/W	The clock selection of PWM0 00b hclk 01b per_clk 10b rco1m 11b slow_clk
reserved	[15:14]			
ext_slow_clk_sel	[13:8]	0x00	R/W	[13]: External slow clock input enable. 0b Disable

				1b Enable [12:8]: Select which GPIOxx is used as the external slow clock input. The value of 0x00 selects GPIO00 , the value 0x01 selects GPIO01 and so on.
reserved	[7:6]			
uart2_clk_sel	[5:4]	00b	R/W	The clock selection of UART2 00b per_clk 01b reserved 10b rco1m 11b rco32k
uart1_clk_sel	[3:2]	00b	R/W	The clock selection of UART1 00b per_clk 01b reserved 10b rco1m 11b rco32k
uart0_clk_sel	[1:0]	00b	R/W	The clock selection of UART0 00b per_clk 01b reserved 10b rco1m 11b rco32k

4. Resets

The reset sources and their targets of CZ20 is summarized in Table 4-1 and graphically illustrated in Figure 4-1.

Table 4-1 Reset Sources and Their Targets

Reset Source	Type	Signal	Targets		
			Deep Power-down Control	RTC GPIO	All others
VBAT	Power	vbat_por_n	O	O	O
RST_N	External pin	ext_rst_n	O	O	O
VDD_DIG	Power	vdig_por_n		O	O
Deep power-down wakeup reset	Internal event	dpd_rst_n		O	O
Deepsleep wakeup reset	Internal event	ds_rst_n			O
Watchdog reset	Internal event	wdt_rst_n			O
Software reset	Internal event	soft_rst_n			O
MCU lockup	Internal event	mcu_lockup_n			O

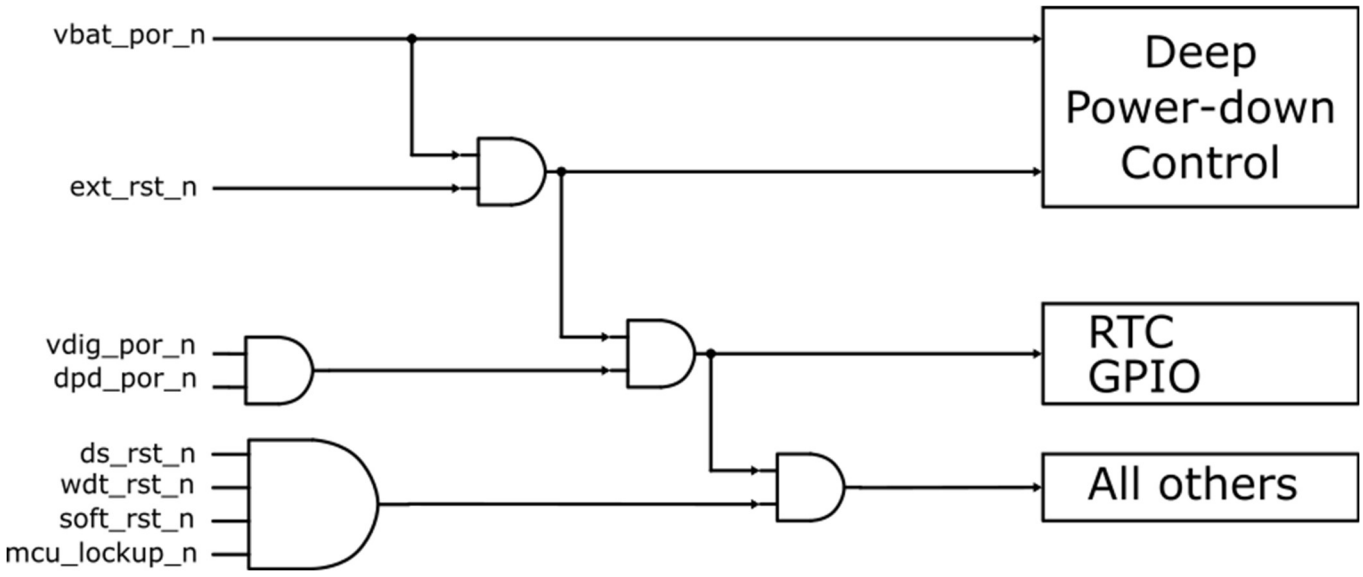


Figure 4-1 CZ20 Reset Scheme

4.1. Power-on Reset

The power-on reset, vbat_por_n, is triggered when the power source, VBAT, is above the pre-defined threshold. It has the highest priority over other reset signals and will reset the whole system as illustrated in Figure 4-1.

4.2. External Reset

There is a low-actively external reset pin, RST_N, that can reset the whole chip externally. To active the external reset, RST_N shall be asserted for at least 100us. A 100K ohm pull-up resistor is attached to the RST_N pin internally so RST_N can be left floating if not used.

4.3. Deep Power-down Wakeup Reset

During Deep Power-down mode, the whole chip is powered off except the Deep Power-down Control. After waking up from the Deep Power-down mode, the reset signal, dpd_rst_n, will be triggerd to reset the whole chip.

4.4. Deepsleep Wakeup Reset

During Deepsleep mode, most of the digital parts are powered off to reduce the leakage current. To let the digital circuits work correctly after waking up from Deepsleep, a Deepsleep wakeup reset signal, ds_rst_n, is generated automatically by the hardware to reset the corresponding digital circuits.

4.5. Watchdog Reset

When Watchdog Timer is not kicked for a certain period, the watchdog reset is triggered. Please refer to Chapter 15 Watchdog Timer for operations of Watchdog Timer in detail.

4.6. Software Reset

The software reset is triggered when the SYSRESETREQ bit in AIRCR register of Cortex-M33 is set. Please refer to the documents of Cortex-M33 for how to set SYSRESETREQ.

4.7. MCU Lockup

The Cortex-M33 enters a lockup state if a fault occurs when it cannot be serviced or escalated. In this situation, the processor does not execute any instructions and asserts the LOCKUP signal. It will then trigger the hardware controller to reset the entire the system.

4.8. Registers of Reset Causes

Because either reset will result in the reboot of the software, there are reset cause registers that can be used to identify what causes the reboot. The reset cause registers are resident in the Deep Power-down Control space whose base address is 0x50005000 for the secure partition or 0x40005000 for the non-secure partition.

The reset cause registers are reset by the power-on reset only and software shall clear them manually by writing 1b to the register bit, clr_rst_cause. Those registers are described below.

- Offset 0x00: DPD_RST_CAUSE

Signal	Bits	Default	R/W	Description
Reserved	[31:7]			
rst_cause_lock	[6]	0b	R	High indicates the reset cause is the mcu lockup. Cleared by clr_rst_cause.
rst_cause_soft	[5]	0b	R	High indicates the reset cause is the software reset. Cleared by clr_rst_cause.
rst_cause_wdt	[4]	0b	R	High indicates the reset cause is the watchdog reset. Cleared by clr_rst_cause.
rst_cause_ds	[3]	0b	R	High indicates the reset cause is the Deepsleep wakeup. Cleared by clr_rst_cause.
rst_cause_dpd	[2]	0b	R	High indicates the reset cause is the Deep Power-down wakeup. Cleared by clr_rst_cause.
rst_cause_ext	[1]	0b	R	High indicates the reset cause is the external reset. Cleared by clr_rst_cause.
rst_cause_por	[0]	1b	R	High indicates the reset cause is the power-on reset. Cleared by clr_rst_cause.

- Offset 0x04: DPD_CMD

Signal	Bits	Default	R/W	Description
Reserved	[31:1]			
clr_rst_cause	[0]	0b	W1C(1)	Write 1 to clear all reset cause to 0.

(1)W1C means “write 1 and auto clear”.

5. Low Power Modes

Almost IoT devices are battery powered and some of them requires very long battery life. This makes the current consumption very critical. CZ20 has several low-power modes that will provide different degrees of power saving. Those low-power modes are WFI/WFE, Sleep, Deep Sleep and Deep Power-down.

Low-power modes are triggered by WFI or WFE instruction of the CPU. In cooperated with the cfg_set_lowpower register in System Control, software can decide whether to enter those low-power modes or not. Software shall set cfg_set_lowpower correctly before executing WFI or WFE instruction. The cfg_set_lowpoer setting for each low-power mode is listed in below table.

Table 5-1 Register Settings of Low-power Modes

cfg_set_lowpoer settings	Power Modes
000b	WFI / WFE only
001b	Sleep
01xb	Deep Sleep
1xxb	Deep Power-down

Major characteristics of those low power states are summarized in Table 5-2. Because states of CPU are lost in Deep Sleep and Deep Power-down modes, CPU will be reboot when waking up from these two modes.

Table 5-2 Summary of Low-power Modes

Power Modes	Active LDOs	Active Clocks	CPU State	Memory	Wakeup
Normal	MV DCDC/(LDO)* LV LDO Retention LDO	RCO32K/(XO32K) XO32M (Baseband PLL) (RCO1M)	Active WFI/WFE	On	Interrupts
Sleep	Retention LDO	RCO32K/(XO32K) (RCO1M)	WFI/WFE	Retention	Interrupts
Deep Sleep	Retention LDO	Off/(RCO32K/XO32K)	Off	Off/(Retention)	GPIO (RTC timer)
Deep Power-down	None	None	Off.	Off	GPIO

*Items inside ( ) are optional.

5.1. Power Domains of Peripherals

In order to further reduce the leakage current in low-power modes, peripherals are partitioned into 4 power domains according to their functionalities in Sleep and Deepsleep modes. Table 5-3 shows the default behaviors of each peripheral power domain in each low-power mode.

Table 5-3 Power Domains of Peripherals

Power Domain	Peripherals	Sleep	Deepsleep	Deep Power-down
Always on	RTC Timer GPIO Control Watchdog Timer AUX Comparator BOD Comparator RCO32K Calibration	On	On	Off
Peripherals 1	Cortex-M33 Flash Control UART 0 I2C Slave Slow-clock Timer 0/1 Security Control	On	Off	Off
Peripherals 2	PWM DMA 0/1 UART 2 QSPI 0/1 Crypto Engine I2C Master 0 I2C Master 1 AUX ADC IRM PUFrt	Off	Off	Off
Peripherals 3	Cache Control GPIO Output MUXs UART 1 Timer 0/1/2 RCO1M Calibration FPU of Cortex-M33 DBG of Cortex-M33	Off	Off	Off

Be aware of peripherals in Peripherals 2 Power domain. Because those peripherals are powered off in Sleep mode by default, their internal states are reset to the default values after waking up from Sleep mode. It is suggested to run initialization procedures of the peripheral every time when it is used.

5.2. Power Modes of COMM Subsys

Like the Host CPU, the COMM Subsys has its own power states and they are WFI/WFE only, Sleep and Deepsleep modes. The WEI/WFE only and Sleep mode are controlled by the COMM Subsys firmware regardless of the power states of Host CPU. Its Deepsleep mode is controlled by the Host CPU to power off entirely the COMM Subsys. Host CPU shall also power off the COMM Subsys if it wants to enter Deepsleep mode to minimize the current consumption.

Be aware to reload the firmware of COMM Subsys when wakes COMM Subsys from the Deepsleep since all status of COMM Subsys are lost in its Deepsleep mode. The Host CPU can do those operations through the COMM Subsys interface described in Chapter 7 COMM Subsys.

Because low-power modes of COMM Subsys also attempts to power down LDOs and Clocks of the system, the actual on/off states of LDOs and Clocks are determined by both Host CPU and COMM subsys. LDOs and Clocks of the system will be kept on if either Host CPU or COMM Subsys is in Normal mode.

5.3. Wakeup in Sleep Mode

Though all interrupts can wake up the system in Sleep Mode, however, only certain peripherals have the capability to generate interrupts in Sleep Mode. GPIOs, slow-clock Timers, RTC Timer, I2C Slave, AUX Comparators, BOD Comparator, UARTs, and COMM Subsys are such peripherals. Wakeup sources in Sleep Mode are summarized in Table 5-4.

Table 5-4 Wakeup Sources in Sleep Mode

Wakeup Sources	Conditions	Comments
GPIOs	No	N/A
Slow-clock Timers	No	N/A
RTC Timer	No	N/A
I2C Slave	No	N/A
AUX Comparator	No	AUX Comparator shall be enabled in Sleep Mode.
BOD Comparator	No	BOD Comparator shall be enabled in Sleep Mode.
UARTx	uartx_clk_sel = 00b	1. uart_sleep_wake_en shall be set to 1b. 2. Only RX Break up to 1.5ms can wake up.
	uartx_clk_sel = 10b	1. uart_sleep_wake_en shall be set to 1b. 2. The rco1m shall be enabled and calibrated to 921.6 KHz. 3. UART can receive data and wake up in Sleep Mode with the baudrate up to 115200.

	uartx_clk_sel = 11b	1. uart_sleep_wake_en shall be set to 1b. 2. The rco32k shall be calibrated to 38.4 KHz. 3. UART can receive data and wake up in Sleep Mode with the baudrate up to 9600.
COMM Subsys	No	Controlled by the firmware of COMM Subsys

5.4. Wakeup in Deep Sleep Mode

Wakeup sources in Deep Sleep Mode can be GPIOs, RTC Timer, AUX comparator and BOD comparator. Their settings are summarized below.

Table 5-5 Wakeup Sources in Deep Sleep Mode

Wakeup Sources	Comments
GPIOx	1. The corresponding set_ds_wakeup_en[x] shall be set. 2. Either set_ds_wakeup_high[x] or set_ds_wakeup_low[x] shall be set to determine the wakeup polarity of GPIOx.
RTC Timer	Either rco32k or xo32k shall be enabled in Deep Sleep Mode.
AUX Comparator	AUX Comparator shall be enabled in Deep Sleep Mode.
BOD Comparator	BOD Comparator shall be enabled in Deep Sleep Mode.

5.5. Wakeup in Deep Power-down Mode

The Deep Power-down mode is the lowest power mode of CZ20. All internal clock sources and LDOs are powered off in this mode. All available GPIOs can be configured as the wakeup source by setting the registers, dpd_gpio_wake_en and dpd_gpio_wake_pol, in the memory space of Deep Power-down Control. Besides, there are 16-byte retention registers, also located in Deep Power-down Control, that can be used by Software. Data of those retention registers won't be lost until updating by Software or power-on reset.

- Offset 0x08: DPD_GPIO_EN

Signal	Bits	Default	R/W	Description
dpd_gpio_wake_en	[31:0]	0x0	RW	Set the corresponding GPIO as the wakeup source in Deep Power-down Mode. For example, bit [0] is for GPIO00, bit [1]

				is for GPIO01 and so on. 0b disable 1b enable

- Offset 0x0C: DPD_GPIO_POL

Signal	Bits	Default	R/W	Description
dpd_gpio_wake_pol	[31:0]	0x0	RW	Set the wakeup polarity of the corresponding GPIO in Deep Power-down Mode. For example, bit [0] is for GPIO00, bit [1] is for GPIO01 and so on. 0b level low 1b level high

- Offset 0x10: DPD_RET0

Signal	Bits	Default	R/W	Description
dpd_retention_0	[31:0]	0x0	RW	Retention register 0.

- Offset 0x14: DPD_RET1

Signal	Bits	Default	R/W	Description
dpd_retention_1	[31:0]	0x0	RW	Retention register 1.

- Offset 0x18: DPD_RET2

Signal	Bits	Default	R/W	Description
dpd_retention_2	[31:0]	0x0	RW	Retention register 2.

- Offset 0x1C: DPD_RET3

Signal	Bits	Default	R/W	Description
dpd_retention_3	[31:0]	0x0	RW	Retention register 3.

5.6. Memory Low-power Control

System memories are consisted of seven memory macros, RAM 0~6. Each memory macro has its own low-power modes, Deep sleep, Shutdown and Power down. When a memory macro is in Deep sleep mode, data is retained but memory access is not allowed. When a memory macro is in Shutdown or Power down mode, data will be lost. Power modes of memory macros are summarized in Table 5-6

Table 5-6 Power Modes of Memory Macros

Power Mode	Controls	Memory Access	Data Retention
Active	sram_pd = 0 sram_sd = 0	Allowed	Yes

	sram_ds = 0		
Deepsleep	sram_pd = 0 sram_sd = 0 sram_ds = 1	Not allowed	Yes
Shutdown	sram_pd = 0 sram_sd = 1 sram_sd = x	Not allowed	No
Powerdown	sram_pd = 1 sram_sd = x sram_sd = x	Not allowed	No

To further reduce the current in system low-power modes, those memories are divided into several power domains and can be completely powered down as illustrated in Figure 5-1. Comparison of leakage current in those memory low-power modes is Powerdown < Shutdown < Deepsleep.

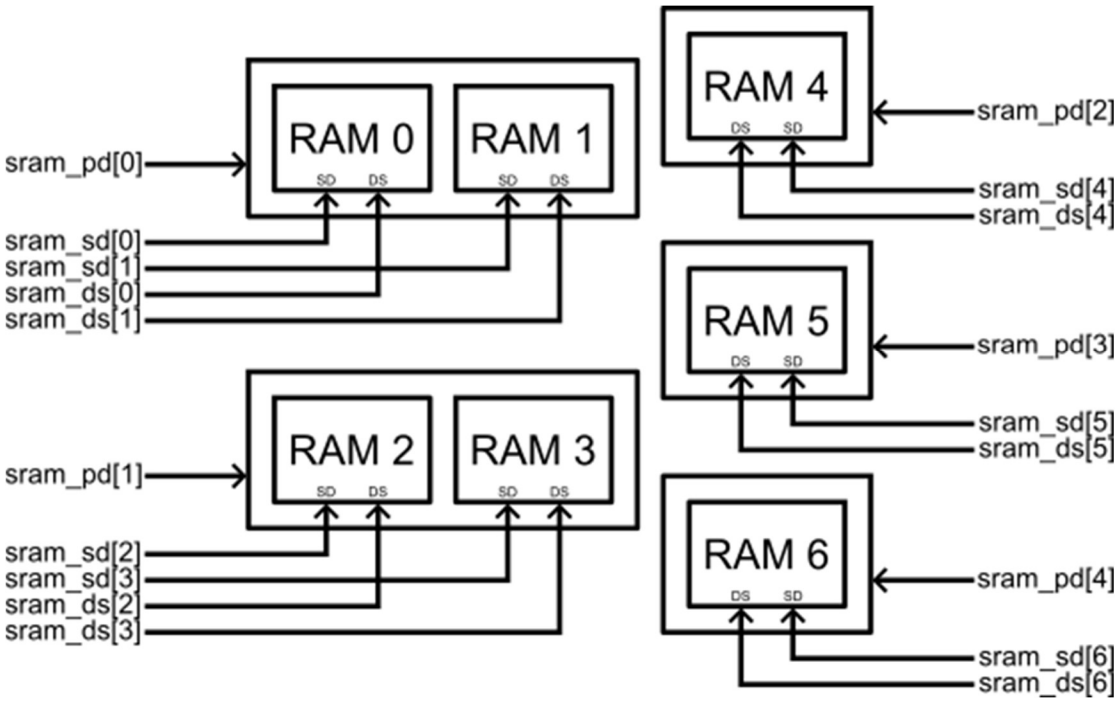


Figure 5-1 Power-domain Partition of Memories

Because RAM 0 and RAM 1 are in the same power domain, power off this power domain will power

down both RAM 0 and RAM 1 at the same time. If you want to retain data of either RAM 0 or RAM 1 is system low-power modes, you can't power off this power domain. Instead, you shall shut down the other. RAM 2 and RAM 3 are similar to RAM 0 and RAM 1 since they are also in one power domain.

RAM 4, RAM 5, and RAM 6 have their own power domain. It is suggested to use powerdown instead of shutdown if you won't to retain their data in system low-power modes.

Users can configure the power mode of individual memory macro in system low-power modes. Hardware will set memories in proper power modes according to the system power modes automatically.

Detailed registers for memory low-power control are resident in the System Control space whose base address is 0x50000000 for the secure partition or 0x40000000 for the non-secure partition. Those configuration registers have the same naming rule like cfg_sram_postfix1_postfix2 while the postfix1 indicates the memory power mode and the postfix2 indicates the system power modes. For example, cfg_sram_ds_sp is used to set the memory in Deepsleep mode when the system is in Sleep mode and cfg_sram_pd_nm is used to set the memory in Powerdown mode when the system is in Normal mode.

- Offset 0x50: SRAM_LOWPOWER_0

Signal	Bits	Default	R/W	Description
cfg_sram_ds_ds[15:0]	[31:16]	0x0	R/W	Set 1b to the individual bit to set SRAM in Deepsleep mode when System is in Deepsleep mode. cfg_sram_ds_ds [0] : RAM 0 cfg_sram_ds_ds [1] : RAM 1 cfg_sram_ds_ds [2] : RAM 2 cfg_sram_ds_ds [3] : RAM 3 cfg_sram_ds_ds [4] : RAM 4 cfg_sram_ds_ds [5] : RAM 5 cfg_sram_ds_ds [6] : RAM 6 Others : Reserved
cfg_sram_ds_sp[15:0]	[15:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Deepsleep mode when System is in Sleep mode. cfg_sram_ds_sp [0] : RAM 0 cfg_sram_ds_sp [1] : RAM 1 cfg_sram_ds_sp [2] : RAM 2 cfg_sram_ds_sp [3] : RAM 3 cfg_sram_ds_sp [4] : RAM 4 cfg_sram_ds_sp [5] : RAM 5 cfg_sram_ds_sp [6] : RAM 6 Others : Reserved

- Offset 0x54: SRAM_LOWPOWER_1

Signal	Bits	Default	R/W	Description
cfg_sram_sd_sp[15:0]	[31:16]	0x0	R/W	Set 1b to the individual bit to set SRAM in

				Shutdown mode when System is in Sleep mode. cfg_sram_sd_sp [0] : RAM 0 cfg_sram_sd_sp [1] : RAM 1 cfg_sram_sd_sp [2] : RAM 2 cfg_sram_sd_sp [3] : RAM 3 cfg_sram_sd_sp [4] : RAM 4 cfg_sram_sd_sp [5] : RAM 5 cfg_sram_sd_sp [6] : RAM 6 Others : Reserved
cfg_sram_sd_nm[15:0]	[15:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Shutdown mode when System is in Normal mode. cfg_sram_sd_nm [0] : RAM 0 cfg_sram_sd_nm [1] : RAM 1 cfg_sram_sd_nm [2] : RAM 2 cfg_sram_sd_nm [3] : RAM 3 cfg_sram_sd_nm [4] : RAM 4 cfg_sram_sd_nm [5] : RAM 5 cfg_sram_sd_nm [6] : RAM 6 Others : Reserved

- Offset 0x58: SRAM_LOWPOWER_2

Signal	Bits	Default	R/W	Description
cfg_sram_pd_sp[7:0]	[31:24]	0x0	R/W	Set 1b to the individual bit to set SRAM in Powerdown mode when System is in Sleep mode. cfg_sram_pd_sp [0] : RAM 0 , RAM 1 cfg_sram_pd_sp [1] : RAM 2 , RAM 3 cfg_sram_pd_sp [2] : RAM 4 cfg_sram_pd_sp [3] : RAM 5 cfg_sram_pd_sp [4] : RAM 6 Others : Reserved
cfg_sram_pd_nm[7:0]	[23:16]	0x0	R/W	Set 1b to the individual bit to set SRAM in Powerdown mode when System is in Normal mode. cfg_sram_pd_nm [0] : RAM 0 , RAM 1 cfg_sram_pd_nm [1] : RAM 2 , RAM 3 cfg_sram_pd_nm [2] : RAM 4 cfg_sram_pd_nm [3] : RAM 5 cfg_sram_pd_nm [4] : RAM 6 Others : Reserved
cfg_sram_sd_ds[15:0]	[15:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Shutdown mode when System is in Deepsleep mode. cfg_sram_sd_ds [0] : RAM 0 cfg_sram_sd_ds [1] : RAM 1 cfg_sram_sd_ds [2] : RAM 2 cfg_sram_sd_ds [3] : RAM 3 cfg_sram_sd_ds [4] : RAM 4 cfg_sram_sd_ds [5] : RAM 5 cfg_sram_sd_ds [6] : RAM 6

				Others : Reserved

- Offset 0x5C: SRAM_LOWPOWER_3

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
cfg_sram_pd_ds[7:0]	[7:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Powerdown mode when System is in Deepsleep mode. cfg_sram_pd_ds [0] : RAM 0 , RAM 1 cfg_sram_pd_ds [1] : RAM 2 , RAM 3 cfg_sram_pd_ds [2] : RAM 4 cfg_sram_pd_ds [3] : RAM 5 cfg_sram_pd_ds [4] : RAM 6 Others : Reserved

5.7. Registers

Detailed registers for memory and peripheral low-power control are resident in the System Control space whose base address is 0x50000000 for the secure partition or 0x40000000 for the non-secure partition. Those configuration registers have the same naming rule like cfg_sram_postfix1_postfix2 while the postfix1 indicates the memory power mode and the postfix2 indicates the system power modes. For example, cfg_sram_ds_sp is used to set the memory in Deepsleep mode when the system is in Sleep mode and cfg_sram_pd_nm is used to set the memory in Powerdown mode when the system is in Normal mode.

- Offset 0x50: SRAM_LOWPOWER_0

Signal	Bits	Default	R/W	Description
cfg_sram_ds_ds[15:0]	[31:16]	0x0	R/W	Set 1b to the individual bit to set SRAM in Deepsleep mode when System is in Deepsleep mode. cfg_sram_ds_ds [0] : RAM 0 cfg_sram_ds_ds [1] : RAM 1 cfg_sram_ds_ds [2] : RAM 2 cfg_sram_ds_ds [3] : RAM 3 cfg_sram_ds_ds [4] : RAM 4 cfg_sram_ds_ds [5] : RAM 5

				cfg_sram_ds_ds [6] : RAM 6 Others : Reserved
cfg_sram_ds_sp[15:0]	[15:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Deepsleep mode when System is in Sleep mode. cfg_sram_ds_sp [0] : RAM 0 cfg_sram_ds_sp [1] : RAM 1 cfg_sram_ds_sp [2] : RAM 2 cfg_sram_ds_sp [3] : RAM 3 cfg_sram_ds_sp [4] : RAM 4 cfg_sram_ds_sp [5] : RAM 5 cfg_sram_ds_sp [6] : RAM 6 Others : Reserved

- Offset 0x54: SRAM_LOWPOWER_1

Signal	Bits	Default	R/W	Description
cfg_sram_sd_sp[15:0]	[31:16]	0x0	R/W	Set 1b to the individual bit to set SRAM in Shutdown mode when System is in Sleep mode. cfg_sram_sd_sp [0] : RAM 0 cfg_sram_sd_sp [1] : RAM 1 cfg_sram_sd_sp [2] : RAM 2 cfg_sram_sd_sp [3] : RAM 3 cfg_sram_sd_sp [4] : RAM 4 cfg_sram_sd_sp [5] : RAM 5 cfg_sram_sd_sp [6] : RAM 6 Others : Reserved
cfg_sram_sd_nm[15:0]	[15:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Shutdown mode when System is in Normal mode. cfg_sram_sd_nm [0] : RAM 0 cfg_sram_sd_nm [1] : RAM 1 cfg_sram_sd_nm [2] : RAM 2

				cfg_sram_sd_nm [3] : RAM 3 cfg_sram_sd_nm [4] : RAM 4 cfg_sram_sd_nm [5] : RAM 5 cfg_sram_sd_nm [6] : RAM 6 Others : Reserved

- Offset 0x58: SRAM_LOWPOWER_2

Signal	Bits	Default	R/W	Description
cfg_sram_pd_sp[7:0]	[31:24]	0x0	R/W	Set 1b to the individual bit to set SRAM in Powerdown mode when System is in Sleep mode. cfg_sram_pd_sp [0] : RAM 0 , RAM 1 cfg_sram_pd_sp [1] : RAM 2 , RAM 3 cfg_sram_pd_sp [2] : RAM 4 cfg_sram_pd_sp [3] : RAM 5 cfg_sram_pd_sp [4] : RAM 6 Others : Reserved
cfg_sram_pd_nm[7:0]	[23:16]	0x0	R/W	Set 1b to the individual bit to set SRAM in Powerdown mode when System is in Normal mode. cfg_sram_pd_nm [0] : RAM 0 , RAM 1 cfg_sram_pd_nm [1] : RAM 2 , RAM 3 cfg_sram_pd_nm [2] : RAM 4 cfg_sram_pd_nm [3] : RAM 5 cfg_sram_pd_nm [4] : RAM 6 Others : Reserved
cfg_sram_sd_ds[15:0]	[15:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Shutdown mode when System is in Deepsleep mode. cfg_sram_sd_ds [0] : RAM 0 cfg_sram_sd_ds [1] : RAM 1

	cfg_sram_sd_ds [2] : RAM 2 cfg_sram_sd_ds [3] : RAM 3 cfg_sram_sd_ds [4] : RAM 4 cfg_sram_sd_ds [5] : RAM 5 cfg_sram_sd_ds [6] : RAM 6 Others : Reserved

- Offset 0x5C: SRAM_LOWPOWER_3

Signal	Bits	Default	R/W	Description
reserved	[31:24]			
cfg_peri3_off_ds	[23]	0x1	R/W	Power off peripheral 3 power domain in Deepsleep mode. 0 Peripheral 3 power domain is on in Deepsleep mode. 1 Peripheral 3 power domain is off in Deepsleep mode.
cfg_peri3_off_sp	[22]	0x1	R/W	Power off peripheral 3 power domain in Sleep mode. 2 Peripheral 3 power domain is on in Sleep mode. 3 Peripheral 3 power domain is off in Sleep mode.
cfg_peri2_off_ds	[21]	0x1	R/W	Power off peripheral 2 power domain in Deepsleep mode. 4 Peripheral 2 power domain is on in Deepsleep mode. 5 Peripheral 2 power domain is off in Deepsleep mode.
cfg_peri2_off_sp	[20]	0x1	R/W	Power off peripheral 2 power domain in Sleep mode. 6 Peripheral 2 power domain is on in Sleep mode. 7 Peripheral 2 power domain is off in Sleep mode.

cfg_peri1_off_ds	[20]	0x1	R/W	Power off peripheral 1 power domain in Deepsleep mode. 8 Peripheral 1 power domain is on in Deepsleep mode. 9 Peripheral 1 power domain is off in Deepsleep mode.
reserved	[19:8]			
cfg_sram_pd_ds[7:0]	[7:0]	0x0	R/W	Set 1b to the individual bit to set SRAM in Powerdown mode when System is in Deepsleep mode. cfg_sram_pd_ds [0] : RAM 0 , RAM 1 cfg_sram_pd_ds [1] : RAM 2 , RAM 3 cfg_sram_pd_ds [2] : RAM 4 cfg_sram_pd_ds [3] : RAM 5 cfg_sram_pd_ds [4] : RAM 6 Others : Reserved

6. Security Controller

With the rapid growth and spread of IoT applications, the security is becoming increasingly important. It is necessary to protect the valuable data and store it securely. It is also important to prevent from unauthorized accesses.

With the TrustZone technology of Cortex-M33 and the assistance of specific hardware, CZ20 can provide several levels of security. Due to the limit function of the SAU (Security Attribution Unit) of Cortex-M33, it should be disabled and use the Security Controller instead.

The security controller controls the security attributes of Flash, RAM and peripherals. Both Flash and RAM can be partitioned into secure, non-secure callable (NSC), and non-secure (NS) regions. Every peripheral can also be set as secure or non-secure. Because Boot ROM is the root of trust, it is always secure.

To enable the Security Controller and therefore enable the security function, set the **sec_idau_en** register at the offset of 0x2C.

6.1. Region ID

Some instructions in Armv8-M architecture can check the security attribute of certain memory region that requires a region ID being assigned. The region ID of each memory region is listed in Table 6-1.

Table 6-1 Region ID of Each Memory Partition

Partition	Security Attribute	Region ID
Flash	secure	0x10
Flash	non-secure callable	0x11
Flash	non-secure	0x12
ROM	secure	0x20
RAM	secure	0x30
RAM	non-secure callable	0x31
RAM	non-secure	0x32
Peripherals	secure	0x40
Peripherals	non-secure	0x42

6.2. Security Partition of Flash

The Flash can be partitioned into secure, non-secure callable, and non-secure regions via four register settings, sec_flash_s_stop, sec_flash_nsc_start, sec_flash_nsc_stop, and sec_flash_ns_stop of Security Controller. The unit of those register is 32 bytes that will match the minimum address granularity of the IDAU interface of Cortex-M33. The memory map of Flash after the security partition is summarized in Table 6-2.

Table 6-2 Memory Map of Flash After Security Partition

Flash Partition	Start Address	End Address
Secure	0x10000000	0x10000000 + sec_flash_s_stop*32 - 1
Non-secure Callable	0x10000000 + sec_flash_ncs_start*32	0x10000000 + sec_flash_nsc_stop*32 -1
Non-secure	0x00000000 + sec_flash_s_stop*32	0x00000000 + sec_flash_ns_stop*32 - 1

Be aware that both sec_flash_nsc_start and sec_flash_nsc_stop must not be within sec_flash_s_stop. In other words, the Non-secure callable region is within the Secure region as illustrated in Figure 6-1.

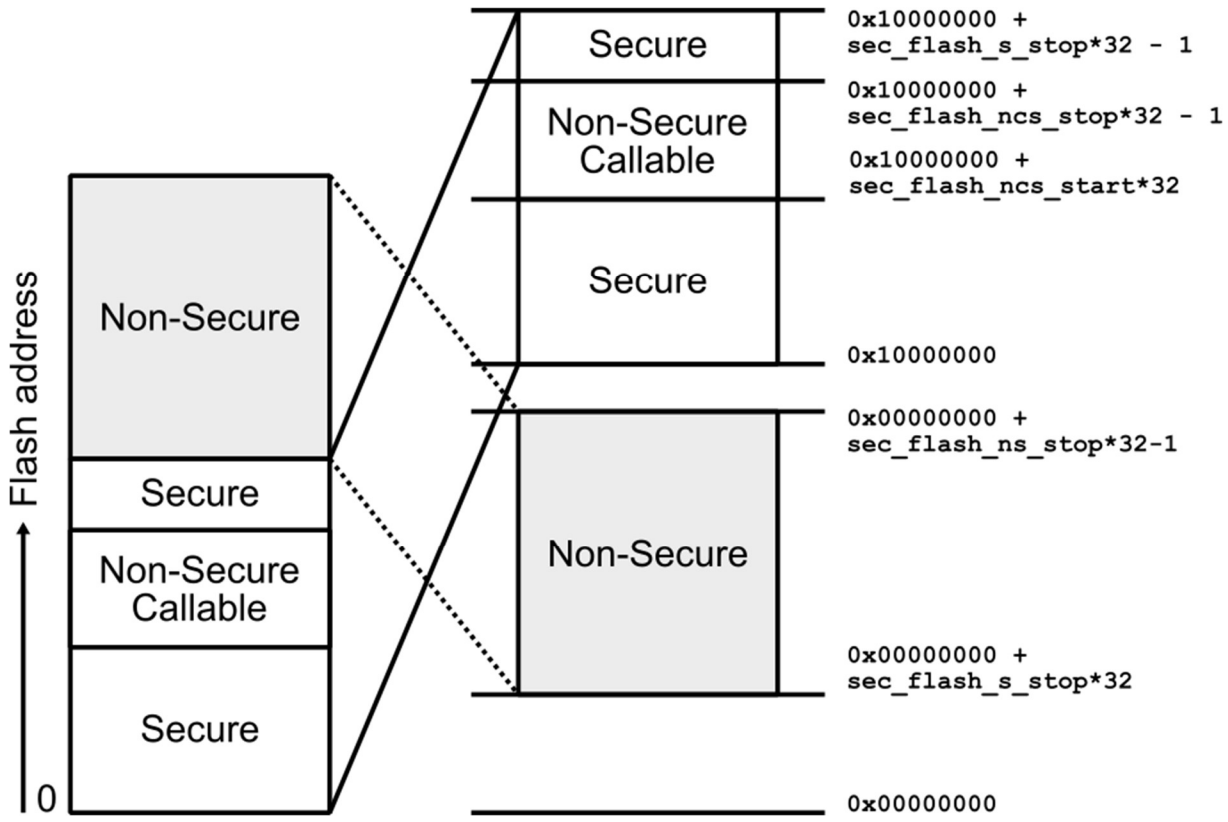


Figure 6-1 Memory Map of Flash After Security Partition

6.3. Security Partition of RAM

Similar to Flash, the RAM can be also partitioned into secure, non-secure callable, and non-secure regions via four register settings, sec_ram_s_stop, sec_ram_nsc_start, sec_ram_nsc_stop, and sec_ram_ns_stop of Security Controller. The unit of those registers is also 32 bytes. The memory map of RAM after the security partition is summarized in Table 6-3.

Table 6-3 Memory Map of RAM After Security Partition

RAM Partition	Start Address	End Address
Secure	0x30000000	0x30000000 + sec_ram_s_stop*32 - 1
Non-secure Callable	0x30000000 + sec_ram_nsc_start*32	0x30000000 + sec_ram_nsc_stop*32 - 1
Non-secure	0x20000000 + sec_ram_s_stop*32 - 1	0x20000000 + sec_ram_ns_stop*32 - 1

Similar to the partition of Flash, both sec_ram_nsc_start and sec_ram_nsc_stop must not be greater than sec_ram_s_stop. In other words, the Non-secure callable region is within the Secure region as illustrated in Figure 6-2.

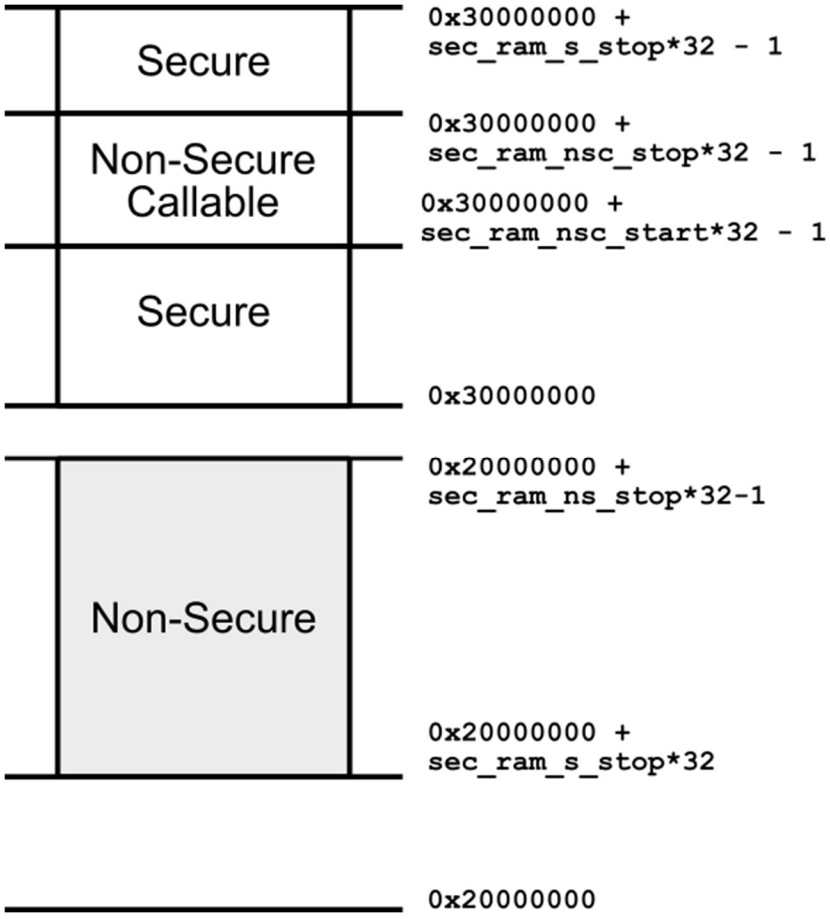


Figure 6-2 Memory Map of RAM After Security Partition

6.4. Security Partition of Peripherals

Each peripheral can be set as either secure or non-secure via registers, sec_peri_ns0, sec_peri_ns1, and sec_peri_ns2. Peripherals with secure attribute will be mapped to 0x50000000~0x5FFFFFFF, while peripherals with non-secure attribute will be mapped to 0x40000000~0x4FFFFFFF. When a bus master accesses a peripheral, the security attribute of the master, the destination address and the peripheral will be checked by the rules described in Table 6-4. Once violating the security rules, a write access will be ignored and a read access will read all zeros. A security error interrupt can be optionally enabled.

Table 6-4 Security Check Rules of Peripherals

Master Attribute	Destination Address Attribute	Peripheral Attribute	Access Result
Secure	Secure	secure	Success

Secure	Secure	non-secure	Failed
Secure	non-secure	secure	failed
Secure	non-secure	non-secure	success
non-secure	Secure	secure	failed
non-secure	Secure	non-secure	failed
non-secure	non-secure	secure	failed
non-secure	non-secure	non-secure	success

The DMAs and some peripherals with DMA capabilities can become bus masters and issue data transactions onto the backbone AHB bus. Security attributes of those bus masters are inherited from their peripheral attributes. For example, UART 0 has the DMA capability and when its security attribute is set to non-secure, all transactions issued from its DMA are all non-secure. Security attribute setting bits and DMA capabilities of peripherals are listed in Table 6-5.

Table 6-5 Security Attribute Setting Bits and DMA Capabilities of Peripherals

Peripheral	Security Attribute Setting Bit	DMA Capability
System Control	sec_peri_ns0[0]	No
GPIO	sec_peri_ns0[1]	No
Security Control	always secure	No
RTC Timer	sec_peri_ns0[4]	No
Deep Power-down Control	sec_peri_ns0[5]	No
Power Management	sec_peri_ns0[6]	No
Flash Control	sec_peri_ns0[9]	Yes
Timer 0	sec_peri_ns0[10]	No
Timer 1	sec_peri_ns0[11]	No
Timer 2	sec_peri_ns0[12]	No
Slow-clock Timer 0	sec_peri_ns0[13]	No
Slow-clock Timer 1	sec_peri_ns0[14]	No
Watchdog Timer	sec_peri_ns0[16]	No
UART 0	sec_peri_ns0[18]	Yes
UART 1	sec_peri_ns0[19]	Yes
I2C Slave	sec_peri_ns0[24]	No
COMM Subsys	sec_peri_ns0[26]	Yes
RCO32K Calibration	sec_peri_ns0[28]	No
BOD Comparator	sec_peri_ns0[29]	No

AUX Comparator	sec_peri_ns0[30]	No
RCO1M Calibration	sec_peri_ns0[31]	No
QSPI 0	sec_peri_ns1[0]	Yes
QSPI 1	sec_peri_ns1[1]	Yes
IRM	sec_peri_ns1[4]	No
UART 2	sec_peri_ns1[5]	Yes
PWM	sec_peri_ns1[6]	Yes
DMA 0	sec_peri_ns1[9]	Yes
DMA 1	sec_peri_ns1[10]	Yes
I2C Master 0	sec_peri_ns1[11]	No
I2C Master 1	sec_peri_ns1[12]	No
I2S	sec_peri_ns1[13]	Yes
AUX ADC	sec_peri_ns1[15]	Yes
Software IRQ 0	sec_peri_ns1[16]	No
Software IRQ 1	sec_peri_ns1[17]	No
PUFrt	sec_peri_ns2[1]	No
Crypto Engine	sec_peri_ns2[0]	Yes

6.5. Registers

- Offset 0x00: SEC_FLASH_S

Signal	Bits	Default	R/W	Description
Reserved	[31:19]			
sec_flash_s_stop	[18:0]	0x10000	R/W	Set the stop of secure region of Flash in the unit of 32 bytes. This will define the secure region of Flash from 0x10000000 to (0x10000000+sec_flash_s_stop*32-1)

- Offset 0x04: SEC_FLASH_NSC0

Signal	Bits	Default	R/W	Description
Reserved	[31:19]			
sec_flash_nsc_start	[18:0]	0x10000	R/W	Set the start of non-secure callable (NSC) region of Flash in the unit of 32 bytes. The NSC region of Flash is start from (0x10000000+sec_flash_nsc_start*32) to (0x10000000+sec_flash_nsc_stop*32 -1). Be aware that both sec_flash_nsc_start and sec_flash_nsc_stop MUST within the sec_flash_s_stop.

- Offset 0x08: SEC_FLASH_NSC1

Signal	Bits	Default	R/W	Description
reserved	[31:19]			
sec_flash_nsc_stop	[18:0]	0x10000	R/W	Set the stop of non-secure callable (NSC) region of Flash in the unit of 32 bytes. The NSC region of Flash is start from (0x10000000+sec_flash_nsc_start*32) to (0x10000000+sec_flash_nsc_stop*32 -1). Be aware that both sec_flash_nsc_start and sec_flash_nsc_stop MUST within the sec_flash_s_stop.

- Offset 0x0C: SEC_FLASH_NS

Signal	Bits	Default	R/W	Description
reserved	[31:19]			
sec_flash_nsc_stop	[18:0]	0x10000	R/W	Set the stop of non-secure callable (NSC) region of Flash in the unit of 32 bytes. The NSC region of Flash is start from (0x10000000+sec_flash_nsc_start*32) to (0x10000000+sec_flash_nsc_stop*32 -1). Be aware that both sec_flash_nsc_start and sec_flash_nsc_stop MUST within the sec_flash_s_stop.

- Offset 0x10: SEC_RAM_S

Signal	Bits	Default	R/W	Description
reserved	[31:15]			
sec_ram_s_stop	[14:0]	0x2C00	R/W	Set the stop of secure region of RAM in the unit of 32 bytes. This will define the secure region of RAM from 0x30000000 to (0x30000000+sec_ram_s_stop*32-1)

- Offset 0x14: SEC_RAM_NSC0

Signal	Bits	Default	R/W	Description
reserved	[31:15]			
sec_ram_nsc_start	[14:0]	0x2C00	R/W	Set the start of non-secure callable (NSC) region of RAM in the unit of 32 bytes. The NSC region of RAM is start from (0x30000000+sec_ram_nsc_start*32) to (0x30000000+sec_ram_nsc_stop*32 -1). Be aware that both sec_ram_nsc_start and sec_ram_nsc_stop MUST within the sec_ram_s_stop.

- Offset 0x18: SEC_RAM_NSC1

Signal	Bits	Default	R/W	Description
reserved	[31:15]			
sec_ram_nsc_stop	[14:0]	0x2C00	R/W	Set the stop of non-secure callable (NSC) region of RAM in the unit of 32 bytes. The NSC region of RAM is start from (0x30000000+sec_ram_nsc_start*32) to (0x30000000+sec_ram_nsc_stop*32 -1). Be aware that both sec_ram_nsc_start and sec_ram_nsc_stop MUST within the sec_ram_s_stop.

- Offset 0x1C: SEC_RAM_NS

Signal	Bits	Default	R/W	Description
reserved	[31:15]			
sec_ram_ns_stop	[14:0]	0x2C00	R/W	Set the stop of non-secure (NS) region of RAM in the unit of 32 bytes. The NS region of RAM is start from (0x00000000+sec_ram_s_stop*32) to (0x00000000+sec_ram_ns_stop*32 -1).

- Offset 0x20: SEC_PERI_ATTR0

Signal	Bits	Default	R/W	Description
sec_peri_ns0	[31:0]	0x00000000	R/W	Set the corresponding peripheral non-secure. 0b: secure 1b: non-secure Please refer to Table 6-5 for the bit map.

- Offset 0x24: SEC_PERI_ATTR1

Signal	Bits	Default	R/W	Description
sec_peri_ns1	[31:0]	0x00000000	R/W	Set the corresponding peripheral non-secure. 0b: secure 1b: non-secure Please refer to Table 6-5 for the bit map.

- Offset 0x28: SEC_PERI_ATTR2

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
sec_peri_ns2	[7:0]	0x00	R/W	Set the corresponding peripheral non-secure. 0b: secure 1b: non-secure Please refer to Table 6-5 for the bit map.

- Offset 0x2C: SEC_IDAU_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
sec_idau_en	[0]	0x00	R/W	Enable Implementation Defined Attribution Unit (IDAU) and related security control units. 0b disable 1b enable

- Offset 0x30: SEC_INT_EN

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
sec_en_puf_err_int	[5]	0b	R/W	Enable PUFrt security error interrupt. 0b disable 1b enable
sec_en_crypto_err_int	[4]	0b	R/W	Enable Crypto Engine security error interrupt. 0b disable 1b enable
sec_en_peri_err_int	[3]	0b	R/W	Enable peripherals security error interrupt.

				0b disable 1b enable
sec_en_ram_err_int	[2]	0b	R/W	Enable RAM security error interrupt. 0b disable 1b enable
sec_en_flash_err_int	[1]	0b	R/W	Enable Flash security error interrupt. 0b disable 1b enable
sec_en_rom_err_int	[0]	0b	R/W	Enable ROM security error interrupt. 0b disable 1b enable

- Offset 0x34: SEC_INT_CLR

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
sec_clr_puf_err_int	[5]	0b	W	Write 1b to clear PUFrt security error interrupt status.
sec_clr_crypto_err_int	[4]	0b	W	Write 1b to clear Crypto Engine security error interrupt status.
sec_clr_peri_err_int	[3]	0b	W	Write 1b to clear peripherals security error interrupt status.
sec_clr_ram_err_int	[2]	0b	W	Write 1b to clear RAM security error interrupt status.
sec_clr_flash_err_int	[1]	0b	W	Write 1b to clear Flash security error interrupt status.
sec_clr_rom_err_int	[0]	0b	W	Write 1b to clear ROM security error interrupt status.

- Offset 0x38: SEC_INT_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
sec_puf_err_status	[5]	0b	R	PUFrt security error interrupt status. 0b Interrupt is no pending. 1b Interrupt is pending.
sec_crypto_err_status	[4]	0b	R	Crypto Engine security error interrupt status. 0b Interrupt is no pending. 1b Interrupt is pending.
sec_peri_err_status	[3]	0b	R	Peripherals security error interrupt status. 0b Interrupt is no pending. 1b Interrupt is pending.
sec_ram_err_status	[2]	0b	R	RAM security error interrupt status. 0b Interrupt is no pending. 1b Interrupt is pending.
sec_flash_err_status	[1]	0b	R	Flash security error interrupt status. 0b Interrupt is no pending. 1b Interrupt is pending.
sec_rom_err_status	[0]	0b	R	ROM security error interrupt status. 0b Interrupt is no pending. 1b Interrupt is pending.

- Offset 0x40: SEC_MCU_DEBUG

Signal	Bits	Default	R/W	Description
reserved	[31:4]			

sec_spniden	[3]	1b	R/W	MCU secure non-invasive debug enable. 0b disable 1b enable
sec_niden	[2]	1b	R/W	MCU non-invasive debug enable. 0b disable 1b enable
sec_spiden	[1]	1b	R/W	MCU secure invasive debug enable. 0b disable 1b enable
sec_dbgen	[0]	1b	R/W	MCU invasive debug enable. 0b disable 1b enable

- Offset 0x44: SEC_LOCK_MCU_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:5]			
sec_locksau	[4]	0b	R/W	LOCKSAU of Cortex-M33 1b Disables writes to the SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers. 0b Unlocks these registers
sec_locknsmpu	[3]	0b	R/W	LOCKNSMPU of Cortex-M33 1b Disables writes to the MPU_CTRL_NS, MPU_RNR_NS, MPU_RBAR_NS, MPU_RLAR_NS, MPU_RBAR_A_NSn and MPU_RLAR_A_NSn registers. 0b Unlocks these registers
sec_locksmpu	[2]	0b	R/W	LOCKSMPU of Cortex-M33 1b Disables writes to the MPU_CTRL, MPU_RNR, MPU_RBAR, MPU_RLAR, MPU_RBAR_An and MPU_RLAR_An registers. 0b Unlocks these registers
sec_locknsvtor	[1]	0b	R/W	LOCKNSVTOR of Cortex-M33 1b Disables writes to the VTOR_NS register. 0b Unlocks this register
sec_locksvtaircr	[0]	0b	R/W	LOCKSVTAIRCR of Cortex-M33 1b Disables writes to the VTOR_S, AIRCR.PRIS and AIRCR.BFHFNMINS registers. 0b Unlocks these registers

- Offset 0x48: SEC_OTP_Write_Protect

Signal	Bits	Default	R/W	Description
sec_otp_write_key	[31:0]	0x00	R/W	Set the correct key, 0x28514260, to disable the write protect to PUFrt memory space. Since the whole PUFrt memory space is protected by this key, not only programming OTP but also enabling TRNG shall set this key first. Be sure to write 0x0 to this key again to enable the write protection after the write access.

7. COMM Subsys

The communication subsystem (COMM Subsys) is a tiny but powerful system which is optimized for handling wireless communications. It incorporates an ARM Cortex-M0+ MCU, a hardware MAC, a Modem, and a RF transceiver that can support multiple wireless communication protocols. With the total 112 KB memory, it is possible to implement multiple protocols in a single firmware and maintains several communication links at the same time.

7.1. Block Diagram

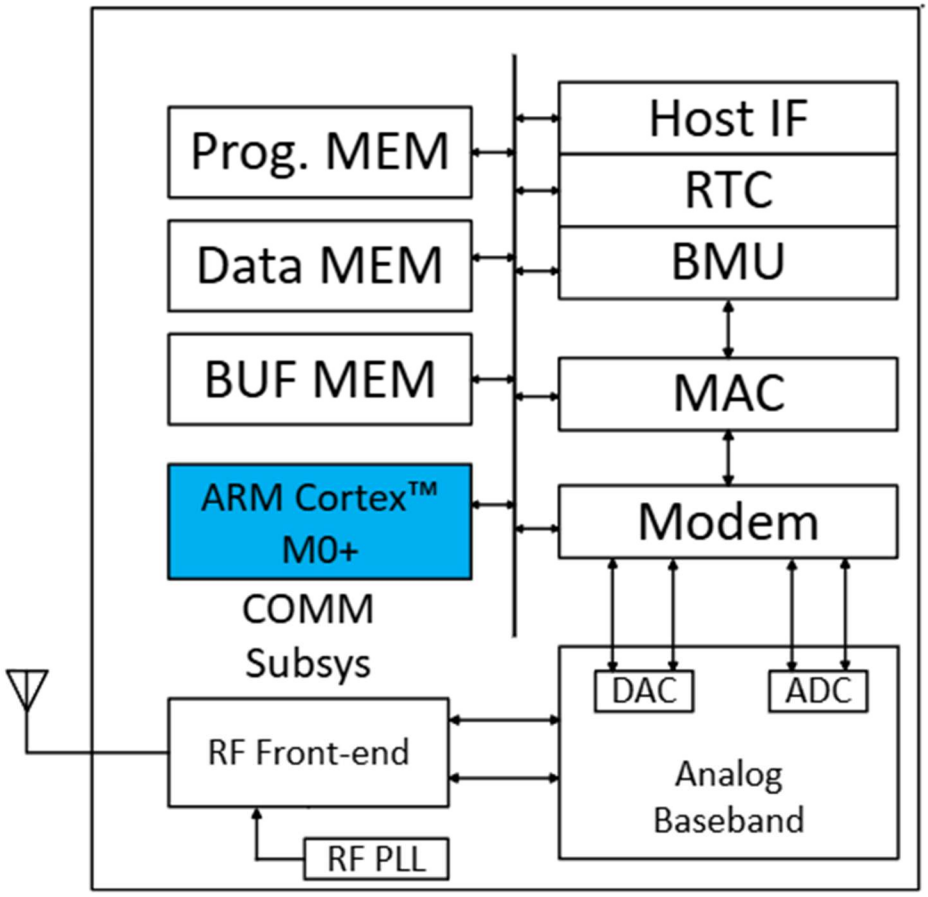


Figure 7-1 Block Diagram of COMM Subsys

7.2. Cortex M0+ MCU

The Cortex M0+ MCU serves as the radio controller and is the core component of the COMM Subsystem. It manages the time-sensitive tasks associated with wireless protocols, schedules transmission and reception timing windows, and handles Listen Before Talk (LBT). This offloads the SoC CPU, freeing up resources for user applications, and significantly reduces power consumption.

7.3. Buffer Memory

The COMM Subsystem features an 8K-byte buffer memory for storing transmission and reception packets. This memory is managed by the Buffer Management Unit (BMU) and is organized into multiple queues. It supports one RX queue and up to seven TX queues, with each queue capable of holding up to 4K bytes. This design offers significant flexibility, making it suitable for supporting a variety of communication protocols.

7.4. Host Interface

The Host CPU communicates and exchanges data with the COMM Subsystem through the Host Interface. This interface includes a DMA, a Host Command register, and various status registers, with interrupt functionality also available.

The DMA acts as a bus master, capable of issuing memory read or write transactions in the background. It can generate an interrupt request to the Host CPU once the transaction is complete. There are two types of DMA operations: memory transactions and I/O transactions.

7.4.1. Memory Transaction of DMA

The memory transaction is the typical DMA operation that move data from the memory space of Host CPU to that of COMM Subsys or vice versa. The memory map of COMM Subsys seen by Host CPU is listed in Table 7-1.

Table 7-1 Memory Map of COMM Subsys Seen by Host CPU

Modules	Size	Address	Condition
Program RAM 0	32KB	0x8000~0xFFFF	dma_pm_sel(1) = 00b
Program RAM 1	32KB	0x8000~0xFFFF	dma_pm_sel(1) = 01b

Program RAM 2	32KB	0x8000~0xFFFF	dma_pm_sel[1] = 10b
Data RAM	8KB	0x2000~0x3FFF	No
Buffer RAM	8KB	0x4000~0x5FFF	No
BMU	256B	0x0000~0x00FF	No
MAC	256B	0x0100~0x01FF	No
Modem	256B	0x0200~0x02FF	No
RF	256B	0x0300~0x03FF	No
PMU	256B	0x0400~0x04FF	No
[1]The address of dma_pm_sel is 0x0448 located at the PMU space of COMM Subsys.			

7.4.2. I/O Transaction of DMA

Unlike the memory transaction, the I/O transaction moves data between memories of Host CPU and buffer queues of COMM Subsys. Queues are initialized by the MCU and managed by BMU of COMM Subsys. Queues are also routed to the MAC directly so that they are used for exchanging communication packets. There are eight TX queues (TxQ #0~7) and two RX queues (RxQ #0~1). All queues are unidirectional and are categorized as TX queues and RX queues. The direction of TX queues is from Host CPU to COMM Subsys while that of RX queues are reversed.

The TxQ #7 is reserved for the command to MCU, and the RxQ #1 is reserved for the command response from MCU. Host CPU can use these two special queues to command the MCU and check the response.

7.4.3. Host Command

The Host Command is another way for the Host CPU to command the MCU. Unlike command queues of the I/O Transaction required the DMA operation, the Host Command is only a single-byte register that the Host CPU can access directly. Meanings of Host Commands are also defined by the firmware of COMM Subsys.

7.4.4. Status

There are also status bits that are useful for Host CPU to know the status of those queues and MCU.

Table 7-2 Status Bits of COMM SUbsys

Status	Description
rxq_rrdy[1:0]	High to indicates the data is available in the corresponding

	RxQ. [0] is for RxQ #0 and [1] is for RxQ #1.
mac_rx_info[3:0]	MAC RX information. [0] Packet length error [1] CRC error [2] MIC error [3] Buffer error
rx_pkt_len[11:0]	Indicate the data length in RxQ #0.
txq_wrdy[7:0]	High to indicates the corresponding TxQ is available. [0] is for TxQ #0, [1] is for TxQ #1, and so on.
mcu_state[7:0]	The MCU state defined by the firmware of COMM Subsys.
cmdr_len[11:0]	Indicate the data length of the command response in RxQ #1.
io_cmd_busy	High to indicates the current I/O transaction is in progress and another I/O transaction is not allowed.
host_cmd_busy	High to indicates the current Host Command is in progress and another Host Command is not allowed.

7.4.5. Interrupts

There are nine interrupt sources from the COMM Subsys and each of them has the individual enable, clear and status bits which are summarized in Table 7-3. Those interrupt sources are combined into a single interrupt request to Host CPU.

Table 7-3 Interrupt Sources of COMM Subsys

Interrupt Sources	Enable Bit	Clear Bit	Status Bit
MCU soft interrupt 0	comm_intr_en[0]	comm_intr_clr[0]	comm_intr_sta[0]
MCU soft interrupt 1	comm_intr_en[1]	comm_intr_clr[1]	comm_intr_sta[1]
MCU soft interrupt 2	comm_intr_en[2]	comm_intr_clr[2]	comm_intr_sta[2]
MCU soft interrupt 3	comm_intr_en[3]	comm_intr_clr[3]	comm_intr_sta[3]
MCU soft interrupt 4	comm_intr_en[4]	comm_intr_clr[4]	comm_intr_sta[4]
RX packet valid	comm_intr_en[5]	comm_intr_clr[5]	comm_intr_sta[5]
Reserved	comm_intr_en[6]	comm_intr_clr[6]	comm_intr_sta[6]
Reserved	comm_intr_en[7]	comm_intr_clr[7]	comm_intr_sta[7]
Host IF DMA done	comm_intr_en[8]	comm_intr_clr[8]	comm_intr_sta[8]

7.5. Host Mode and MCU Mode

The COMM Subsys has two operational modes, the Host Mode and the MCU Mode. In Host Mode, the

MCU is in the reset state and the Host CPU can use only the memory transactions. After the power-on reset, COMM Subsys is in the Host Mode to prevent the MCU from running with the undefined firmware in its program memories. The Host CPU shall load the firmware of COMM Subsys first via the memory transaction of the Host Interface. When the firmware is ready, the Host CPU can change the COMM Subsys to the MCU Mode by writing 1b to the dis_host_mode register bit and the MCU will start to run the firmware.

7.6. Registers

- Offset 0x00: COMM_DMA0

Signal	Bits	Default	R/W	Description
dma_ahb_addr	[31:0]	0x0	R/W	The start address of Host memory. It can be any valid address of the Host. Word or double-word aligned is not necessary.

- Offset 0x04: COMM_DMA1

Signal	Bits	Default	R/W	Description
dma_length	[31:16]	0x0	R/W	Data length to be transferred. It shall be greater than 0.
reserved	[15:14]			
dma_mcu_addr	[13:0]	0x0	R/W	The MSB part, [15:2], of the start address of MCU. Thus, the start address is always the double-word aligned.

- Offset 0x08: COMM_DMA2

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
dma_type	[1:0]	0x0	R/W	The DMA transaction type. 00b I/O read transaction. 01b I/O write transaction. 10b Memory read transaction. 11b Memory write transaction. The read or write is from the perspective of Host CPU.

- Offset 0x0C: COMM_COMMAND

Signal	Bits	Default	R/W	Description
reserved	[31:26]			
dis_host_mode	[25]	0x0	W1C	Write 1 to disable the Host Mode (enable the MCU mode).
en_host_mode	[24]	0x0	W1C	Write 1 to enable the Host Mode (disable the MCU mode).
reserved	[23:17]			
dma_en	[16]	0x0	W1C	Write 1 to enable the DMA. One DMA is enabled, it won't be stopped until the transaction is done. The dma_busy flag reflects the status of the DMA.
reserved	[15:12]			
comm_sys_reset	[11]	0x0	W1C	Write 1 to reset the whole COMM Subsys.

comm_deep_sleep_en	[10]	0x0	W1C	Write 1 to force the COMM Subsys into Deep Sleep Mode.
comm_sleep_en	[9]	0x0	W1C	Write 1 to force the COMM Subsys into Sleep Mode. Only valid in Host Mode.
comm_wakeup	[8]	0x0	W1C	Write 1 to wake up the COMM Subsys from low-power modes.
host_cmd	[7:0]	0x0	W1C	Host commands.

- Offset 0x10: COMM_INTR0

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
comm_intr_en	[8:0]	0x0	R/W	Enable interrupt sources of COMM Subsys. Please refer to Table 7-3 for the source of each bit.

- Offset 0x14: COMM_INTR1

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
comm_intr_clr	[8:0]	0x0	W1C	Write 1 to clear the corresponding interrupts. Please refer to Table 7-3 for the source of each bit.

- Offset 0x18: COMM_INTR2

Signal	Bits	Default	R/W	Description
reserved	[31:25]			
dma_busy	[24]	0x0	R	DMA busy flag. 0b DMA is not busy. 1b DMA is busy.
reserved	[23:9]			
comm_intr_sta	[8:0]	0x0	R	Interrupt status of COMM Subsys. Please refer to Table 7-3 for the source of each bit.

- Offset 0x1C: COMM_RX_INFO

Signal	Bits	Default	R/W	Description
reserved	[31:28]			
rx_pkt_len	[27:16]	0x0	R	Indicate the data length in RxQ #0.
reserved	[15:12]			
mac_rx_info	[11:8]	0x0	R	MAC RX information. [0] Packet length error [1] CRC error [2] MIC error [3] Buffer error
reserved	[7:2]			
rxq_rrdy	[1:0]	0x0	R	High to indicates the data is available in the corresponding RxQ. [0] is for RxQ #0 and [1] is for RxQ #1.

- Offset 0x20: COMM_TX_INFO

Signal	Bits	Default	R/W	Description
reserved	[31:28]			
cmdr_len	[27:16]	0x0	R	Indicate the data length of the command response in RxQ #1.
mcu_state	[15:8]	0x0	R	The MCU state defined by the firmware of COMM Subsys
txq_wrdy	[7:0]	0x0	R	High to indicates the corresponding TxQ is available. [0] is for TxQ #0, [1] is for TxQ #1, and so on.

- Offset 0x24: COMM_SYS_INFO

Signal	Bits	Default	R/W	Description
reserved	[31:18]			
host_cmd_busy	[17]	0x0	R	High to indicates the current Host Command is in progress and another Host Command is not allowed.
io_cmd_busy	[16]	0x0	R	High to indicates the current I/O transaction is in progress and another I/O transaction is not allowed.
reserved	[15:8]			
chip_rev	[7:4]	0x0	R	The hardware revision of COMM Subsys.
reserved	[3]			
pwr_state	[2:1]	0x0	R	The power state of COMM Subsys. 00b Other transition states 01b Deep Sleep 10b Sleep 11b Normal
sys_ready	[0]	0x0	R	High to indicate the COMM Subsys is ready.

8. Flash Control

The Flash Control controls the XiP flash through the dedicated SPI interface. It responds to the Cache Control for the requests of program codes. It also provides the software interface to erase and program the flash or to read the data in Flash manually.

The Flash Control includes a QSPI control, a TRX FIFO and a Control register as shown in Figure 8-1. The dedicated QSPI control will generate SPI commands to access the Flash. The TRX FIFO can store up to one page of data which is responsible for the page program or the page read.

8.1. Block Diagram

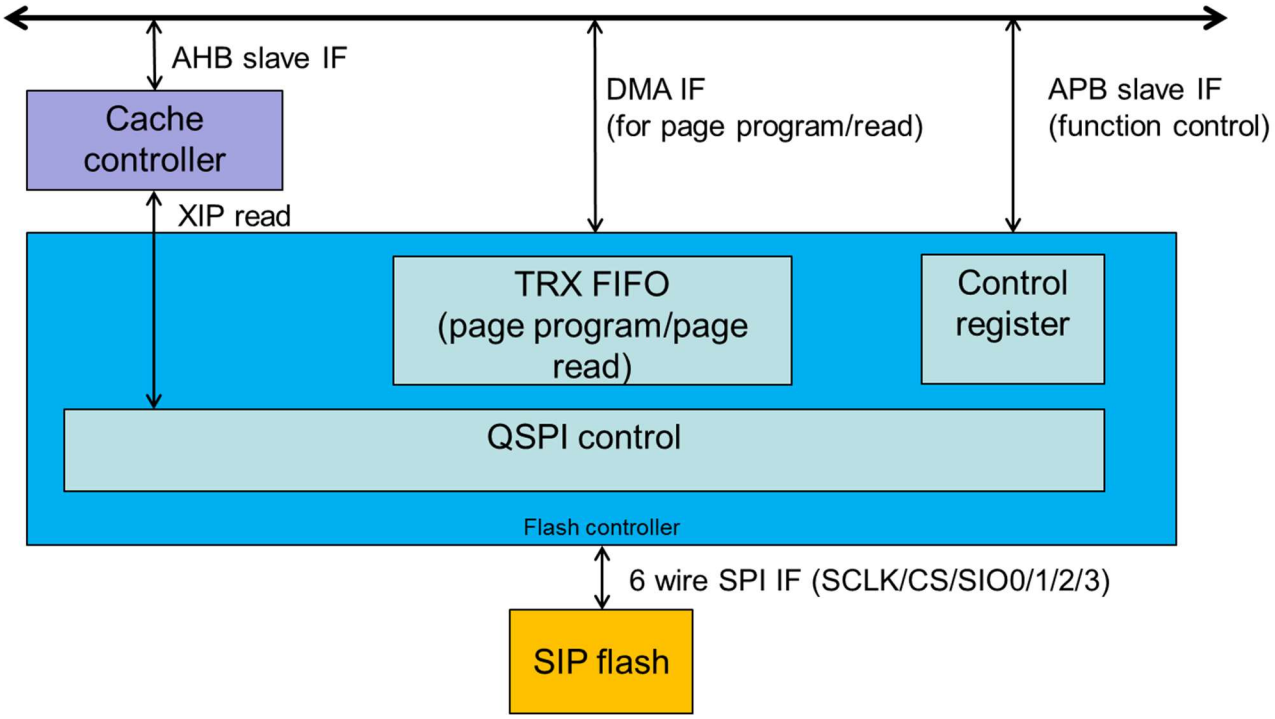


Figure 8-1 Block Diagram of Flash Control

8.2. Registers

- Offset 0x00: Controller Command

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
SecurityRegisterOperation	[6]		R/W	Security register control

			1'b0: memory array 1'b1: security register
Command	[5:0]	R/W	6'b00_XXXX: Read function XXXX definitions: 4'b0000: read byte (data @ reg: FlashReadData) 4'b0010: read SFDP (data @ reg: FlashReadData) 4'b0100: read verify (data @ reg: PageCRC) 4'b1000: read page (data @ mem: MemoryAddress) 4'b1001: read unique ID (data @ mem: MemoryAddress) 4'b1010: read SFDP (data @ mem: MemoryAddress) 4'b1101: read status register 1 (data @ reg: FlashReadData) 4'b1110: read status register 2 (data @ reg: FlashReadData) 4'b1111: read status register 3 (data @ reg: FlashReadData) 6'b01_XXXX: Write function XXXX: 4'b0000: program byte (data@ reg: FlashWriteData) 4'b1000: program page (data @ mem:MemoryAddress) 4'b1100: write status register 1 (&2, by 01h instruction setting by Sr12, data @ reg: StatusRegister1&2) 4'b1101: write status register 1 (data @ reg: StatusRegister1) 4'b1110: write status register 2 (data @ reg: StatusRegister2) 6'b10_XXXX: Erase function XXXX: 4'b0001: sector erase 4'b0010: block 32K erase 4'b0100: block 64K erase 4'b1000: chip erase 6'b11_XXXX:other function XXXX: 4'b0000: ID repolling 4'b1000: manual function

- Offset 0x04: Flash Address

Signal	Bits	Default	R/W	Description
FlashAddress	[31:0]		R/W	Flash address for operation

- Offset 0x08: Controller Start

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
ControllerBusy	[8]		R	Controller status
reserved	[7:1]			
ControllerStart	[0]		W	Control start, needs match with the set pattern

- Offset 0x0C: Flash Status Register

Signal	Bits	Default	R/W	Description
reserved	[31:16]		R/W	
StatusRegister2	[15:8]		R/W	Write flash status register 2
StatusRegister1	[7:0]		R/W	Write flash status register 1

- Offset 0x10: Flash Data

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
FlashReadData	[15:8]		R	Read data from flash: byte read data, byte read SFDP, read status register
FlashWriteData	[7:0]		R/W	Write data into flash: byte program data

- Offset 0x14: Memory Address Setting

Signal	Bits	Default	R/W	Description
MemoryAddress	[31:0]		R/W	Page program/read data from/to memory address (need 4 byte aligned, [1:0] need to be 2'b00)

- Offset 0x18: Controller Setting

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
SuspendEnable	[9]		R/W	When flash program/erase, cache can access flash 1'b0: disable 1'b1: enable
XIPReadEnable	[8]		R/W	Cache can access flash 1'b0: disable 1'b1: enable
SrQe	[7]		R/W	QE @ status register 1'b0: status register 1[6] 1'b1: status register 2[1]
Sr12	[6]		R/W	Status register 1 & 2 can use instruction 01h write 1'b0: no 1'b1: yes
ProgramMode	[5:4]		R/W	2'b00: 1-1-1 program (instruction = 02h) 2'b10: 1-1-4 program (instruction = 32h) 2'b11: 1-4-4 program (instruction = 38h)

ReadMode	[3:1]	R/W	3'b000: 1-1-1 read (instruction = 0Bh) 3'b001: 1-1-2 read (instruction = 3Bh) 3'b01x: 1-1-4 read (instruction = 6Bh) 3'b100: 1-1-1 read (instruction = 0Bh) 3'b101: 1-2-2 read (instruction = BBh, no support continuous mode) 3'b11x: 1-4-4 read (instruction = EBh, no support continuous mode)
EnterDpdEnable	[0]	R/W	Enter deep power-down when system is sleep

- Offset 0x1C: CRC

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
PageCRC	[7:0]		R	Page program/verify for page CRC result

- Offset 0x20: DPD Time

Signal	Bits	Default	R/W	Description
reserved	[31:14]			
DPDTime	[13:0]		R/W	Deep power-down time

- Offset 0x24: RDPD Time

Signal	Bits	Default	R/W	Description
reserved	[31:14]			
RDPDTime	[13:0]		R/W	Release deep power-down time

- Offset 0x28: Suspend Time

Signal	Bits	Default	R/W	Description
reserved	[31:14]			
SuspendTime	[13:0]		R/W	Suspend time

- Offset 0x2C: Resume Time

Signal	Bits	Default	R/W	Description
reserved	[31:14]			
ResumeTime	[13:0]		R/W	Resume time

- Offset 0x34: Page Read Byte Number

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
PageReadByte	[7:0]		R/W	Page read byte number, need 4 byte (double word, [1:0] must = 2'b11)

- Offset 0x38: Flash Information

Signal	Bits	Default	R/W	Description

Flash_standby	[31]	R	Flash can access
reserved	[30:24]		
FlashCapacityID	[23:16]	R	Flash capacity ID
FlashTypeID	[15:8]	R	Flash type ID
ManufacturerID	[7:0]	R	Flash manufacturer ID

- Offset 0x40: Flash Control Interrupt

Signal	Bits	Default	R/W	Description
reserved	[31:17]			
InterruptClear	[16]		W	Interrupt clear
reserved	[15:9]			
InterruptEnable	[8]		R/W	Interrupt enable
reserved	[7:1]			
InterruptStatus	[0]		R	Interrupt status

- Offset 0x44: Operation Pattern Match

Signal	Bits	Default	R/W	Description
PatternMatch	31:0		R/W	Need write RABT ASCII (0x52_41_42_54) for start operation

9. DMA

The DMA is intended to be used as a controller to transfer data directly from memory space to another memory space, including system memories and peripheral devices.

Once configured and enabled, the DMA controller is controlled data flow from an AHB Master write port bus to another AHB Master read port bus, which initiates data transfers across the two AHB bus through the DMA Buffer.

There are two identical DMA controllers, DMA 0 and DMA 1 in CZ20, that can operate independently.

9.1. Block Diagram

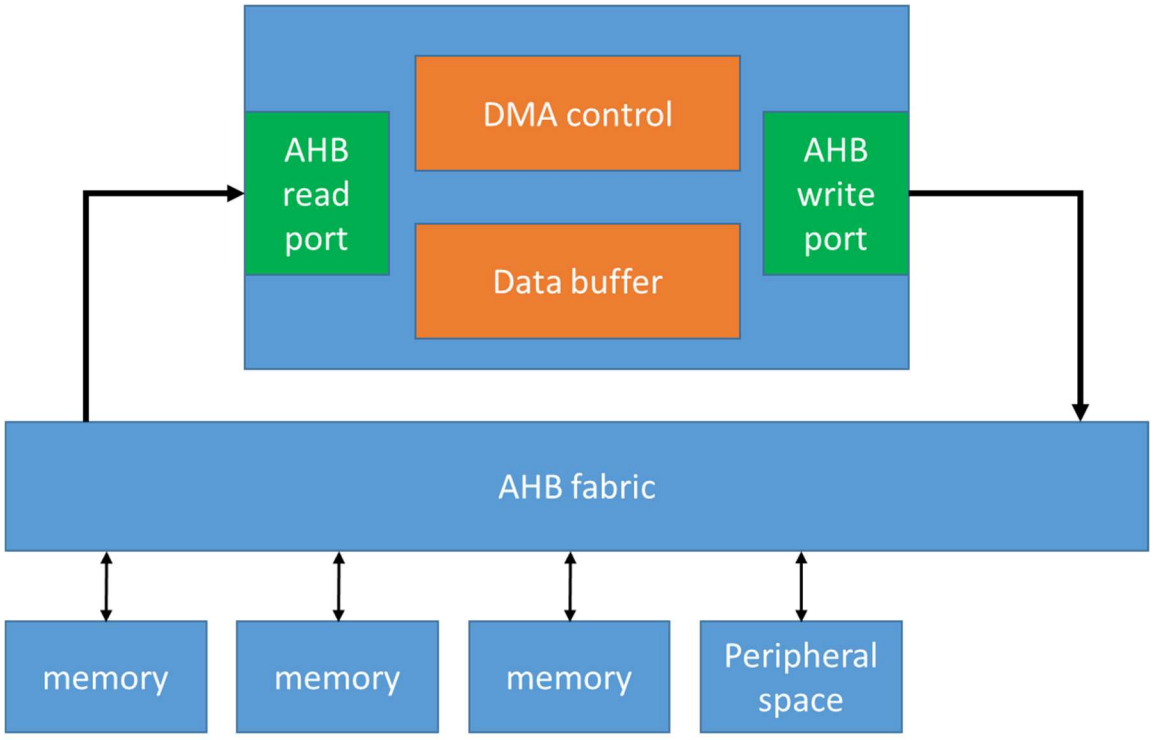


Figure 9-1 Block Diagram of DMA Controller

9.2. Features

- Memory space to Memory space, memory space including peripheral mapping space
- Transfer complete interrupt

- Transfer complete interrupt

9.3. Functional Description

The DMA controller has two AHB master interfaces. One is for read data and another is for write data. we need to set the read source address, write destination address and transfer data sizes. We use write 1 clear command to start the DMA controller to move data from one address map to another. Once finishing the data moving, the controller can produce the interrupt to inform the SW.

9.4. Registers

- Offset 0x00: Read source address

Signal	Bits	Default	R/W	Description
Read source address	[31:0]	0b	R/W	Setting read source address

- Offset 0x04: Write destination address

Signal	Bits	Default	R/W	Description
Write destination address	[31:0]	0b	R/W	Setting write destination address

- Offset 0x08: Moving data size

Signal	Bits	Default	R/W	Description
Moving data size	[31:0]	1b	R/W	Moving data size

- Offset 0x0C: Controller Start

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
ControllerBusy	[8]	0b	R	Controller busy for moving data
reserved	[7:1]			
ControllerStart	[0]	0b	W	Write 1b to start moving data

- Offset 0x10: DMA Controller Interrupt

Signal	Bits	Default	R/W	Description
reserved	[31:17]			
InterruptClear	[16]	0b	W	Write 1b to clear DMA controller Interrupt status
reserved	[15:10]			
WakeupEnable	[9]	0b	R/W	Wakeup enable
InterruptEnable	[8]	0b	R/W	Interrupt enable
reserved	[7:1]			
InterruptStatus	[0]	0b	R	Interrupt status

- Offset 0x14: Address bus protection

Signal	Bits	Default	R/W	Description

reserved	[31:15]			
AHB write port prot	[14:8]	1b	R/W	AHB write port prot setting
reserved	[7]			
AHB Read port prot	[6:0]	1b	R/W	AHB read port prot setting

10. xDMA

The xDMA is a simple data memory access module that is hooked for several peripherals. The xDMA will start to access the memory data from the start address pointer whenever a peripheral need to read or write memory. Depending on the direction of the data flow, the xDMA will be WDMA (from peripheral to memory) or RDMA (from memory to peripheral). Peripherals hooked to the xDMA are I2S, PWM and AUX ADC.

There are two operation modes supported by the xDMA. The single-shot mode, like the conventional DMA, moves data from or to the memory with a predefined length. The continuous mode treats a block of memory as a ring buffer which is very useful for streaming applications. A round-robin arbiter is embedded into the xDMA that will do arbitrations between requests of peripherals.

10.1. Block Diagram

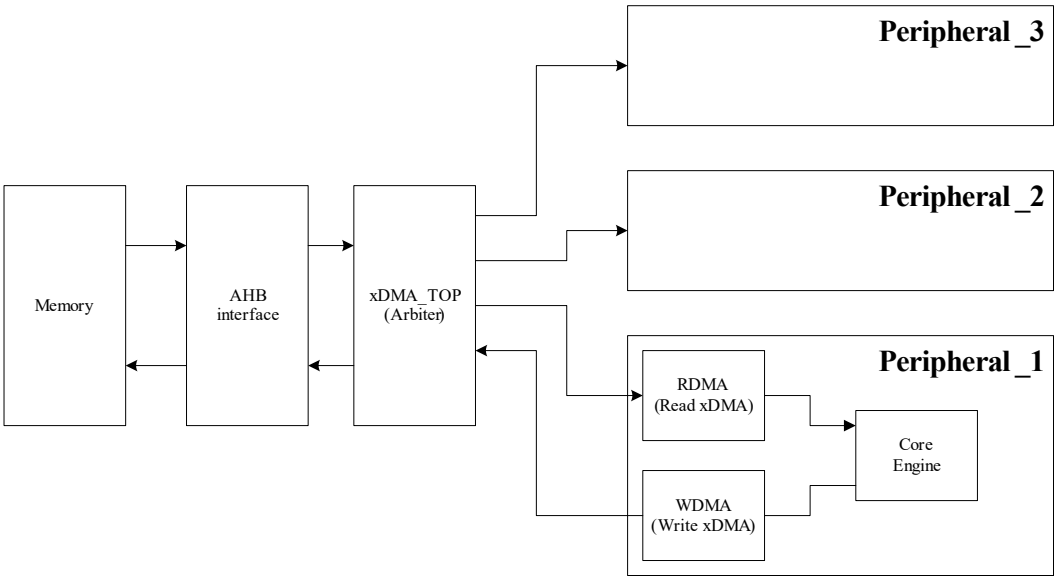


Figure 10-1 Block Diagram of xDMA

10.2. Functional Descriptions

To understand the operation of xDMA more easily, we first define some parameters which are

summarized below.

- PRT_START is the start pointer
- SEG_N is the segment size
- BLK_M is the block size
- INT is the interrupt
- INT_CLR is the interrupt clear
- DMA_START is the start trigger of xDMA

The start address pointer is a double buffer design and it is latched into xDMA control whenever DMA_START triggers xDMA to start. The xDMA control will increase the pointer from PTR_START to PRT_END. The addressing mode for increasing pointer steps should be 4-byte or 2-bytes align depending on the memory access type of each peripheral.

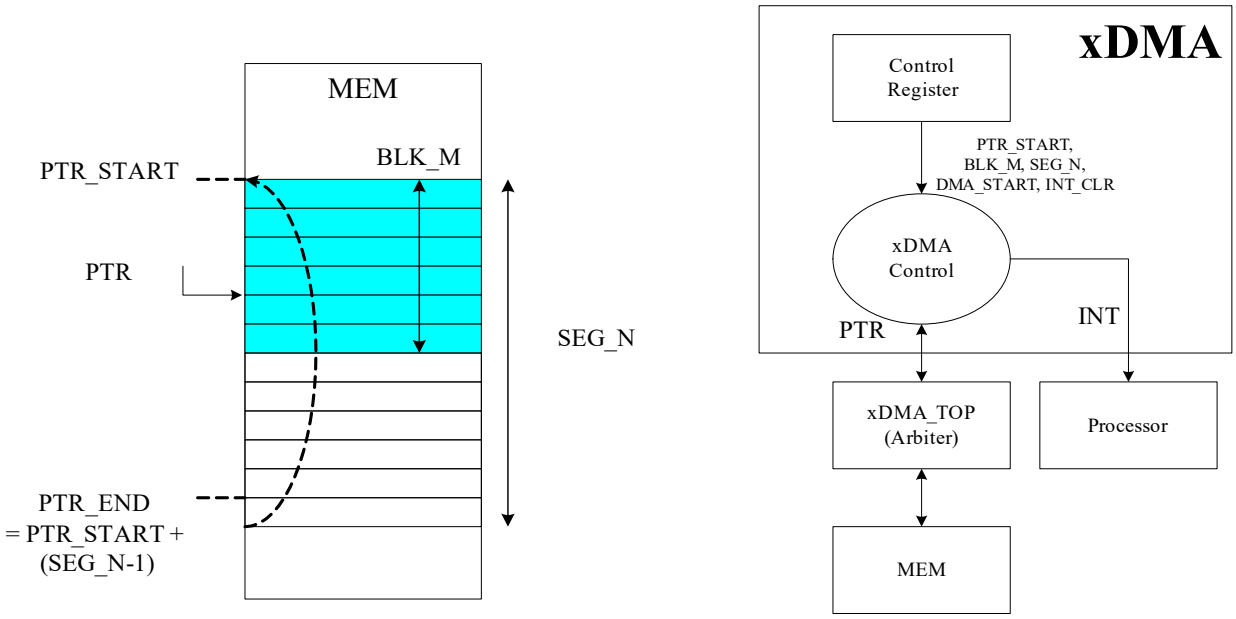


Figure 10-2 Concept of xDMA Operation

10.2.1. Singe-shot Mode

Single-shot mode configures parameters PRT_START and SEG_N as desired values and sets BLK_M to zero to enter single-shot mode. It sets the control register to start xDMA accessing memory from address PTR_START to PTR_END. Once the memory access SEG_N-times is done, the xDMA will issue an interrupt and stop xDMA to access memory. When the processor receives an interrupt it will

prepare a next single-shot access (or not) and clear teh interrupt before exiting the interrupt service routine.

Figure 10-3 is a simple behavior of single-shot xDMA with BLK_M equal to zero.

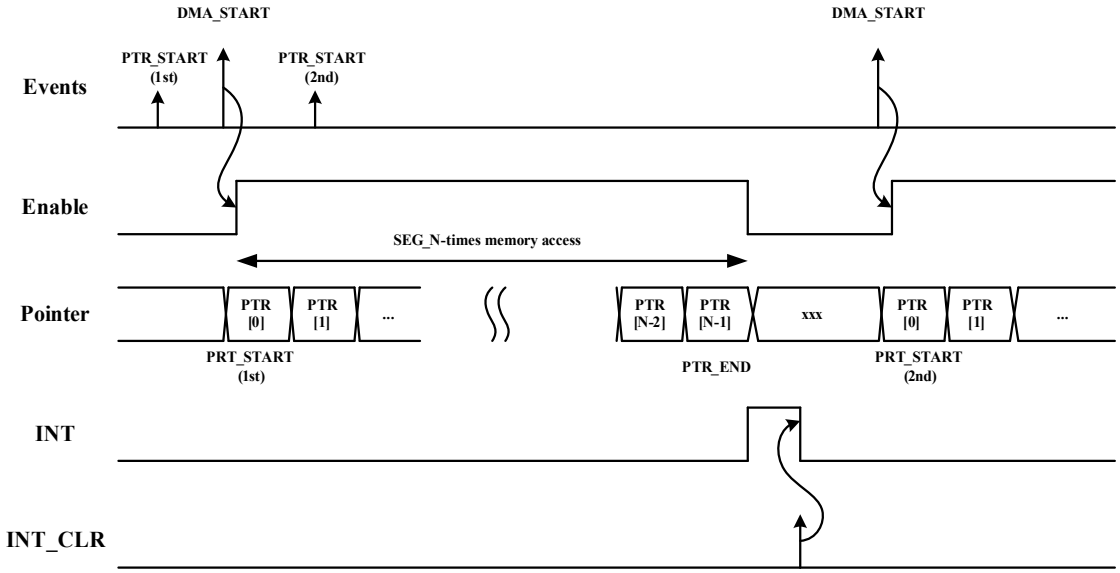


Figure 10-3 An Example of Single-shot Mode of xDMA

10.2.2. Continuous Mode

Continuous mode configures the parameters PRT_START and SEG_N as their desired values and sets BLK_M to non-zero and smaller than SEG_N to enter continuous mode. Sets control register to start xDMA accessing memory from address PTR_START to the PTR_END. Once the memory access BLK_M-times is done, the xDMA will issue an interrupt and continuously access memory. If the pointer reaches PRT_END, it will wrap around to the start address PTR_START for the next memory access (ring buffer function).

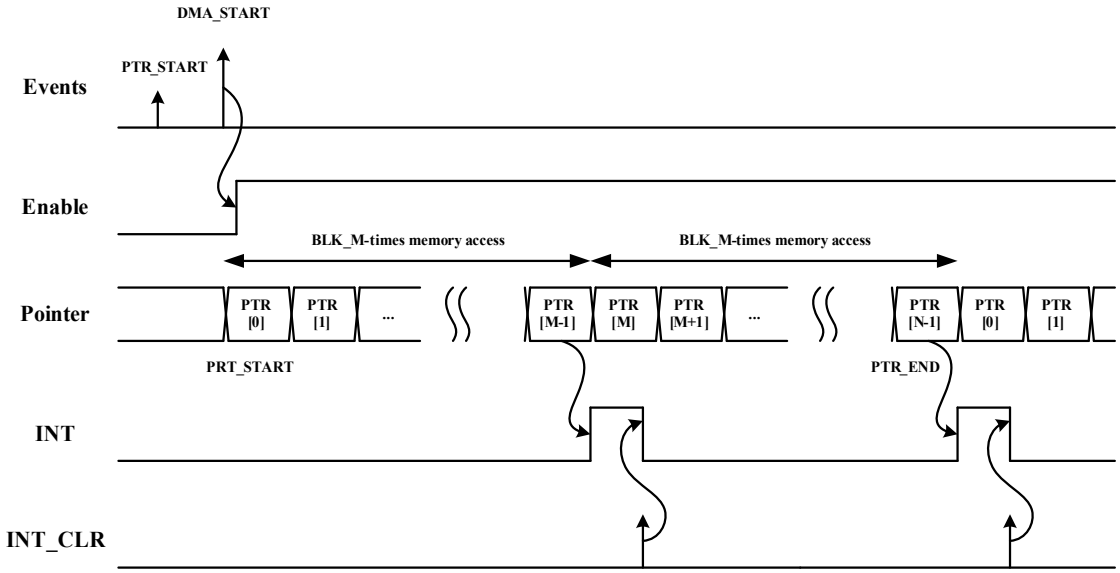


Figure 10-4 An Example of Continuous Mode of xDMA

When the processor receives an interrupt, it can write/read the memory content defined by xDMA for further processing and clear the interrupt before the exit interrupt service routine. Figure 10-4 is a simple behavior of continuous xDMA with SEG_N = 2*BLK_M as a ping-pong buffer configuration.

10.3. Registers

There is only one enable control, cfg_xdma_en, in the xDMA register space. It shall be enabled first whenever a xDMA supported peripheral wants to use the DMA function. Those peripherals have their own register sets for xDMA. Please refer to their register descriptions for the detail.

- Offset 0x00: XDMA_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_xdma_en	[0]	0b	R/W	Enable xDMA. 0b: Disable 1b: Enable

11. PUFrt

CZ20 incorporates PUFrt® from PUFsecurity Corp.. PUFrt® (Root-of-Trust) is a PUF-based (Physical-Unclonable-Function) secure macro which offers the essential features required for establishing a secure Hardware Root of Trust, including a TRNG, UID and Secure OTP. The block diagram of PUFrt is illustrated in Figure 11-1.

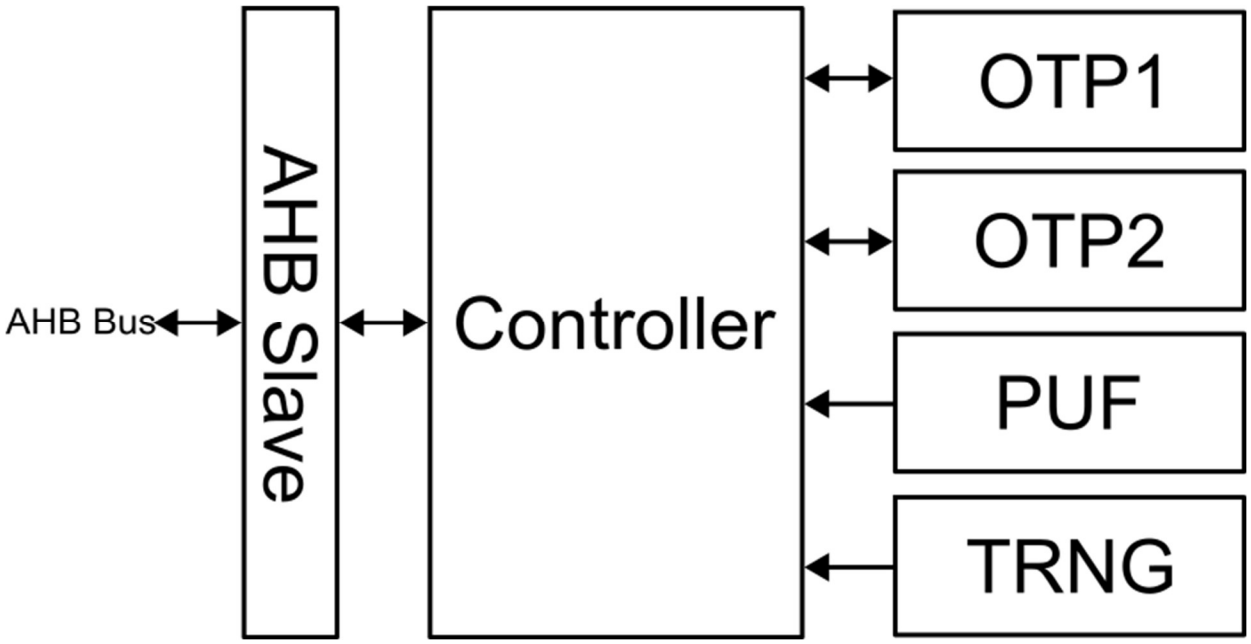


Figure 11-1 Block Diagram of PUFrt

11.1. PUF

Based on NeoFUF® technology, the PUF has the near ideal characteristics of 50% HW (Hamming-Weight), 50% Inter-HD (Hamming-Distance), 0% Intra-ID HD and 0ppm BER (Bit-Error-Rate). It is able to be used for generating true random bits, which can act as a silicon fingerprint. NeoFUF® passes the NIST SP800-22 and NIST SP800-90B IID statistical analysis test suites.

Due to the inherent randomness and uniqueness of the PUF, it can be used directly as a Unique Identifier (UID) and/or a Hardware Unique Key (HUK). Additionally, it can serve as the seed for any key derivation function (KDF) to generate a HUK.

The PUF has a total size of 1024 bits, which is divided into 32 words (puf_000 to puf_031) in the PUFrt

register space. It provides a high level of security and uniqueness, making it highly adaptable for various cryptographic and authentication applications. For instance, if an application requires a 256-bit key, users can derive up to four unique keys from the PUF. These keys could be organized as key0 from puf_000 to puf_007, key1 from puf_008 to puf_015, key2 from puf_016 to puf_023, and key3 from puf_024 to puf_031.

11.2. OTP

The PUFrt contains two OTPs, labeled OTP1 and OTP2. Each OTP has a total size of 1K bytes, divided into 256 words: otp1_000 to otp1_255 for OTP1 and otp2_000 to otp2_255 for OTP2. Each OTP word is protected by ECC to ensure data integrity. As a result, programming the OTP requires writing the entire word, and byte-level writes are not permitted. Any unprogrammed OTP word will return 0x0 when read.

Users can store data in otp1_000 to otp1_191 and otp2_064 to otp2_255. The remaining OTP words are reserved for Secure Boot. For details on how to enable Secure Boot, please refer to the "CZ20_Secure_Boot_Guide." A summary of OTP usage is provided in Table 11-1.

Table 11-1 Summary of OTP Usage

OTP Words	Usage
otp1_000 ~ otp1_191	User data
otp1_192 ~ otp1_255	Reserved for Secure Boot.
otp2_000 ~ otp2_063	Reserved for Secure Boot.
otp2_064 ~ otp2_255	User data

All OTP words are memory-mapped, meaning that performing a read or write operation at the specific memory address will allow access to the corresponding OTP word. Before programming the OTP, users must write the correct keyword to the register *sec_otp_write_key*, as outlined in Section 6.5, Registers of the Security Controller, in order to disable the OTP write protection. Without this step, all write attempts to OTP words will be ignored.

Since the entire PUFrt memory space is protected by the *sec_otp_write_key*, this key must also be set before both programming the OTP and enabling the TRNG. Once the write operations are complete, ensure that the key is reset to 0x0 to re-enable the write protection.

11.3. PUF and OTP Zeroization

The zeroization function is an automatic process that programs each PUF or OTP cell to erase the data stored within them, effectively making the data irretrievable. This function is useful for permanently destroying sensitive data when the chip reaches the end of its life. Only PUF and OTP1 support the zeroization function. The granularity of the zeroization function is 8 words, meaning it requires four separate operations to fully zeroize the entire PUF.

While OTP2 does not support the zeroization function, users can manually zeroize it by programming each OTP2 word with the value 0xFFFFFFFF.

The procedures for performing PUF or OTP zeroization are outlined as follows:

- Set the correct key in the *sec_otp_write_key* register to disable the PUFrt write protection.
- Set the appropriate command in the *cmd_zero_puf* or *cmd_zero_otp* register to zeroize the corresponding PUF or OTP words. Refer to the register descriptions of *cmd_zero_puf* and *cmd_zero_otp* for the specific commands.
- Poll the command status using the *check_flag* register until it returns 0x0.
- Repeat steps 2 and 3 to zeroize other PUF or OTP words as needed.
- Once the zeroization is complete, write 0x0 to the *sec_otp_write_key* register to re-enable the PUFrt write protection.

11.4. TRNG

The true random number generator of PUFrt® complies with NIST SP800-90B recommendations. It is equipped with a digital noise source derived from four independent ring oscillators, entropy health tests, and an entropy conditioning engine based on the PUF.

To generate random data, follow these steps:

- Set the correct key in the *sec_otp_write_key* register to disable PUFrt write protection.
- Enable the TRNG clock by writing 1 to the *rn_clk_en* register.
- Enable the TRNG function by writing 1 to the *rn_func_en* register.
- Poll the *rn_run_st* status bit until it becomes 1.
- Read the random data from the *rn_data* register.

- Disable the TRNG function and clock.
- Write 0x0 to the *sec_otp_write_key* register to re-enable PUFrt write protection.

11.5. Registers

- Offset 0x000~0x3FC: OTP2

Signal	Bits	Default	R/W	Description
otp2_xxx	[31:0]	0x0	R/W→R	OTP2 location xxx (000~255). All OTP shall be programmed once except zeroization.

- Offset 0x400~0x7FC: OTP1

Signal	Bits	Default	R/W	Description
otp1_xxx	[31:0]	0x0	R/W→R	OTP1 location xxx (000~255). All OTP shall be programmed once except zeroization.

- Offset 0x800~0x87C: PUF

Signal	Bits	Default	R/W	Description
puf_xxx	[31:0]	0x0	R	PUF location xxx (000~031).

- Offset 0xA00: TRNG_VERSION

Signal	Bits	Default	R/W	Description
rn_version	[31:0]	0x0	R	Version=0x39304200

- Offset 0xA10: TRNG_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:12]			
rn_halt_st	[11]	0b	R	0b RNG is not halted 1b RNG is halted due to entropy error. RNG error handling procedure is required.
rn_run_st	[10]	0b	R	0b Random data is not generated. 1b Random data is generated and can be read from rn_data.
rn_chk_st	[9]	0b	R	0b Health test is inactive 1b Health test is active
rn_idle_st	[8]	0b	R	0b RNG is enabled 1b RNG is not enabled
reserved	[7:3]			
rn_entropy_st	[2]	0b	R	0b Entropy source is not available 1b Entropy source is available
rn_fifo_st	[1]	0b	R	0b FIFO are not yet cleared for software 1b FIFO are cleared for software
rn_ctrl_st	[0]	0b	R	0b Controller is idle/puse 1b Controller is busy

- Offset 0xA20: TRNG_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
rn_clk_en	[1]	1b	R/W	TRNG clock enable. 0b disable 1b enable
rn_func_en	[0]	0b	R/W	TRNG function enable 0b disable 1b enable

- Offset 0xA70: TRNG_DATA

Signal	Bits	Default	R/W	Description
rn_data	[31:0]	0x0	R	TRNG data output.

- Offset 0xB80: CMD_STATUS

Signal	Bits	Default	R/W	Description
check_flag	[31:0]	0x0	R	[0]: Busy. Indicate if Zeroization has finished (L) or not (H). [1]: Error. A result flag, indicating if Zeroization has passed (L) or failed (H). [2]: Warning. The high of this flag indicates a previous Zeroization is executing. [3]: Wrong. The high of this flag indicates the Zeroization command is not correctly set.

- Offset 0xBA8: PUF_ZEROIZATION

Signal	Bits	Default	R/W	Description
cmd_zero_puf	[31:0]	0x0	W	0x4b: zeroize puf_000~puf_007 0xad: zeroize puf_008~puf_015 0xd2: zeroize puf_016~puf_023 0x34: zeroize puf_024~puf_031

- Offset 0xBB8: OTP_ZEROIZATION

Signal	Bits	Default	R/W	Description
cmd_zero_otp	[31:0]	0x0	W	0x80: zeroize otp1_000~otp1_007 0x81: zeroize otp1_008~otp1_015 0x82: zeroize otp1_016~otp1_023 … 0x9e: zeroize otp1_240~otp1_247 0x9f: zeroize otp1_248~otp1_255

12. Software IRQ

Besides the built-in software interrupts of Cortex-M33, CZ20 provides two additional software interrupts, Software IRQ 0 and Software IRQ 1. Each software interrupt has 32 status bits and one additional data register that can exchange information between software tasks.

12.1. Registers

- Offset 0x00: SOFT_SET_IRQ

Signal	Bits	Default	R/W	Description
sw_set_irq	[31:0]	0x0	W	Write 1 to any bit to trigger the software interrupt.

- Offset 0x04: SOFT_CLK_IRQ

Signal	Bits	Default	R/W	Description
sw_clr_irq	[31:0]	0x0	W	Write 1 to any bit to clear the corresponding interrupt status.

- Offset 0x08: SOFT_IRQ_STATE

Signal	Bits	Default	R/W	Description
state_sw_clr	[31:0]	0x0	R	The status bit of the software interrupt.

- Offset 0x0C: SOFT_IRQ_DATA

Signal	Bits	Default	R/W	Description
data_sw_irq	[31:0]	0x0	R/W	For setting additional information of the software interrupt.

13. Timer

The Timer module is a 32-bit up/down counter with a selectable and programmable pre-scaler whose value can be set between 1 and 1024. The pre-scaler extends the timer's range at the expense of precision.

The Timer provides two modes of operation: free running counting and periodic counting. One-shot counting function is also supported for a single time counting and stopping the timer.

For supporting PWM application in low power mode, the Timer provides simple PWM (pulse-width modulation) function. Timer counts to generate waveform based on the threshold and phase register setting.

Input signal time counting is supported in the Timer. Timer capture the counting value temperately when input signal positive or negative edges.

The Timer contains several configuration registers that can be written and read by the processor. A 10-bit pre-scaler precedes a 32-bit timer counter. The counter value can be loaded from a preload register. The timer interrupt can be enabled and check the status by the control registers.

13.1. Block Diagram

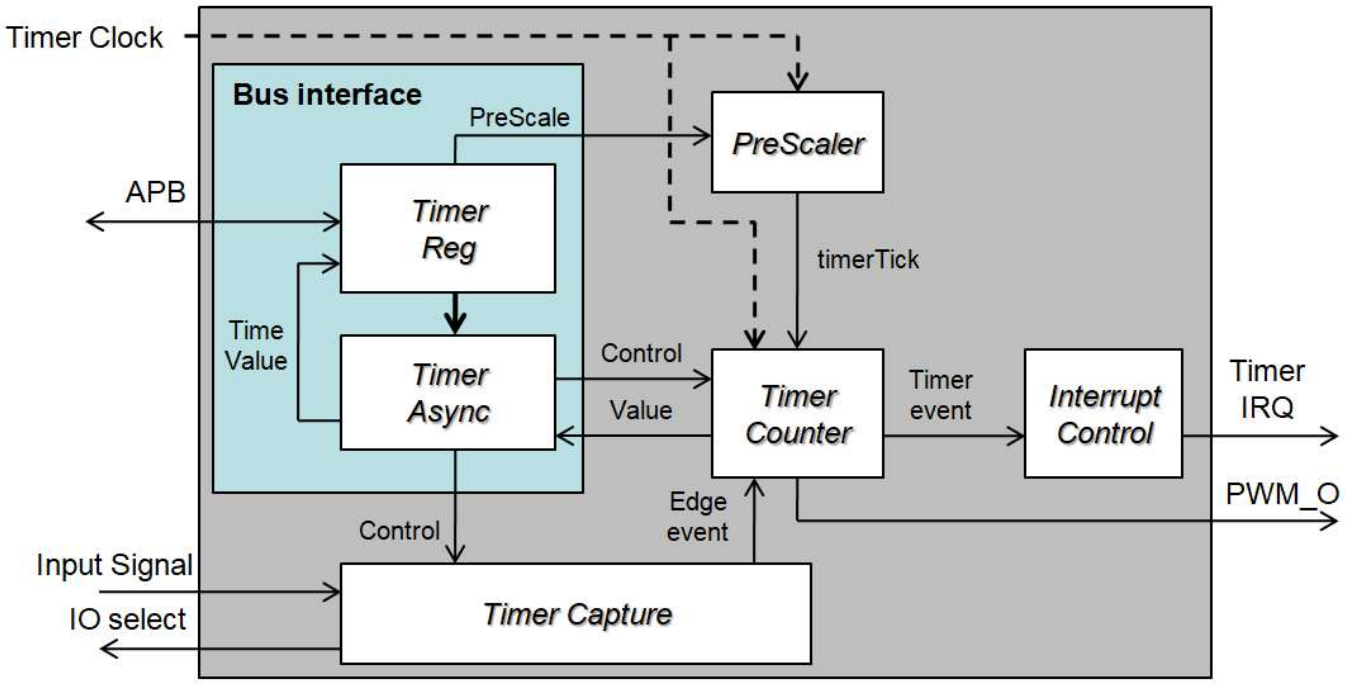


Figure 13-1 Block Diagram of Timer

13.2. Functional Description

13.2.1. Pre-scaler

The timer is a basic 32-bit up/down counter. When a new value is set in the **load** register, the **load** register is loaded into the timer counter. **Prescale** and **prescale_val**, the control registers set the scaling of ***timerTick*** creating a timer clock. **Prescale** register provide 8 different common timer clock rate, and the **prescale_val** register support programmable clock rate if the needed timer clock rate not included in the **Prescale** 8 default timer clock. The ***timerTick*** period is given by the follow formula:

$$timerTick\ period = timer\ clock \times (prescal\_value + 1)$$

13.2.2. Interrupt

The timer counter value can be read from the **Value** register. The ***intReq*** output is activated when timer

count to the **timer_expired_value**. The ***intReq*** is cleared by writing to the **clear** register, and status can be checked from **int_status** register.

13.2.3. Operational Modes

The timer has two modes of operation determined by the Control Register **mode,** free-running and periodic. Counting can select up counting or down counting by **up_counting_mode** register.

Timer start counting values after writing the **enable** register to 1. When the counting value reaches the timer expired value and the free-running mode is selected, timer value count continuously after the interrupt active. If the expired value is 0 and the down counting is continuous, the timer value wrap around to 0xFFFFFFFF. When the timer count reaches expired value and periodic running mode is selected, the **load** register is reloaded into the timer clock. In either case, the difference between these two modes is the value reload or not when the timer count to expired value. The timer support one-shot function, that timer counter stop when the timer expired. Timer one-shot function can be enabled by set **oneshot_en** register.

Below shows some timer counting examples in different modes. **Load** value set to 15, **time_expired_value** set to 0 when down counting mode, and set to 30 when up counting mode. Examples are all down counting except the up counting mode example. After timer is enabled, the timer count value changes every ***timerTick***, and active ***intReq*** when count to **time_expired_value**. Timer counter keep counting when timer expired if **mode** set to 0 (Free running mode), reload timer value if **mode** set to 1 (Period mode), and stop at expired value if **onshot_en** set to 1.

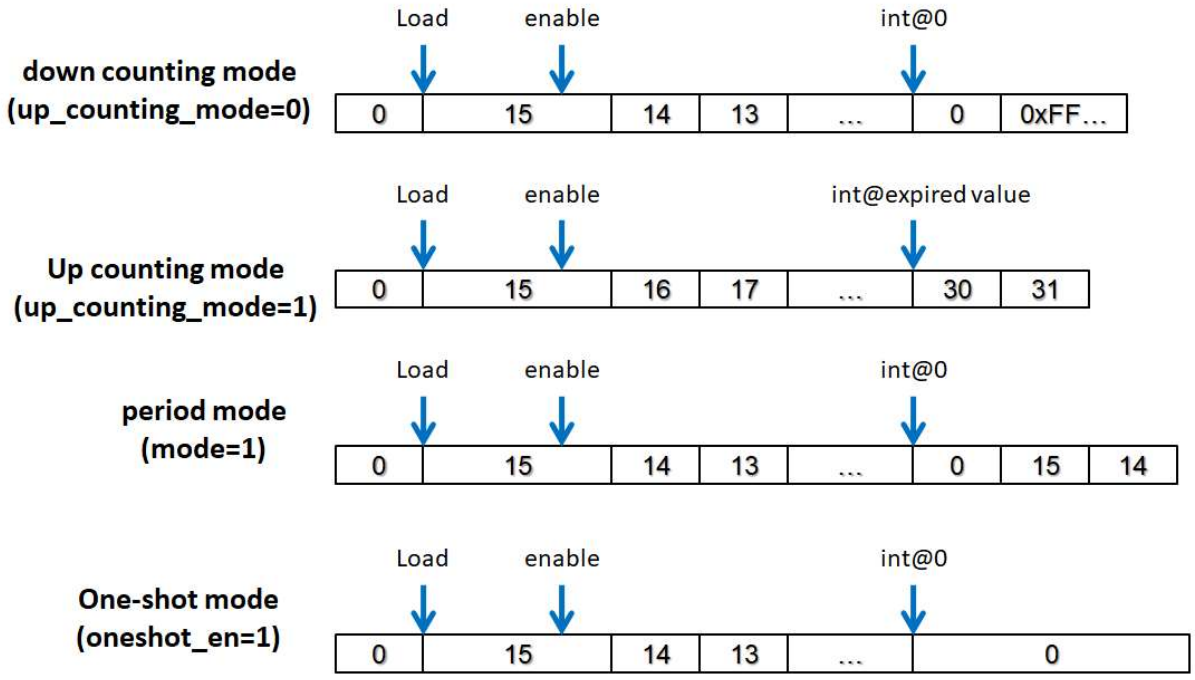


Figure 13-2 Timer Operation Modes

13.2.4. PWM Function

The timer can support simple PWM function. The target is providing simple output waveform generation at low power mode. For simplify the function, the timer only provide **timer_pwm_en** to control PWM function enable and two registers, **value_thd** (value threshold) and **pha** (phase), to configure the PWM output. The timer PWM is based on timer counter and **value_thd** defines the boundary of signal change, **pha** defines the PWM output when timer value>= **value_thd.**

Below picture is timer PWM example, timer **load** is 15 and down count to 0 in both mode. PWM configuration **value_thd** is 11. If **pha** set to 0, pwm_o output high at timer value>= **value_thd.** Otherwise, pwm_o output low at timer value>= **value_thd** if **pha** set to 1. The pwm_o default output is low and pwm_o have 1 ***timerTick*** delay the timer counter value.

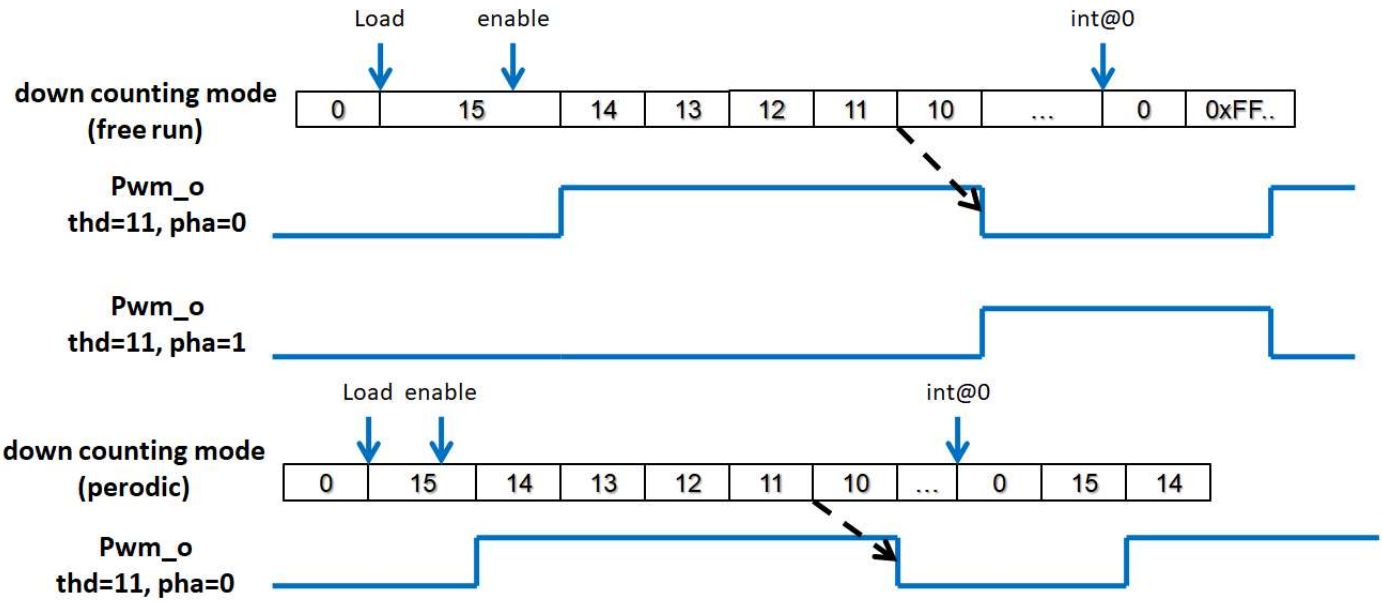


Figure 13-3 PWM Mode of Timer

13.2.5. Input Capture

Input signal timing capture is another additional function of the timer. Signal input from GPIO and timer capture the input edge, generate interrupt and latch the timer value. The timer support 2 channels for input timing capture, each channel have individual input, configuration, and status. Register **ch0_capture_en** to control capture channel0 enable, register **ch0_capture_io_sel** select which GPIO as channel0 input, and read register **ch0_cap_value** after interrupt active to get the latched time value. Below picture shows input signal timing capture example. The timer latches timer value and generate interrupt at positive or negative input signal edge. Set register **ch0_capture_edge** can select positive or negative edge. Simple deglitch function is support for signal capture. Set register **ch0_deglich_en** to 1 can enable the deglitch function, input pulse smaller than 4 timer clock is deglitched.

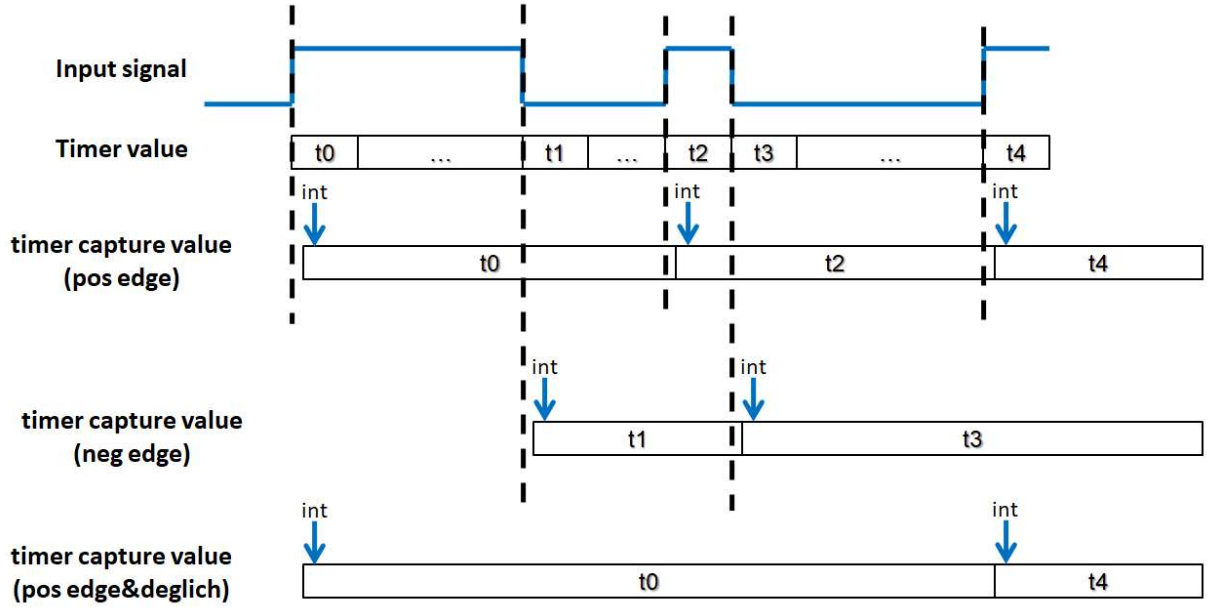


Figure 13-4 Input Capture Mode of Timer

13.3. Registers

- Offset 0x00: TIMER_LOAD

Signal	Bits	Default	R/W	Description
load	[31:0]	0x0	R/W	The initial value of the timer, also the reload value when the timer is in periodic mode

- Offset 0x04: TIMER_VALUE

Signal	Bits	Default	R/W	Description
value	[31:0]	0x0	R	The timer current value.

- Offset 0x08: TIMER_CONTROL

Signal	Bits	Default	R/W	Description
reserved	[31:22]			
ch1_capture_int_en	[21]	b0	R/W	ch1 capture interrupt enable 0:disable 1:enable
ch0_capture_int_en	[20]	b0	R/W	ch0 capture interrupt enable 0:disable 1:enable
ch1_deglich_en	[19]	b0	R/W	ch1 input deglitch enable 0: disable 1:enable
ch0_deglich_en	[18]	b0	R/W	ch0 input deglitch enable 0: disable 1:enable
ch1_capture_edge	[17]	b0	R/W	ch1 capture edge select 0: posedge 1:negedge
ch0_capture_edge	[16]	b0	R/W	ch0 capture edge select 0: posedge 1:negedge
reserved	[15:12]			
ch1_capture_int_status	[11]	b0	R	timer ch1 capture interrupt status

ch0_capture_int_status	[10]	b0	R	timer ch0 capture interrupt status
timer_enable_status	[9]	b0	R	timer enable status at timer clk domain
int_status	[8]	b0	R	timer interrupt status
enable	[7]	b0	R/W	timer enable; one shot mode trigger 0:disable 1:enable
mode	[6]	b0	R/W	timer mode select 0:free running mode 1:periodic mode
int_enable	[5]	b0	R/W	timer interrupt enable 0:disable 1:enable
prescale	[4:2]	b0	R/W	0: timer clock is divided by 1 1: timer clock is divided by 16 2: timer clock is divided by 256 3: timer clock is divided by 2 4: timer clock is divided by 8 5: timer clock is divided by 32 6: timer clock is divided by 128 7: timer clock is divided by 1024 only valid when prescale_val=0
oneshot_en	[1]	b0	R/W	timer one shot mode enable 0: one shot mode disable 1: one shot mode enable
up_counting_mode	[0]	b0	R/W	timer up counting mode enable 0: down counting mode 1: up counting mode

- Offset 0x0C: TIMER_INT_CLR

Signal	Bits	Default	R/W	Description
int_clear	[31:0]		W	Clear Interrupt – write any value to clear the timer interrupt

- Offset 0x10: TIMER_CAPTURE_CLR

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
ch1_capture_int_clear	[1]	b0	W	Write any value to clear timer ch1_capture interrupt status
ch0_capture_int_clear	[0]	b0	W	Write any value to clear timer ch0_capture interrupt status

- Offset 0x14: TIMER_CAPTURE_CH0

Signal	Bits	Default	R/W	Description
ch0_cap_value	[31:0]	b0	R	ch0 capture value, clear when timer load

- Offset 0x18: TIMER_CAPTURE_CH1

Signal	Bits	Default	R/W	Description
ch1_cap_value	[31:0]	b0	R	Ch1 capture value, clear when timer load

- Offset 0x1C: TIMER_PRESCALE_VALUE

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
prescale_val	[9:0]	b0	R/W	timer clock is divided by prescale+1, only valid when>0

- Offset 0x20: TIMER_EXPIRE

Signal	Bits	Default	R/W	Description
timer_expired_value	[31:0]	b0	R/W	timer expired value

- Offset 0x24: TIMER_CAPTURE_ENABLE

Signal	Bits	Default	R/W	Description
reserved	[31:3]			
timer_pwm_en	[2]	0	R/W	Enable Timer PWM mode. 0b disable 1b enable
ch1_capture_en	[1]	0	R/W	Enable Timer capture mode for CH1. 0b disable 1b enable
Ch0_capture_en	[0]	0	R/W	Enable Timer capture mode for CH0. 0b disable 1b enable

- Offset 0x28: TIMER_CAPTURE_IOSEL

Signal	Bits	Default	R/W	Description
reserved	[31:13]			
ch1_capture_io_sel	[12:8]	0x0	R/W	Select the GPIO input for CH1. 0x0 for GPIO00, 0x1 for GPIO01 and so on.
reserved	[7:5]			
Ch0_capture_io_sel	[4:0]	0x0	R/W	Select the GPIO input for CH0. 0x0 for GPIO00, 0x1 for GPIO01 and so on.

- Offset 0x2C: TIMER_PWM_THRESHOLD

Signal	Bits	Default	R/W	Description
value_thd	[31:0]	0x0	R/W	Timer threshold for generate simple PWM waveform

- Offset 0x30: TIMER_PWM_PHA

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
pha	[0]	0	R/W	Timer pha for generate simple PWM waveform.

14. Slow-clock Timer

The Slow-clock Timer is a timer running in the slow clock domain. Because slow_clk is still active in low-power modes, the Slow-clock Timer can be operated in low-power modes normally and can become the wakeup source. With the interrupt repeat function, it is possible to support the very long sleep time.

For basic timer functions, the functions are the same as the Timer described in Chapter 13. The input capture mode and the PWM mode are not supported by the Slow-clock Timer.

14.1. Block Diagram

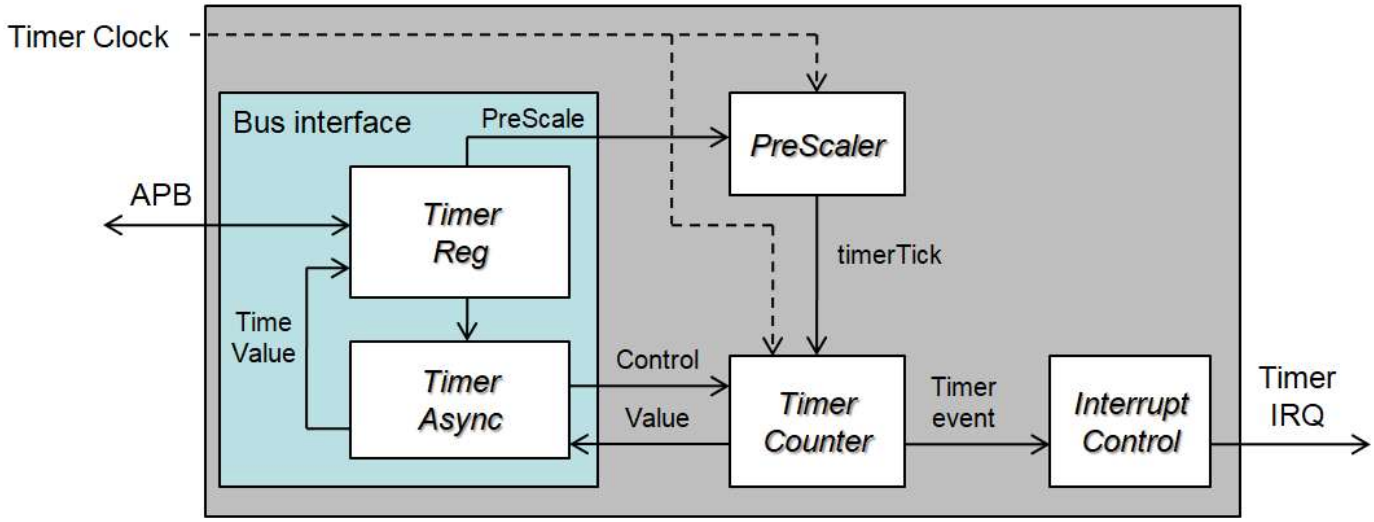


Figure 14-1 Block Diagram of Slow-clock Timer

14.2. Functional Descriptions

14.2.1. Pre-scaler

The timer is a basic 32-bit up/down counter. When a new value is set in the **load** register, the **load** register is loaded into the timer counter. **Prescale** and **prescale_val**, the control registers set the scaling of ***timerTick*** creating a timer clock. **Prescale** register provide 8 different common timer clock rate, and the **prescale_val** register support programmable clock rate if the needed timer clock rate not included in the **Prescale** 8 default timer clock. The ***timerTick*** period is given by the follow formula:

$$timerTick \text{ period} = \text{timer clock} \times (prescal\_value + 1)$$

14.2.2. Interrupt

The timer counter value can be read from the **Value** register. The ***intReq*** output is activated when timer count to the **timer_expired_value,** if the interrupt repeat function is disabled. The ***intReq*** is cleared by writing to the **clear** register, and status can be checked from **int_status** register.

14.2.3. Interrupt Repeat Delay

For courting very long period in low power mode, interrupt repeat function is supported. Interrupt repeat function is a counter in interrupt control for counting timer expired event numbers. If the interrupt repeat function is not disabled, the timer expired event counter add counting number when timer event happens. **Int_repeat_delay** register set the value, that timer expired event counting value need match, to active ***intReq.***

14.2.4. Operational Modes

The timer has two modes of operation determined by the Control Register **mode,** free-running and periodic. Counting can select up counting or down counting by **up_counting_mode** register.

Timer start counting values after writing the **enable** register to 1. When the counting value reaches the timer expired value and the free-running mode is selected, timer value count continuously after the interrupt active. If the expired value is 0 and the down counting is continuous, the timer value wrap around to 0xFFFFFFFF. When the timer count reaches expired value and periodic running mode is selected, the **load** register is reloaded into the timer clock. In these two modes, the timer both count continuously, the difference is that the timer count value reload or not when the timer count to expired value. The timer support one-shot function, that timer counter stop when the timer expired. Timer one-shot function can be enabled by set **oneshot_en** register.

Below is the timer counting example in different modes. **Load** value set to 15, **time_expired_value** set to 0 when down counting mode, and set to 30 when up counting mode. Examples are all down counting except the up counting mode example. After timer is enabled, the timer count value changes every ***timerTick***, and active ***intReq*** when count to **time_expired_value**. Timer counter keep counting when timer expired if **mode** set to 0 (Free running mode), reload timer value if **mode** set to 1 (Period mode), stop at expired value if **onshot_en** set to 1.

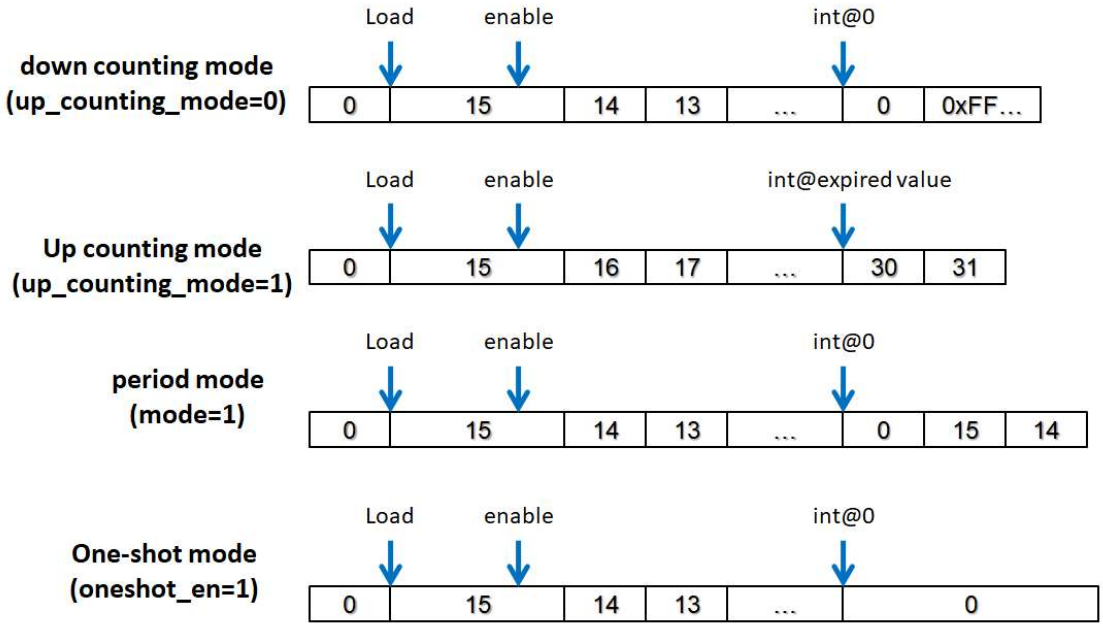


Figure 14-2 Operation Modes of Slow-clock Timer

14.3. Registers

- Offset 0x00: TIMER32K_LOAD

Signal	Bits	Default	R/W	Description
load	[31:0]	0x0	R/W	The initial value of the timer, also the reload value when the timer is in periodic mode

- Offset 0x04: TIMER32K_VALUE

Signal	Bits	Default	R/W	Description
value	[31:0]	0x0	R	The timer current value.

- Offset 0x08: TIMER32K_CONTROL

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
timer_enable_status	[9]	b0	R	timer enable status at timer clk domain
int_status	[8]	b0	R	timer interrupt status
enable	[7]	b0	R/W	timer enable; one shot mode trigger 0:disable 1:enable
mode	[6]	b0	R/W	timer mode select 0:free running mode 1:periodic mode

int_enable	[5]	b0	R/W	timer interrupt enable 0:disable 1:enable
prescale	[4:2]	b0	R/W	0: timer clock is divided by 1 1: timer clock is divided by 16 2: timer clock is divided by 256 3: timer clock is divided by 2 4: timer clock is divided by 8 5: timer clock is divided by 32 6: timer clock is divided by 128 7: timer clock is divided by 1024 only valid when prescale_val=0
oneshot_en	[1]	b0	R/W	timer one shot mode enable 0: one shot mode disable 1: one shot mode enable
up_counting_mode	[0]	b0	R/W	timer up counting mode enable 0: down counting mode 1: up counting mode

- Offset 0x0C: TIMER32K_INT_CLR

Signal	Bits	Default	R/W	Description
int_clear	[31:0]		W	Clear Interrupt – write any value to clear the timer interrupt

- Offset 0x10: TIMER32K_INTERRUPT_REPEAT

Signal	Bits	Default	R/W	Description
reserved	[31:17]			
int_repeat_delay_disable	[16]	b0	R/W	timer interrupt repeat delay function disable 0: enable int_repeat_delay 1: disable int_repeat_delay
int_repeat_delay	[15:0]	b0	R/W	timer interrupt repeat delay for times, for sleep mode only

- Offset 0x14: TIMER32K_PRESCALE_VALUE

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
prescale_val	[9:0]	b0	R/W	timer clock is divided by prescale+1, only valid when>0

- Offset 0x18: TIMER32K_EXPIRE

Signal	Bits	Default	R/W	Description
timer_expired_value	[31:0]	b0	R/W	timer expired value

15. Watchdog Timer

Watchdog Timer is very common in current system to prevent some unexpected condition happen to stuck the whole system. The Watchdog Timer is a down counting timer which can trigger the system reset and reset the stuck system. The Watchdog Timer need to be “kick” periodic to reload the counting value, or system will be reset when the Watchdog Timer count to 0.

This module implements a 32-bit down counter with selectable or programmable pre-scaler to produce a watchdog reset signal, as well as a warning interrupt. The reset and interrupt can be enabled separately.

The Watchdog Timer also supports the Windowing feature. This feature makes the watchdog timer not only have to be kicked in a certain period, but also that can’t be kicked too frequently. This mechanism protects the system can be reset when stuck in an always kicking situation.

15.1. Block Diagram

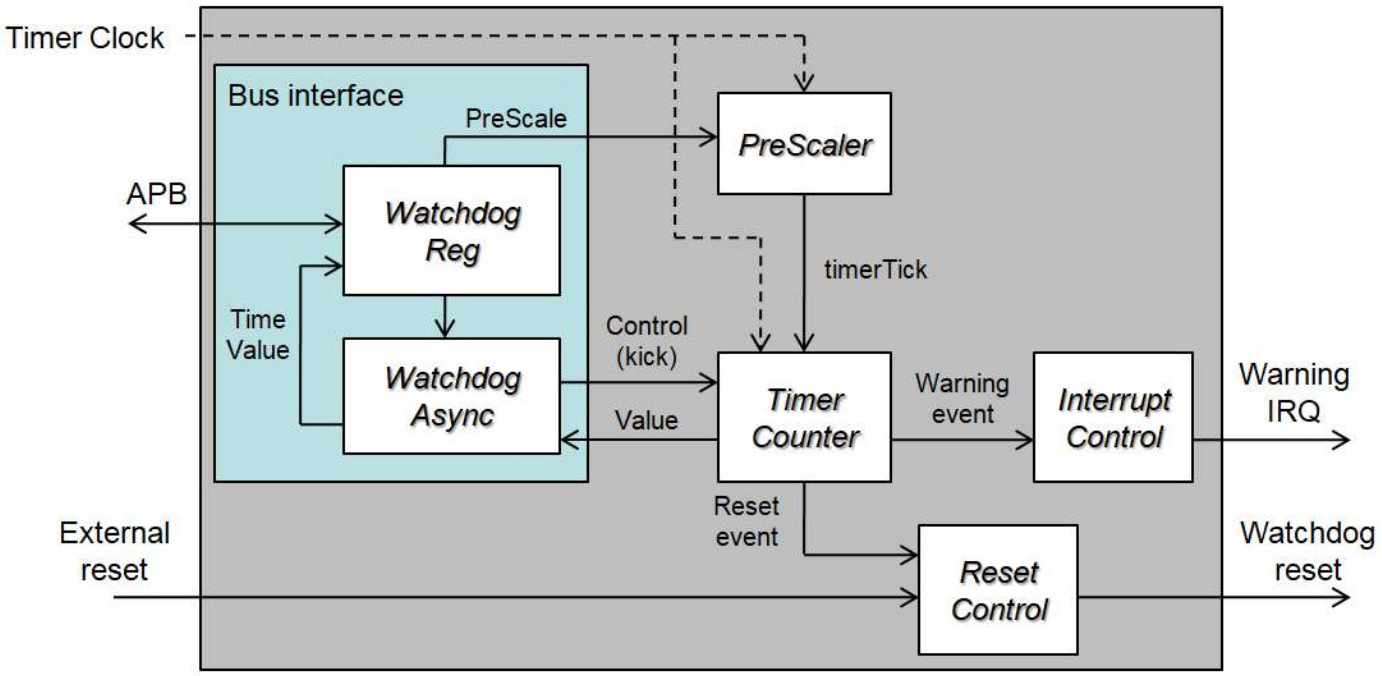


Figure 15-1 Block Diagram of Watchdog Timer

15.2. Functional Descriptions

15.2.1. Pre-scaler

The ***timerTick*** generate from the 12 bit ***Prescaler***. The ***Prescaler*** supports programmable or selectable prescale value. The **prescale** register provide 8 different common prescale value, but the selected prescale value work only if the **prescale_value** register is set to 0. If the **prescale_value** register is not 0, the **prescale_value** register provide programmable 12 bit prescale value. The ***timerTick*** period is given by the follow formula:

$$timerTick = timer\ clock \times (prescale_{value} + 1)$$

The timer clock is the ***Timer Clock*** period and prescale_value is **prescale_value** register setting when not 0.

15.2.2. Windowing

The **windowMax** register set the reload value and load value to timer counter after **windowMax** register writing. After **enable** register is set to 1(enable), the timer counter count down at every ***timerTick***. The **windowMin** register define the minimum kick time period to generate the reset event. The **windowMin** setting only work if the value of **windowMin**<**windowMax** and the **windowMin** is not 0. The Watchdog reset can be enabled by **reset_enable** register.

15.2.3. Kick

The **kick** register is a write only register to kick the Watchdog timer and reload the timer value. For kicking the watchdog timer, needing to write specific pattern (0x0000A5A5) into the **kick** register.

15.2.4. Warning IRQ

The ***Warning event*** is generated by timer counter, the setting of **int_value** register define the time of event happening. The ***Warning IRQ*** can be enabled by **int_enable** register, and cleared by writing the **clear** register. If the timer count to 0 or is kicked too frequently, the reset event happen and trigger the reset control function.

15.2.5. Examples

Figure 15-2 shows examples when the Watchdog Timer working. The **WindowMax** register value set 15 and **WindowMin** register value set 10. The first example is normal counting without any kick. The reset happen after counting to 0. The second example is a legal kick, kick the timer in **windowMin** and reload the timer value. The third function is an illegal kick happen, kick the timer out of **windowMin** and trigger the reset event. The red color timer counting values in below figure are the illegal kick timer values when **WindowMin** register value set 10.

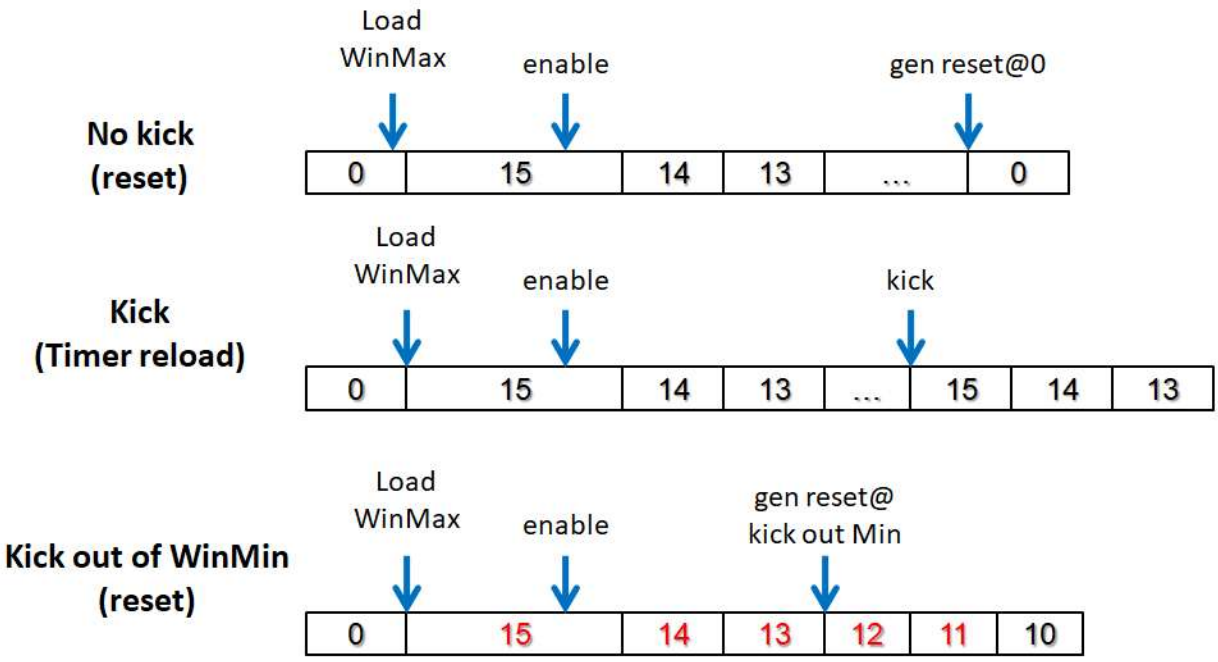


Figure 15-2 Examples of Watchdog Timer

To prevent modify the watchdog registers cause the system reset unexpectedly. There have setting lock function for the Watchdog Timer. The main registers (window value and control registers) setting can be protected with the **lockout** register set to 1. If the Watchdog Timer is locked, the only way to unlock is reset the Watchdog Timer.

15.3. Registers

- Offset 0x00: WDT_WINDOW_MAX

Signal	Bit	Default	R/W	Description
windowMax	[31:0]	b0	R/W	The watchdog window maximum value, and also reload this value when have a specific write to "Kick" register

- Offset 0x04: WDT_TIMER_VAL

Signal	Bit	Default	R/W	Description
timeVal	[31:0]	b0	R	The timer current value. This value is from timerClk domain synchronized to PCLK domain.

- Offset 0x08: WDT_CONTROL

Signal	Bit	Default	R/W	Description
reserved	[31:8]			
enable	[7]	b0	R/W	Wdtimer enable 0:disable 1:enable
int_enable	[6]	b0	R/W	Wdtimer interrupt enable 0:disable 1:enable
reserved	[5:1]			
lockout	[0]	b0	R/W	watch dog timer lock 0: unlock 1: lock when locked, windowMax, control,Timer reset ocurr clear, Timer interrupt, windowMin can't be changed

- Offset 0x0C: WDT_KICK

Signal	Bit	Default	R/W	Description
Kick	[31:0]	0	W	WatchDog timer kick register，write this register with specific value 0xa5a5 to kick and reload the watchdog timer

- Offset 0x10: WDT_RESET_OCCURE

Signal	Bit	Default	R/W	Description
reserved	[31:8]			
reset_occur_clear	[7:0]	0	R/W	reset occur counter，write any value to clear the reset occur counter，Read this register to read current reset occur value

- Offset 0x14: WDT_CLEAR

Signal	Bit	Default	R/W	Description
clear	[31:0]	0	W	Write any value to clear the Wdtimer interrupt

- Offset 0x18: WDT_VALUE

Signal	Bit	Default	R/W	Description
int_value	[31:0]	0	R/W	Timer interrupt value. Set to send warning interrupt when interrupt value is meet.

- Offset 0x1C: WDT_WIN_MIN

Signal	Bit	Default	R/W	Description
windowMin	[31:0]	0	R/W	If windowMin=0 or windowMin > windowMax this feature is unused. Window minimum value to prevent kick too frequently.

- Offset 0x20: WDT_PRESCALE

Signal	Bit	Default	R/W	Description
reserved	[31:12]	0		
prescale_val	[11:0]	0	R/W	Timer clock is divided by prescale+1, only valid when>0

16. RTC Timer

The RTC (Real Time Clock) timer provides the calendar function and maintains accurate date and time information. It enables the device to keep track of the current time and date, day of the week, month, and year, represented in a BCD format, which is a binary encoding for decimal numbers. It also allows the system to keep track of time for various functions such as scheduling tasks, time-stamping events, coordinating communication protocols, and maintaining accurate time for applications like alarms, timers, and logging.

16.1. Block Diagram

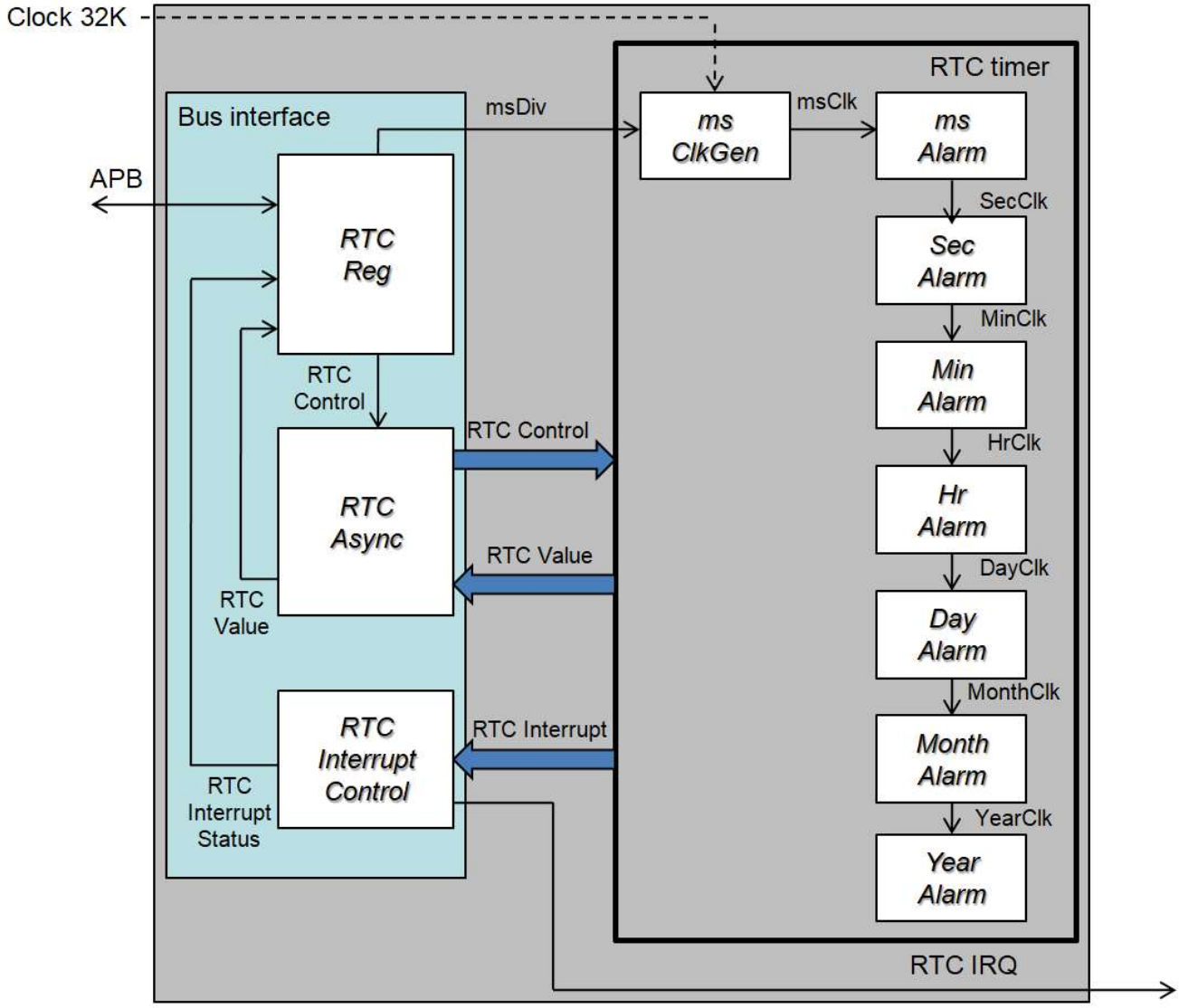


Figure 16-1 Block Diagram of RTC Timer

16.2. Function Description

The RTC Timer has the following features:

- The calendar in BCD format
  - MilliSeconds

- Seconds
- Minutes
- Hours
- Days
- Months
- Years

- Alarm for each “Time Unit”
- Event Alarm for “Time Unit” combination
- Interrupt control

16.2.1. Clock Generate

The RTC Timer counts the time based on the slow_clk and generates a base tick of 1 millisecond. The block “***ms ClkGen”*** generates the millisecond clock tick for counting every millisecond time period when **enable** set to 1. Since the frequency of slow_clk may vary for different applications, the **ms_DivVal** register can be set to adjust the base tick to exactly 1 millisecond. The new **ms_DivVal** value is not applied to the RTC timer until the **Ld_DivVal** register is written.

16.2.2. Calendar

The value of each time unit counter can be loaded from the registers. The time value can be set into timer value registers (0x10~0x14) and write the **Ld_timeVal** register to load value into Time Unit counters. The RTC counting continuously after the register value is loaded into timer.

Each Time Unit counter count time in BCD format and generate tick to trigger the next Time Unit counter at specific time event (for example, ***MinClk*** triggers when second value is 59 and ***SecClk*** trigger). The Day Time Unit already include the day numbers information every month and the day numbers which related to years, but for the special case when the year number can be divisible by 400, the register **year400_flag** need set to 1 at this case.

The table below shows the BCD counting range and carry trigger event value at etch Time Unit counter. When the Time Unit count to the carry trigger event value, the Timer Unit generate carry pulse clock to next Timer Unit and return the counting value to the smallest BCD counting value. The Time Unit counting BCD value can be read from register (0x00~0x14). The value read from these registers is RTC counting value. These registers are also used for setting each Time Unit counter counting value, writing

the wanted value into the registers and active time value load by writing 1 to **Ld_timeVal**. The wanted value load into RTC Time Unit counter after loading the synchronize value to the slow_clk domain.

Table 16-1 The BCD Format of RTC Calendar

Timer Unit	Hundreds digits	tens digits	digits	BCD counting	Carry trigger event value
MilliSecond	0~9	0~9	0~9	0~999	1000
Second	0	0~5	0~9	0~59	60
Minute	0	0~5	0~9	0~59	60
Hour	0	0~2	0~9	0~23	24
Day	0	0~3	0~9	0~31	28 or 29 @ Month=2
					30 @ Month=4, 6, 9, 11
					31 @ Month=1, 3, 5, 7, 8,10, 12
Month	0	0~1	0~9	1~12	12
Year	0	0~9	0~9	00~99	100

16.2.3. Alarm

The alarm function provide user to set a specific alarm time event and RTC generate interrupt when time event happen. The offset address of alarm registers are 0x020~0x34. Each Time Unit can set the alarm value and generate alarm IRQ independently. The alarm value load into timer counters after the **Ld_Alarm_timeVal** register set 1. Besides generating alarm IRQ at the alarm setting time, by changing the mode register setting, each Time Unit can generating alarm IRQ when time value change. As minute Time Unit for example, when **min_irq_mode** register set 1, the minute Timer unit generate IRQ at every minute value change.

16.2.4. Alarm Event

Besides the IRQ of each Time Unit, the event IRQ provides the alarm with different Time Unit combination. By set the **irq_repeat** registers and **irq_mode** registers, providing more combination of alarm time. There are some event time alarm IRQ examples shows in below picture. First case is normal event IRQ, with default setting and only set time alarm time value. The IRQ happen when all Time Unit match the alarm value. Second case is an example that **evt_year_irq_repeat** register is set 1. When the irq_repeat of the Time Unit is set 1, the IRQ trigger when all the Time Units, which timescale are smaller than year, match their alarm value. The third and forth case shows those only the irq_repeat of the smallest Time Unit is functional when multiple irq_repeat registers are set. The fifth case shows the

combination of irq_repeat and irq_mode. If the irq_mode register of Time Unit set 1, the Time Unit alarm value is don't cared.

- All Time Unit alarm value set 07, and irq repeat all set 0. (No irq repeat)
  The IRQ happen RTC time is 07/07/07, 7:7:7(Y/M/D, Hr:Min:Sec)
- If evt_year_irq_repeat set 1
  The IRQ happen RTC time is xx/07/07, 7:7:7(Y/M/D, Hr:Min:Sec) (Every year happen)
- If evt_month_irq_repeat set 1
  The IRQ happen RTC time is xx/xx/07, 7:7:7(Y/M/D, Hr:Min:Sec) (Every month happen)
- If evt_year_irq_repeat set 1 and evt_month_irq_repeat set 1
  The IRQ happen RTC time is xx/xx/07, 7:7:7(Y/M/D, Hr:Min:Sec) (Every month happen)
  (Ignore the evt_year_irq_repeat setting, smaller time unit have higher IRQ repeat priority)
- If evt_year_irq_repeat set 1 and min_irq_mode set 1
  The IRQ happen RTC time is xx/xx/07, 7:xx:7(Y/M/D, Hr:Min:Sec)
  (Every month happen, minute alarm don't care)

16.2.5. Interrupt

The interrupts of each Time Unit and event can be controlled independently. The **int_en** registers control the enable function of interrupts. The interrupt status can be read from the **int_st** registers and the interrupts can be cleared by writing the **int_clr** registers.

16.2.6. Test Mode

Due to some Time Unit, like month or year, need long time period to change value. For the testability, test mode is included in the RTC design. The **testMode** register controls test mode enable of each Time Unit counter. Each Time Unit counter change value every second in test mode for shortening the verification time.

The **ClrPls** register control the RTC counter reset. After the **ClrPls** register writing 1, all Time Unit counter clear the setting to initial value.

16.3. Registers

- Offset 0x00: RTC_SECOND

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
SecVal_00_50	[6:4]	0x0	R/W	The RTC Timer second setting value tens digits in BCD format.
SecVal_0_9	[3:0]	0x0	R/W	The RTC Timer second setting value digits in BCD format.

- Offset 0x04: RTC_MINUTE

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
MinVal_00_50	[6:4]	0x0	R/W	The RTC Timer minute setting value tens digits in BCD format.
MinVal_0_9	[3:0]	0x0	R/W	The RTC Timer minute setting value digits in BCD format.

- Offset 0x08: RTC_HOUR

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
HrVal_00_20	[5:4]	0x0	R/W	The RTC Timer hour setting value tens digits in BCD format.
HrVal_0_9	[3:0]	0x0	R/W	The RTC Timer hour setting value digits in BCD format.

- Offset 0x0C: RTC_DAY

Signal	Bits	Default	R/W	Description
Reserved	[31:6]			
DayVal_00_30	[5:4]	0x0	R/W	The RTC Timer day setting value tens digits in BCD format.
DayVal_0_9	[3:0]	0x0	R/W	The RTC Timer day setting value digits in BCD format.

- Offset 0x10: RTC_MONTH

Signal	Bits	Default	R/W	Description
reserved	[31:5]			
MonthVal_00_10	[4]	0x0	R/W	The RTC Timer month setting value tens digits in BCD format.
MonthVal_0_9	[3:0]	0x0	R/W	The RTC Timer month setting value digits in BCD format.

- Offset 0x14: RTC_YEAR

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
YearVal_00_90	[7:4]	0x0	R/W	The RTC Timer year setting value tens digits in BCD format.
YearVal_0_9	[3:0]	0x0	R/W	The RTC Timer year setting value digits in BCD format.

- Offset 0x18: RTC_CONTROL

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
enable	[8]	0	R/W	RTC timer counter enable control, 0:disable 1:enable
ClrPls	[7]	0	R/W	When 1 is written, creates a pulse on the clk32k domain to clear all counters and values there to their initial state.

				A write of 0 to this bit has no effect. Read: 1=Clear in progress, 0=Clear done.
year400_flag	[6]	0	R/W	If year div 400=0, this flag set to 1.
testMode	[5:0]	0x0	R/W	Used to speed up Time Unit counters for test. If a bit is set to 1, then the corresponding time unit counter will change on the same timing as Seconds. [0]=Unused, [1]=Minutes, [2]=Hours, [3]=Days, [4]=Months, [5]=Years.

- Offset 0x1C: RTC_CLK_DIV

Signal	Bits	Default	R/W	Description
reserved	[31:24]			
ms_DivVal	[23:0]	0x200000	R/W	millisecond Divided setting value of clk32K. Setting to generate 1 ms Clk. ms_DivVal[23:16] is integer part and can't be 0, ms_DivVal[15:0] is fractional part Ex. 32.768K, 32.768=0x20.c49c, ms_DivVal=20c49c

- Offset 0x20: RTC_SEC_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
evt_sec_irq_repeat	[9]	0	R/W	if set to 1, event irq trigger at every second
sec_irq_mode	[8]	0	R/W	sec irq mode setting value Irq mode=0, irq at alarm second Irq mode=1, irq at every second
reserved	[7]			
Alarm_SecVal_00_50	[6:4]	0x0	R/W	Second Timer alarm setting value tens digits in BCD format.
Alarm_SecVal_0_9	[3:0]	0x0	R/W	Second Timer alarm setting value digits in BCD format.

- Offset 0x24: RTC_MIN_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
evt_min_irq_repeat	[9]	0	R/W	Note: This field is relevant only when evt_sec_irq_repeat is not set if set to 1, event irq trigger at every minute alarm match
min_irq_mode	[8]	0	R/W	min irq mode setting value Irq mode=0, irq at alarm minute Irq mode=1, irq at every minute
reserved	[7]			
Alarm_MinVal_00_50	[6:4]	0x0	R/W	Minute Timer alarm setting value tens digits in BCD format.
Alarm_MinVal_0_9	[3:0]	0x0	R/W	Minute Timer alarm setting value digits in BCD format

- Offset 0x28: RTC_HR_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
evt_hr_irq_repeat	[9]	0	R/W	Note: This field is relevant only when evt_sec_irq_repeat and evt_min_irq_repeat is not set if set to 1, event irq trigger at every second alarm match and minute alarm match
hr_irq_mode	[8]	0	R/W	hour irq mode setting value Irq mode=0, irq at alarm hour Irq mode=1, irq at every hour
reserved	[7:6]			
Alarm_HrVal_00_20	[5:4]	0x0	R/W	Hour Timer alarm setting value tens digits in BCD format
Alarm_HrVal_0_9	[3:0]	0x0	R/W	Hour Timer alarm setting value digits in BCD format

- Offset 0x2C: RTC_DAY_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
evt_day_irq_repeat	[9]	0	R/W	Note: This field is relevant only when evt_sec_irq_repeat, evt_min_irq_repeat and evt_hr_irq_repeat is not set if set to 1, event irq trigger at every second alarm match, minute alarm match, and hour alarm match
day_irq_mode	[8]	0	R/W	day irq mode setting value Irq mode=0, irq at alarm day Irq mode=1, irq at every day
reserved	[7:6]			
Alarm_DayVal_00_30	[5:4]	0x0	R/W	Day Timer alarm setting value tens digits in BCD format
Alarm_DayVal_0_9	[3:0]	0x0	R/W	Day Timer alarm setting value digits in BCD format

- Offset 0x30: RTC_MONTH_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
evt_month_irq_repeat	[9]	0	R/W	Note: This field is relevant only when evt_sec_irq_repeat, evt_min_irq_repeat ,evt_hr_irq_repeat and evt_day_irq_repeat is not set if set to 1, event irq trigger at every second alarm match, minute alarm match, hour alarm match, and day alarm match
month_irq_mode	[8]	0	R/W	month irq mode setting value Irq mode=0, irq at alarm month Irq mode=1, irq at every month
Reserved	[7:5]			
Alarm_MonthVal_00_10	[4]	0x0	R/W	Month Timer alarm setting value tens digits in BCD format
Alarm_MonthVal_0_9	[3:0]	0x0	R/W	Month Timer alarm setting value digits in BCD format

- Offset 0x34: RTC_YEAR_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
evt_year_irq_repeat	[9]	0	R/W	Note: This field is relevant only when evt_sec_irq_repeat, evt_min_irq_repeat ,evt_hr_irq_repeat, evt_day_irq_repeat and evt_month_irq_repeat is not set

				if set to 1, event irq trigger at every second alarm match, minute alarm match, hour alarm match, day alarm match, and month alarm match
year_irq_mode	[8]	0	R/W	year irq mode setting value Irq mode=0, irq at alarm year Irq mode=1, irq at every year
Alarm_YearVal_00_90	[7:4]	0x0	R/W	Year Timer alarm setting value tens digits in BCD format
Alarm_YearVal_0_9	[3:0]	0x0	R/W	Year Timer alarm setting value digits in BCD format

- Offset 0x38: RTC_INT_EN

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
int_en	[7:0]	0x0	R/W	RTC timer interrupt enable int_en[0]=sec alarm irq enable int_en[1]=min alarm irq enable int_en[2]=hour alarm irq enable int_en[3]=day alarm irq enable int_en[4]=month alarm irq enable int_en[5]=year alarm irq enable int_en[6]=event alarm irq enable int_en[7]=ms alarm irq enable

- Offset 0x3C: RTC_INT_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
int_st	[7:0]	0x0	R	RTC timer interrupt status int_st[0]=sec alarm irq status int_st[1]=min alarm irq status int_st[2]=hour alarm irq status int_st[3]=day alarm irq status int_st[4]=month alarm irq status int_st[5]=year alarm irq status int_st[6]=event alarm irq status int_st[7]=ms alarm irq status

- Offset 0x40: RTC_INT_CLEAR

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
int_clr	[7:0]	0x0	W1C	RTC timer interrupt clear int_clr[0]=sec alarm irq clear int_clr[1]=min alarm irq clear int_clr[2]=hour alarm irq clear int_clr[3]=day alarm irq clear int_clr[4]=month alarm irq clear int_clr[5]=year alarm irq clear int_clr[6]=event alarm irq clear int_clr[7]=ms alarm irq clear

- Offset 0x44: RTC_LOAD

Signal	Bits	Default	R/W	Description
reserved	[31:3]			
Ld_DivVal	[2]	0	R/W	Write 1 to load time value to 32K clk domain. If read=1, load in progress, if read=0, load done.
Ld_Alarm_timeVal	[1]	0	R/W	Write 1 to load alarm time value to 32K clk domain. If read=1, load in progress, if read=0, load done.
Ld_timeVal	[0]	0	R/W	Write 1 to load time value to 32K clk domain. If read=1, load in progress, if read=0, load done.

- Offset 0x48: RTC_MS

Signal	Bits	Default	R/W	Description
reserved	[31:12]			
msVal_000_900	[11:8]	0x0	R/W	The RTC Timer millisecond setting value hundreds digits in BCD format
msVal_00_90	[7:4]	0x0	R/W	The RTC Timer millisecond setting value tens digits in BCD format
msVal_0_9	[3:0]	0x0	R/W	The RTC Timer millisecond setting value digits in BCD format

- Offset 0x4C: RTC_MS_ALARM

Signal	Bits	Default	R/W	Description
reserved	[31:14]			
evt_ms_irq_repeat	[13]	0	R/W	if set to 1, event irq trigger at every millisecond
ms_irq_mode	[12]	0	R/W	sec irq mode setting value Irq mode=0, irq at alarm millisecond Irq mode=1, irq at every millisecond
Alarm_msVal_000_900	[11:8]	0x0	R/W	ms Timer alarm setting value hundreds digits in BCD format
Alarm_msVal_00_90	[7:4]	0x0	R/W	ms Timer alarm setting value tens digits in BCD format
Alarm_msVal_0_9	[3:0]	0x0	R/W	ms Timer alarm setting value digits in BCD format

17. GPIO Control

The GPIO control module provides the manual control interface to CZ20 GPIOs. Each GPIO can be programmed to either input or output mode. For the output mode, Host CPU can drive the output to high or low as needed. For the input mode, an interrupt can be set to the rising edge, falling edge, or both edges. There are up to 32 GPIOs and each of them can be controlled independently. All available GPIOs can be configured to be the wakeup source in low-power modes.

17.1. Block Diagram

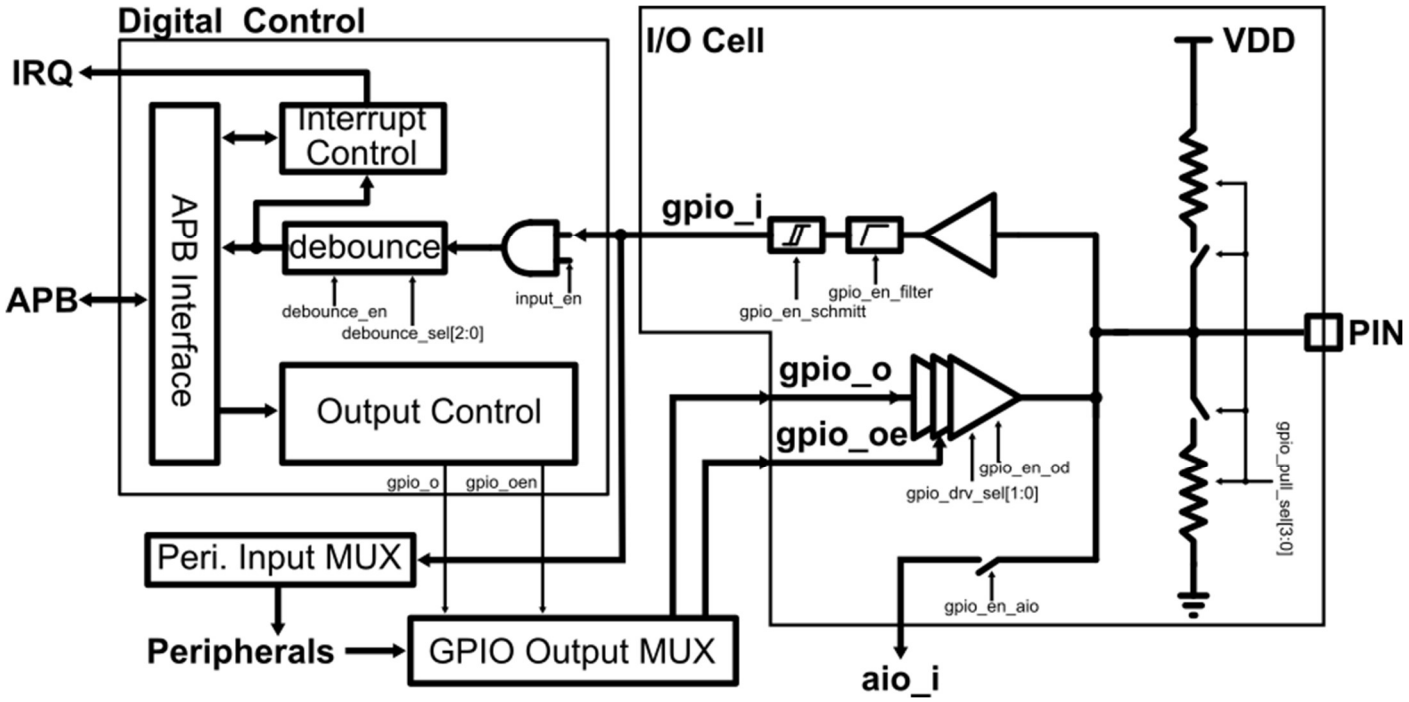


Figure 17-1 Block Diagram of GPIO

17.2. Functional Description

Each GPIO is divided into the Digital Control and the I/O Cell as illustrated in Figure 17-1. Its feature is summarized below.

- I/O pull control
- Input de-glitch filter

- Input Schmitt trigger
- Analog input for AUX ADC
- Output driving control
- Output open-drain control
- Input de-bounce
- Interrupt control

The Digital Control part provides the interface that can control the I/O direction of the GPIO. The Host CPU can drive the output high or low and read the state of the input. A digital de-bounce can be enabled to filter out the unwanted input bouncing. An interrupt can be set at the rising, falling or both edges of the GPIO input.

There is an input enable control bit, “input_en”, that can use to disable the GPIO input. The input enable is disabled after the power-on reset to prevent from unwanted noise and be aware to enable them in the input mode. If “input_en” is enabled in the output mode, the output state can be loop back to the input through the I/O Cell as illustrated in Figure 17-1.

The I/O Cell also has some configurable functions related to the analog circuits. The pull up and down resistors can be enabled and adjusted. All GPIOs are set to have the 100 kOhm pull-up resistor after the power-on reset. The output stage has several driving capabilities and can be configured to be open-drained. In the input path, there is an analog de-glitch filter and a Schmitter trigger that can be enabled individually. The analog de-glitch filter has the capability to filter out the pulse with the width less than 20 ns.

Some I/O cells have the analog input (AIO) function, aio_i, from the PAD PIN directly. Those analog inputs are routed to AUX ADC and AUX Comparator and can be configured as the inputs of them. GPIOs having AIO capabilities are summarized in Table 17-1.

Table 17-1 Mapping of AIO and GPIO

AIO	GPIO
AIO0	GPIO21
AIO1	GPIO22
AIO2	GPIO23
AIO3	N/A
AIO4	GPIO28
AIO5	GPIO29
AIO6	GPIO30
AIO7	GPIO31

17.3. GPIO Multi-function MUXs

The GPIO multi-function MUXs provide the great flexibility for configuring peripheral interface through GPIOs. For a peripheral, its output pins can be routed to any available GPIO while its input pins can also be selected from any available GPIO. The multi-function output of a GPIO is controlled by the register, gpio_omux_xx (xx=00~31), and is summarized in Table 17-2.

Table 17-2 GPIO Multi-function Output MUX

gpio_omux_xx	GPIO output function	gpio_omux_xx	GPIO output function
0x00	GPIO output	**0x20**	QSPI 0 CSN[0] (Master mode)
0x01	UART 0 TX	**0x21**	QSPI 0 CSN[1] (Master mode)
0x02	UART 1 TX	**0x22**	QSPI 0 CSN[2] (Master mode)
0x03	UART 1 RTSN	**0x23**	QSPI 0 CSN[0] (Master mode)
0x04	UART 2 TX	**0x24**	I²S BCK
0x05	UART 2 RTSN	**0x25**	I²S WCK
0x06	PWM0	**0x26**	I²S SDO
0x07	PWM1	**0x27**	I²S MCLK
0x08	PWM2	**0x28**	Reserved
0x09	PWM3	**0x29**	Reserved
0x0A	PWM4	**0x2A**	Reserved
0x0B	IRM	**0x2B**	Reserved
0x0C	I²C Master 0 SCL	**0x2C**	Reserved
0x0D	I²C Master 0 SDA	**0x2D**	Reserved
0x0E	I²C Master 1 SCL	**0x2E**	Reserved
0x0F	I²C Master 1 SDA	**0x2F**	**SWDIO (For GPIO11 only)**
0x10	I²C Slave SCL	**0x30**	Reserved
0x11	I²C Slave SDA	**0x31**	Reserved
0x12	QSPI 0 SCLK (Master mode)	**0x32**	Reserved
0x13	QSPI 0 SDATA[0]	**0x33**	Reserved
0x14	QSPI 0 SDATA[1]	**0x34**	Reserved
0x15	QSPI 0 SDATA[2]	**0x35**	Reserved
0x16	QSPI 0 SDATA[3]	**0x36**	Reserved
0x17	QSPI 0 CSN[0] (Master mode)	**0x37**	Reserved
0x18	QSPI 0 CSN[1] (Master mode)	**0x38**	Reserved
0x19	QSPI 0 CSN[2] (Master mode)	**0x39**	Reserved
0x1A	QSPI 0 CSN[0] (Master mode)	**0x3A**	Reserved
0x1B	QSPI 1 SCLK (Master mode)	**0x3B**	Reserved
0x1C	QSPI 1 SDATA[0]	**0x3C**	Reserved
0x1D	QSPI 1 SDATA[1]	**0x3D**	Reserved
0x1E	QSPI 1 SDATA[2]	**0x3E**	Reserved
0x1F	QSPI 1 SDATA[3]	**0x3F**	Reserved

The gpio_omux_xx only controls the output path. For input paths of a peripheral, additional input MUXs shall also be set. The input path MUXs include an input enable registers and an input GPIO selection registers. Please refer to Section 17.4.2 for those registers in detail. Some examples are given in Table 17-3 and Table 17-4 to help understanding how to set.

Table 17-3 Example of GPIO Multi-function Settings for UART 2 with Flowcontrol

Pin Name	I/O	GPIO	Register Setting
UART2_TX	O	GPIO21	gpio_omux_21 = 0x04
UART2_RX	I	GPIO22	en_uart2_rx = 0x1 gsel_uart2_rx = 0x16
UART2_RTSN	O	GPIO23	gpio_omux_23 = 0x05
UART2_CTSN	I	GPIO24	en_uart2_ctsn = 0x1 gsel_uart2_ctsn = 0x18

Table 17-4 Example of GPIO Multi-function Settings for I²C Master 0

Pin Name	I/O	GPIO	Register Setting
I2CM0_SCL	I/O	GPIO04	gpio_omux_04 = 0x0C en_i2cm0_scl_i = 0x1 gsel_ i2cm0_scl_i = 0x04
I2CM0_SDA	I/O	GPIO07	gpio_omux_07 = 0x0D en_i2cm0_sda_i = 0x1 gsel_ i2cm0_sda_i = 0x07

For peripheral interface pins with the bi-directional capabilities like the SCL and SDA of the I2C interface, both registers of output MUXs and input MUXs shall be set separately as shown in Table 17-4.

The GPIO output MUXs are located in the Peripheral 3 power domain. In most applications, Peripheral 3 is powered down during low-power modes such as Sleep or Deep Sleep to minimize standby current. As a result, the GPIO output MUXs are disabled in these modes, and control of the GPIO outputs is handed over to the GPIO Control module. Therefore, it is necessary to configure the corresponding GPIOs as outputs, with their output states maintained by the GPIO Control during low-power operation.

17.4. Registers

Registers of GPIO are also divided into two parts. Registers related to the Digital Control part are located at the memory space of GPIO Control while those related to the I/O Cells are located at the memory space of System Control.

17.4.1. Registers of GPIO Control

Registers below are resident in the GPIO Control space whose base address is 0x50001000 for the secure partition or 0x40001000 for the non-secure partition.

- Offset 0x00: GPIO_SET_STATE

Signal	Bits	Default	R/W	Description
set_out_state	[31:0]	0x0	W1C	Write 1 to set the corresponding output signal.
state_input	[31:0]	0x0	R	The states of GPIO input.

- Offset 0x04: GPIO_CLR_STATE

Signal	Bits	Default	R/W	Description
clr_out_state	[31:0]	0x0	W1C	Write 1 to clear the corresponding output signal.
state_intr_status	[31:0]	0x0	R	The states of GPIO interrupt status

- Offset 0x08: GPIO_SET_OUT

Signal	Bits	Default	R/W	Description
set_output_en	[31:0]	0x0	W1C	Write 1 to set the corresponding GPIO as an output.
state_output_en	[31:0]	0x0	R	The states of GPIO output enable.

- Offset 0x0C: GPIO_SET_IN

Signal	Bits	Default	R/W	Description
clr_output_en	[31:0]	0x0	W1C	Write 1 to set the corresponding GPIO as an input.
state_output	[31:0]	0x0	R	The states of GPIO output.

- Offset 0x10: GPIO_INTR_EN

Signal	Bits	Default	R/W	Description
set_intr_en	[31:0]	0x0	W1C	Write 1 to enable the corresponding interrupt.
state_intr_en	[31:0]	0x0	R	The states of GPIO interrupt enable.

- Offset 0x14: GPIO_INTR_DIS

Signal	Bits	Default	R/W	Description
clr_intr_en	[31:0]	0x0	W1C	Write 1 to disable the corresponding interrupt.

- Offset 0x18: GPIO_INTR_EDGE

Signal	Bits	Default	R/W	Description
set_intr_edge	[31:0]	0x0	W1C	Write 1 to set the corresponding interrupt as the

				edge trigger.
state_intr_edge	[31:0]	0xFFFFFFFF	R	The states of GPIO interrupt edge trigger enable

- Offset 0x1C: GPIO_INTR_LEVEL

Signal	Bits	Default	R/W	Description
set_intr_level	[31:0]	0x0	W1C	Write 1 to set the corresponding interrupt as the level trigger.
state_intr_level	[31:0]	0x0	R	The states of GPIO interrupt level trigger enable

- Offset 0x20: GPIO_INTR_HIGH

Signal	Bits	Default	R/W	Description
set_edge_high	[31:0]	0x0	W1C	Write 1 to set interrupts to active high or rising edge.
state_edge_hl	[31:0]	0x0	R	The states of GPIO interrupt high/low level or rising/falling edge

- Offset 0x24: GPIO_INTR_LOW

Signal	Bits	Default	R/W	Description
set_edge_low	[31:0]	0x0	W1C	Write 1 to set interrupts to active low or falling edge.

- Offset 0x28: GPIO_SET_INTR_ANYEDGE

Signal	Bits	Default	R/W	Description
set_intr_anyedge	[31:0]	0x0	W1C	Write 1 to override interrupt edge selection & instead interrupt on ANY edge.
state_intr_anyedge	[31:0]	0x0	R	The states of GPIO interrupt anyedge trigger enable

- Offset 0x2C: GPIO_CLR_INTR_ANYEDGE

Signal	Bits	Default	R/W	Description
clr_intr_anyedge	[31:0]	0x0	W1C	Write 1 to clear the any edge setting of the corresponding interrupt.

- Offset 0x30: GPIO_CLR_INTR_STATUS

Signal	Bits	Default	R/W	Description
clr_intr_status	[31:0]	0x0	W1C	Write 1 to clear the corresponding interrupt status.

- Offset 0x38: GPIO_SET_INPUT_EN

Signal	Bits	Default	R/W	Description
set_input_en	[31:0]	0x0	W1C	Write 1 to set the corresponding input enable..
state_input_en	[31:0]	0x0	R	The states of GPIO input enable

- Offset 0x3C: GPIO_DIS_INPUT_EN

Signal	Bits	Default	R/W	Description
clr_input_en	[31:0]	0x0	W1C	Each bit that is set to 1 allows/enables the corresponding input signal. Read for current state of all active low output enable signals. All inputs are disabled by default.

- Offset 0x40: GPIO_DEB_EN

Signal	Bits	Default	R/W	Description
set_debounce_en	[31:0]	0x0	W1C	Write 1 to enable the corresponding input debounce.
state_debounce_en	[31:0]	0x0	R	The states of GPIO debounce enable

- Offset 0x44: GPIO_DEB_DIS

Signal	Bits	Default	R/W	Description
clr_debounce_en	[31:0]	0x0	W1C	Write 1 to disable the corresponding input debounce.

- Offset 0x48: GPIO_DEB_SEL

Signal	Bits	Default	R/W	Description
reserved	[31:3]			
debounce_sel	[2:0]	0x0	R/W	De-bounce time selection 000b 32 slow clocks 001b 64 slow clocks 010b 128 slow clocks 011b 256 slow clocks 100b 512 slow clocks 101b : 1024 slow clocks 3'b110 : 2048 slow clocks 3'b111 : 4096 slow clocks

- Offset 0x50: GPIO_SET_DS_EN

Signal	Bits	Default	R/W	Description
set_ds_wakeup_en	[31:0]	0x0	W1C	Write 1 to set the corresponding GPIO for enabling Deep Sleep wakeup. Once enabling Deep Sleep wakeup, System will wakeup in Deep Sleep when the corresponding GPIO matches the polarity set by state_ds_wakeup_pol.
state_ds_wakeup_en	[31:0]	0x0	R	The states of Deep Sleep wakeup enable for each GPIO.

- Offset 0x54: GPIO_DIS_DS_EN

Signal	Bits	Default	R/W	Description
clr_ds_wakeup_en	[31:0]	0x0	W1C	Write 1 to disable Deep Sleep wakeup for the corresponding GPIO.

- Offset 0x58: GPIO_SET_DS_HIGH

Signal	Bits	Default	R/W	Description
set_ds_wakeup_high	[31:0]	0x0	W1C	Write 1 to set the corresponding GPIO wakeup polarity of high in Deep Sleep.
state_ds_wakeup_pol	[31:0]	0x0	R	The states of Deep Sleep wakeup polarity for each GPIO. 0b level-low wakeup 1b level-high wakeup

- Offset 0x5C: GPIO_SET_DS_LOW

Signal	Bits	Default	R/W	Description
set_ds_wakeup_low	[31:0]	0x0	W1C	Write 1 to set the corresponding GPIO wakeup polarity of low in Deep Sleep.

17.4.2. Registers of I/O Cells and Multi-function MUXs

Registers below are resident in the System Control space whose base address is 0x50000000 for the secure partition or 0x40000000 for the non-secure partition.

- Offset 0x20: GPIO_PULL0

Signal	Bits	Default	R/W	Description
gpio_pull_sel7	[31:28]	0x6	R/W	GPIO7 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel6	[27:24]	0x6	R/W	GPIO6 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel5	[23:20]	0x6	R/W	GPIO5 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull

				0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel4	[19:16]	0x6	R/W	GPIO4 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel3	[15:12]	0x6	R/W	GPIO3 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel2	[11:8]	0x6	R/W	GPIO2 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel1	[7:4]	0x6	R/W	GPIO1 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel0	[3:0]	0x6	R/W	GPIO0 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up

- Offset 0x24: GPIO_PULL1

Signal	Bits	Default	R/W	Description
gpio_pull_sel15	[31:28]	0x6	R/W	GPIO15 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel14	[27:24]	0x6	R/W	GPIO14 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel13	[23:20]	0x6	R/W	GPIO13 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel12	[19:16]	0x6	R/W	GPIO12 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel11	[15:12]	0x6	R/W	GPIO11 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel10	[11:8]	0x6	R/W	GPIO10 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down

				0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel9	[7:4]	0x6	R/W	GPIO9 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel8	[3:0]	0x6	R/W	GPIO8 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up

- Offset 0x28: GPIO_PULL2

Signal	Bits	Default	R/W	Description
gpio_pull_sel23	[31:28]	0x6	R/W	GPIO23 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel22	[27:24]	0x6	R/W	GPIO22 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel21	[23:20]	0x6	R/W	GPIO21 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up

				0x6 100K pull up 0x7 1M pull up
gpio_pull_sel20	[19:16]	0x6	R/W	GPIO20 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel19	[15:12]	0x6	R/W	GPIO19 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel18	[11:8]	0x6	R/W	GPIO18 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel17	[7:4]	0x6	R/W	GPIO17 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel16	[3:0]	0x6	R/W	GPIO16 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up

- Offset 0x2C: GPIO_PULL3

Signal	Bits	Default	R/W	Description

gpio_pull_sel31	[31:28]	0x6	R/W	GPIO31 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel30	[27:24]	0x6	R/W	GPIO30 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel29	[23:20]	0x6	R/W	GPIO29 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel28	[19:16]	0x6	R/W	GPIO28 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel27	[15:12]	0x6	R/W	GPIO27 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel26	[11:8]	0x6	R/W	GPIO26 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull

				0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel25	[7:4]	0x6	R/W	GPIO25 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up
gpio_pull_sel24	[3:0]	0x6	R/W	GPIO24 pull select 0x0 no pull 0x1 10K pull down 0x2 100K pull down 0x3 1M pull down 0x4 no pull 0x5 10k pull up 0x6 100K pull up 0x7 1M pull up

- Offset 0x30: GPIO_DRV0

Signal	Bits	Default	R/W	Description
gpio_drv_sel15	[31:30]	11b	R/W	GPIO15 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel14	[29:28]	11b	R/W	GPIO14 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel13	[27:26]	11b	R/W	GPIO13 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel12	[25:24]	11b	R/W	GPIO12 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel11	[23:22]	11b	R/W	GPIO11 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel10	[21:20]	11b	R/W	GPIO10 driving select

				00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel9	[19:18]	11b	R/W	GPIO9 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel8	[17:16]	11b	R/W	GPIO8 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel7	[15:14]	11b	R/W	GPIO7 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel6	[13:12]	11b	R/W	GPIO6 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel5	[11:10]	11b	R/W	GPIO5 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel4	[9:8]	11b	R/W	GPIO4 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel3	[7:6]	11b	R/W	GPIO3 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel2	[5:4]	11b	R/W	GPIO2 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel1	[3:2]	11b	R/W	GPIO1 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel0	[1:0]	11b	R/W	GPIO0 driving select

				00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)

- Offset 0x34: GPIO_DRV1

Signal	Bits	Default	R/W	Description
gpio_drv_sel31	[31:30]	11b	R/W	GPIO31 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel30	[29:28]	11b	R/W	GPIO30 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel29	[27:26]	11b	R/W	GPIO29 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel28	[25:24]	11b	R/W	GPIO28 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel27	[23:22]	11b	R/W	GPIO27 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel26	[21:20]	11b	R/W	GPIO26 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel25	[19:18]	11b	R/W	GPIO25 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel24	[17:16]	11b	R/W	GPIO24 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel23	[15:14]	11b	R/W	GPIO23 driving select 00b driving strength 0 (weakest) 01b driving strength 1

				10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel22	[13:12]	11b	R/W	GPIO22 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel21	[11:10]	11b	R/W	GPIO21 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel20	[9:8]	11b	R/W	GPIO20 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel19	[7:6]	11b	R/W	GPIO19 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel18	[5:4]	11b	R/W	GPIO18 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel17	[3:2]	11b	R/W	GPIO17 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)
gpio_drv_sel16	[1:0]	11b	R/W	GPIO16 driving select 00b driving strength 0 (weakest) 01b driving strength 1 10b driving strength 2 11b driving strength 3 (strongest)

Driving Strength	VAB & VCCIO = 1.8V	VAB & VCCIO = 3.3V
00b driving strength 0	1.8 mA	6 - 6.5 mA
01b driving strength 1	4.5 mA	14.5 – 16 mA
10b driving strength 2	6 mA	20 – 22 mA
11b driving strength 3	8.5 mA	28.5 – 30 mA

- Offset 0x38: GPIO_EN_OD

Signal	Bits	Default	R/W	Description
gpio_en_od	[31:0]	0x0	R/W	Enable open-drained output of the corresponding GPIO and bit 0 is for GPIO0. 0b Disable 1b Enable

- Offset 0x3C: GPIO_EN_SCHMITT

Signal	Bits	Default	R/W	Description
gpio_en_schmitt	[31:0]	0x0	R/W	Enable Schmitt Trigger of the corresponding GPIO and bit 0 is for GPIO0. 0b Disable 1b Enable

- Offset 0x40: GPIO_EN_FILTER

Signal	Bits	Default	R/W	Description
gpio_en_filter	[31:0]	0x0	R/W	Enable the analog de-glitch filter of the corresponding GPIO and bit 0 is for GPIO0. 0b Disable 1b Enable

- Offset 0x44: GPIO_EN_AIO

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
gpio_en_aio	[7:0]	0x0	R/W	Enable the corresponding AIOs and bit 0 is for AIO0. 0b Disable 1b Enable

- Offset 0x80: GPIO_OMUX0

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_03	[29:24]	0x00	R/W	GPIO03 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_02	[21:16]	0x00	R/W	GPIO02 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_01	[13:8]	0x00	R/W	GPIO01 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_00	[5:0]	0x00	R/W	GPIO00 output MUX. Please refer to Table 17-2.

- Offset 0x84: GPIO_OMUX1

Signal	Bits	Default	R/W	Description
reserved	[31:30]			

gpio_omux_07	[29:24]	0x00	R/W	GPIO07 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_06	[21:16]	0x00	R/W	GPIO06 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_05	[13:8]	0x00	R/W	GPIO05 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_04	[5:0]	0x00	R/W	GPIO04 output MUX. Please refer to Table 17-2.

- Offset 0x88: GPIO_OMUX2

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_11	[29:24]	0x2F	R/W	GPIO11 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_10	[21:16]	0x00	R/W	GPIO10 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_09	[13:8]	0x00	R/W	GPIO09 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_08	[5:0]	0x00	R/W	GPIO08 output MUX. Please refer to Table 17-2.

- Offset 0x8C: GPIO_OMUX3

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_15	[29:24]	0x00	R/W	GPIO15 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_14	[21:16]	0x00	R/W	GPIO14 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_13	[13:8]	0x00	R/W	GPIO13 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_12	[5:0]	0x00	R/W	GPIO12 output MUX. Please refer to Table 17-2.

- Offset 0x90: GPIO_OMUX4

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_19	[29:24]	0x00	R/W	GPIO19 output MUX. Please refer to Table 17-2.
reserved	[23:22]			

gpio_omux_18	[21:16]	0x00	R/W	GPIO18 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_17	[13:8]	0x01	R/W	GPIO17 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_16	[5:0]	0x00	R/W	GPIO16 output MUX. Please refer to Table 17-2.

- Offset 0x94: GPIO_OMUX5

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_23	[29:24]	0x00	R/W	GPIO23 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_22	[21:16]	0x00	R/W	GPIO22 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_21	[13:8]	0x00	R/W	GPIO21 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_20	[5:0]	0x00	R/W	GPIO20 output MUX. Please refer to Table 17-2.

- Offset 0x98: GPIO_OMUX6

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_27	[29:24]	0x00	R/W	GPIO27 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_26	[21:16]	0x00	R/W	GPIO26 output MUX. Please refer to Table 17-2.
reserved	[15:14]			
gpio_omux_25	[13:8]	0x00	R/W	GPIO25 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_24	[5:0]	0x00	R/W	GPIO24 output MUX. Please refer to Table 17-2.

- Offset 0x9C: GPIO_OMUX7

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
gpio_omux_31	[29:24]	0x00	R/W	GPIO31 output MUX. Please refer to Table 17-2.
reserved	[23:22]			
gpio_omux_30	[21:16]	0x00	R/W	GPIO30 output MUX. Please refer to Table 17-2.
reserved	[15:14]			

gpio_omux_29	[13:8]	0x00	R/W	GPIO29 output MUX. Please refer to Table 17-2.
reserved	[7:6]			
gpio_omux_28	[5:0]	0x00	R/W	GPIO28 output MUX. Please refer to Table 17-2.

- Offset 0xA0: GPIO_IMUX0

Signal	Bits	Default	R/W	Description
en_uart2_ctsn	[31]	0x0		Enable UART 2 CTSN input.
reserved	[30:29]			
gsel_uart2_ctsn	[28:24]	0x00	R/W	Select the related GPIOxx input for UART 2 CTSN.
en_uart2_rx	[23]	0x0		Enable UART2_CTSN input.
reserved	[22:21]			
gsel_uart2_rx	[20:16]	0x00	R/W	Select the related GPIOxx input for UART 2 RX
en_uart1_ctsn	[15]	0x0		Enable UART 2 RX input.
reserved	[14:13]			
gsel_uart1_ctsn	[12:8]	0x00	R/W	Select the related GPIOxx input for UART 1 CTSN.
en_uart1_rx	[7]	0x0		Enable UART 1 RX input.
reserved	[6:5]			
gsel_uart1_rx	[4:0]	0x00	R/W	Select the related GPIOxx input for UART 1 RX.

- Offset 0xA4: GPIO_IMUX1

Signal	Bits	Default	R/W	Description
en_i2cs_sda_i	[31]	0x0		Enable I^2C Slave SDA input.
reserved	[30:29]			
gsel_i2cs_sda_i	[28:24]	0x00	R/W	Select the related GPIOxx input for I^2C Slave SDA.
en_i2cs_scl_i	[23]	0x0		Enable I^2C Slave SCL input.
reserved	[22:21]			
gsel_i2cs_scl_i	[20:16]	0x00	R/W	Select the related GPIOxx input for I^2C Slave SCL
en_i2s_sdi	[15]	0x0		Enable I2S SDI input.
reserved	[14:13]			
gsel_i2s_sdi	[12:8]	0x00	R/W	Select the related GPIOxx input for I2S SDI.
en_uart0_rx	[7]	0x1		Enable UART 0 RX input.
reserved	[6:5]			
gsel_uart0_rx	[4:0]	0x10	R/W	Select the related GPIOxx input for UART 0 RX.

- Offset 0xA8: GPIO_IMUX2

Signal	Bits	Default	R/W	Description
en_i2cm1_sda_i	[31]	0x0		Enable I^2C Master 1 SDA input.
reserved	[30:29]			
gsel_i2m1_sda_i	[28:24]	0x00	R/W	Select the related GPIOxx input for I^2C Master 1 SDA.
en_i2cm1_scl_i	[23]	0x0		Enable I^2C Master 1 SCL input.
reserved	[22:21]			

gsel_i2cm1_scl_i	[20:16]	0x00	R/W	Select the related GPIOxx input for I²C Master 1 SCL
en_i2cm0_sda_i	[15]	0x0		Enable I²C Master 0 SDA input.
reserved	[14:13]			
gsel_i2cm0_sda_i	[12:8]	0x00	R/W	Select the related GPIOxx input for I²C Master 0 SDA.
en_i2cm0_scl_i	[7]	0x0		Enable I²C Master 0 SCL input.
Reserved	[6:5]			
gsel_i2cm0_scl_i	[4:0]	0x00	R/W	Select the related GPIOxx input for I²C Master 0 SCL.

- Offset 0xAC: GPIO_IMUX3

Empty in this offset.

- Offset 0xB0: GPIO_IMUX4

Signal	Bits	Default	R/W	Description
en_spi0_sdata1_i	[31]	0x0		Enable QSPI 0 SDATA[1] input.
Reserved	[30:29]			
gsel_spi0_sdata1_i	[28:24]	0x00	R/W	Select the related GPIOxx input for QSPI 0 SDATA[1].
en_spi0_sdata0_i	[23]	0x0		Enable QSPI 0 SDATA[0] input.
Reserved	[22:21]			
gsel_spi0_sdata0_i	[20:16]	0x00	R/W	Select the related GPIOxx input for QSPI 0 SDATA[0].
en_spi0_sclk_i	[15]	0x0		Enable QSPI 0 SCLK iput. For Slave mode only.
Reserved	[14:13]			
gsel_spi0_sclk_i	[12:8]	0x00	R/W	Select the related GPIOxx input for QSPI 0 SCLK. For Slave mode only.
en_spi0_csn_i	[7]	0x0		Enable QSPI 0 CSN input. For Slave mode only.
reserved	[6:5]			
gsel_spi0_csn_i	[4:0]	0x00	R/W	Select the related GPIOxx input for QSPI 0 CSN. For Slave mode only.

- Offset 0xB4: GPIO_IMUX5

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
en_spi0_sdata3_i	[15]	0x0		Enable QSPI 0 SDATA[3] input.
reserved	[14:13]			
gsel_spi0_sdatar_i	[12:8]	0x00	R/W	Select the related GPIOxx input for QSPI 0 SDATA[3].
en_spi0_sdata2_i	[7]	0x0		Enable QSPI 0 SDATA[2] input.
reserved	[6:5]			
gsel_spi0_sdata2_i	[4:0]	0x00	R/W	Select the related GPIOxx input for QSPI 0 SDATA[2].

- Offset 0xB8: GPIO_IMUX6

Signal	Bits	Default	R/W	Description

en_spi1_sdata1_i	[31]	0x0		Enable QSPI 1 SDATA[1] input.
reserved	[30:29]			
gsel_spi1_sdata1_i	[28:24]	0x00	R/W	Select the related GPIOxx input for QSPI 1 SDATA[1].
en_spi1_sdata0_i	[23]	0x0		Enable QSPI 1 SDATA[0] input.
reserved	[22:21]			
gsel_spi1_sdata0_i	[20:16]	0x00	R/W	Select the related GPIOxx input for QSPI 1 SDATA[0].
en_spi1_sclk_i	[15]	0x0		Enable QSPI 1 SCLK iput. For Slave mode only.
reserved	[14:13]			
gsel_spi0_sclk_i	[12:8]	0x00	R/W	Select the related GPIOxx input for QSPI 0 SCLK. For Slave mode only.
en_spi0_csn_i	[7]	0x0		Enable QSPI 0 CSN input. For Slave mode only.
reserved	[6:5]			
gsel_spi0_csn_i	[4:0]	0x00	R/W	Select the related GPIOxx input for QSPI 0 CSN. For Slave mode only.

- Offset 0xBC: GPIO_IMUX7

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
en_spi1_sdata3_i	[15]	0x0		Enable QSPI 1 SDATA[3] input.
reserved	[14:13]			
gsel_spi1_sdatar_i	[12:8]	0x00	R/W	Select the related GPIOxx input for QSPI 1 SDATA[3].
en_spi1_sdata2_i	[7]	0x0		Enable QSPI 1 SDATA[2] input.
reserved	[6:5]			
gsel_spi1_sdata2_i	[4:0]	0x00	R/W	Select the related GPIOxx input for QSPI 1 SDATA[2].

18. UART

The UART module support Universal Asynchronous Receiver/Transmitter protocol. The data of transmit and receive paths is connected to FIFOs. The core converts bytes into serial bits for transmission, and vice versa. The MCU can access data through FIFOs directly or through DMA controller.

18.1. Block Diagram

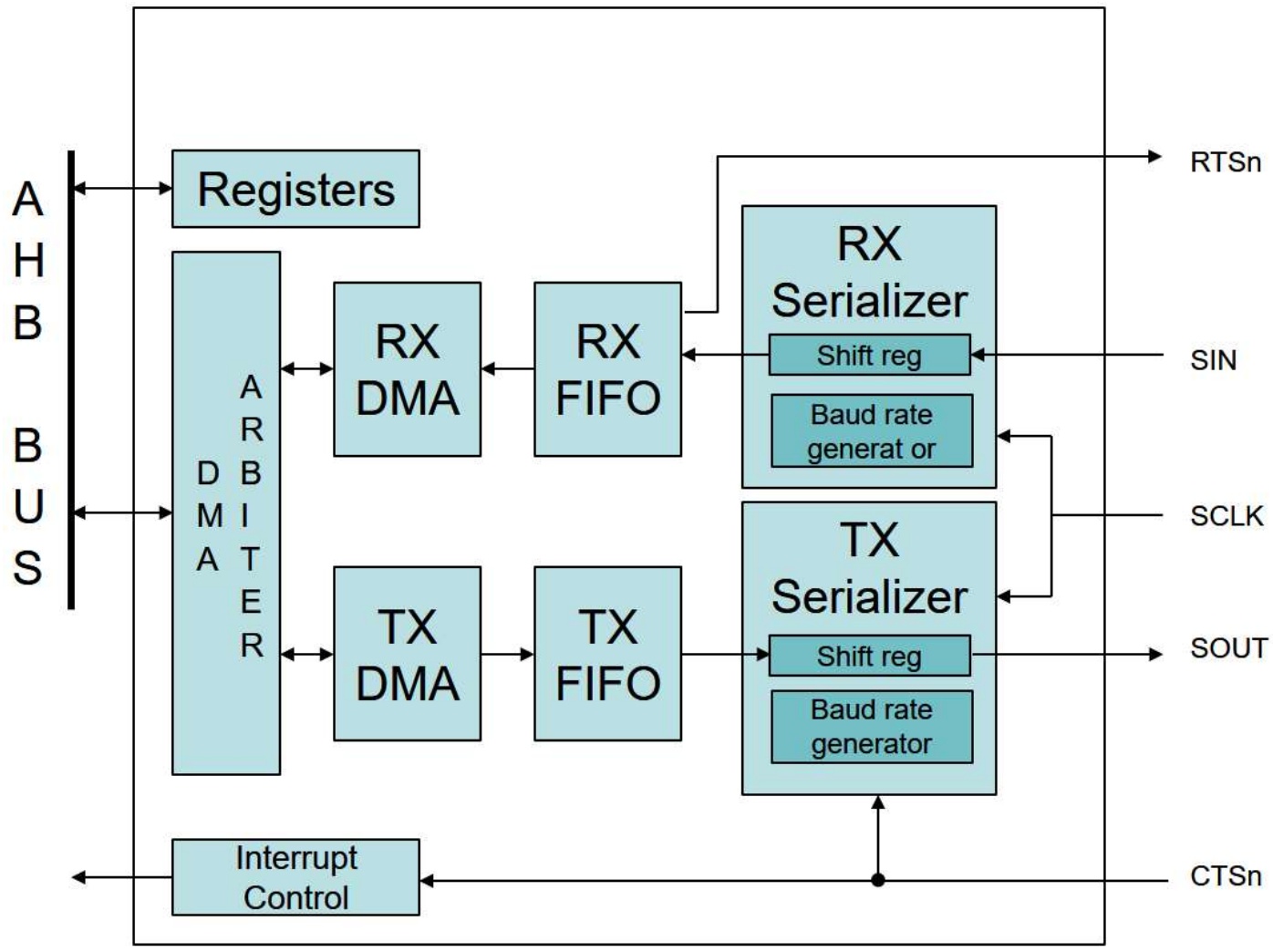


Figure 18-1 Block Diagram of UART

18.2. Features

- Programmable baud rates
- Transmit and Receive FIFOs
- FIFO and DMA modes
- Interrupt control
- Modem control (only flow control signals, RTSn and CTSn)

18.3. Functional Description

18.3.1. Transmitter Operation

The UART Transmitter section is relatively simple. The data is written to FIFO and goes to TX serializer. The state machine controls the shift register and mux to ensure the bits get transmitted in the proper sequence. The baud rate generator controls the pace of the transmission. Parity is calculated if required and inserted into the bit stream.

18.3.2. Receiver Operation

The receiver is more complicated than the transmitter. Since an independent UART is transmitting the signal, timing has to be recovered from the incoming serial bit stream. The falling edge of the start bit triggers a counter that counts into the middle of the bit time programmed into the baud rate generator. Thereafter, strobes are generated which occur in the middle of the bit time and control the timing for the duration of the received word. The timing of the far end transmitter can be substantially different from that of the receiver and the word will be successfully received as long as the cumulative error across the word (including start and stop bits) does not exceed one half of the bit time.

A state machine sequences the receive operation. When all bits are received, parity is calculated and compared to the expected parity. Parity errors, framing errors, break conditions and overrun error are reflected to LSR and interrupt. A framing error is defined as the absence of an expected stop bit. A break condition is declared when too many space symbols occur in a row. That is when the serial input stays in the logic 0 state for a number of bit times greater than the sum of the number of start, stop, parity and data bits.

18.3.3. Baud Rate Generator

The baud rate generator can be configured to generate a wide range of baud rates depending on the system clock frequency, the divisor latch and the fractional divisor latch. The relationship is shown by the following expression:

$$BaudRate = \frac{SystemClockFrequency}{8 \times (DivisorLatch + FracDivisorLatch)}$$

Equation 18-1 Equation of UART Baud Rate Generator for lsm=0

The divisor latch provides the integer part, whose value DivisorLatch can be between 0x0001 and 0xFFFF. The fractional divisor latch provides fractional part, which can support multiples of 1/8 along with divisor latch. The value of FracDivisorLatch is the sum of 1's of each bit in FDL divided by 8.

For example, to configure the UART for a baud rate of 2400 with a system clock frequency of 32 MHz, the DivisorLatch would be 1666 (set DLM to 0x06 and DLL 0x82) and FracDivisorLatch 5/8 (set FDL to 0b00111110).

There is a special mode for UART when system is at sleep mode. Set "lsm" to 1 and FDL to 0x00 and the formula for baud rate is shown as:

$$BaudRate = \frac{SystemClockFrequency(at\ sleep)}{4 \times DivisorLatch}$$

Equation 18-2 Equation of UART Baud Rate Generator for lsm=1

18.3.4. Interrupts

The UART can generate seven types of interrupts: receiver line status, received data available, character timeout, transmitter holding register empty, modem status, TX DMA done and RX DMA done.

Enabling these interrupts is controlled by the setting & clearing bits in the Interrupt Enable Register (IER). A set bit in the IER corresponds to an enabled interrupt and a clear bit in the IER corresponds to a disabled interrupt.

The UART will activate its interrupt signal any time one of the seven interrupt sources is active and enabled by the appropriate bit in the IER.

Table 18-1 Interrupt Sources of UART

IER Bit Number	Corresponding Interrupt Enabled
0	Received Data Available

1	Transmitter Holding Register Empty
2	Receiver Line Status
3	Modem Status
4	Character Timeout
5	RX DMA done
6	TX DMA done

18.3.4.1. Received Data Available

This interrupt occurs when the amount of data in receiver FIFO equals to or exceeds the trigger level (controlled by FCR[7:6]). This interrupt is cleared by write 1 to ISR[0].

18.3.4.2. Transmitter Holding Register (THR) Empty

This interrupt occurs when the transmitter FIFO becomes empty. This interrupt is cleared by write 1 to ISR[1].

Note that the “thre” bit in the LSR will remain set until a character has been loaded into transmitter FIFO even when the interrupt is cleared.

18.3.4.3. Receiver Line Status

This interrupt occurs when any one of the following conditions exist: framing error, parity error, overrun error or break interrupt. This interrupt is cleared by clearing all of the 4 bits of the line status register. Write 1 to clear corresponding bit in line status register.

18.3.4.4. Modem Status

Only clear to send(CTS) and request to send(RTS) are implemented related to modem. The modem status interrupt occurs when delta clear to send (***DCTS***) is set. This interrupt is cleared by write 1 to the Modem Status Register (MSR).

Be aware that in loopback mode the “delta” bit is not controlled by the modem inputs but rather by bit[1] of the Modem Control Register (MCR).

18.3.4.5. Character Timeout

This interrupt occurs when no characters have been written to or read from the FIFO in the last 4 character times. This interrupt is cleared by write 1 to ISR[4].

18.3.4.6. RX DMA done

This interrupt occurs when the desired length of data is written to SRAM from FIFO. This interrupt is cleared by write 1 to ISR[5].

18.3.4.7. TX DMA done

This interrupt occurs when the desired length of data is written to FIFO from SRAM. This interrupt is cleared by write 1 to ISR[6].

Note that when this interrupt occurs, the data in FIFO may still be non-empty.

18.3.5. FIFO Control

The FIFOs are asynchronous to accommodate differences in clock rates between ***SCLK*** & ***PCLK***.

18.3.5.1. Receiver FIFO

The receiver FIFO stores received data until it is read by software or DMA. The depth of the receiver FIFO is 32 bytes. Once the programmed trigger level has been reached within the FIFO, the data ready bit (and received data available interrupt, if enabled) will be set. The FCR[7:6] control the receiver trigger level.

Table 18-2 Trigger Level Setting of UART Receiver FIFO

FCR[7:6]	Receiver FIFO Trigger Level (Bytes)
00	1
01	4
10	8
11	14

The receiver FIFO can be reset by setting FCR[1]. This bit is self-clearing.

18.3.5.2. Transmitter FIFO

The transmitter FIFO is used to store data that is intended to be transmitted. The depth of the transmitter FIFO is 32 bytes. Each byte written to the THR location is loaded into the transmitter FIFO. The transmitter FIFO can be reset by setting FCR[2]. This bit is self-clearing.

18.3.6. Line Control

18.3.6.1. Asynchronous Serial Data Format

The line control register (LCR) specifies the format of the asynchronous data that is transmitted and received by the UART. The number of data bits in each serial character is specified by LCR[1:0].

Table 18-3 UART Character Length Settings

LCR[1:0]	Character Length
00	5 Bits
01	6 Bits
10	7 Bits
11	8 Bits

The number of stop bits in each serial character is specified by LCR[2] along with the current character length selected by LCR[1:0]. Note that this is different from the standard 16550 implementations. When the character length is 5 bits, the number of stop bits should be 1.5. For simplicity, this implementation generates 2 stop bits.

Table 18-4 UART Stop Bits Settings

LCR[2]	Number of Stop Bits
0	1
1	2

Parity bit generation is controlled by LCR[4:3]. Bit 3 enables parity and bit 4 selects even or odd parity.

Table 18-5 UART Parity Settings

LCR[4:3]	Type of Parity
X0	No Parity
01	Odd Parity
11	Even Parity

Even parity sets the parity bit to 1 when there are an even number of ‘1’s in the serial character. The parity bit is set to 0 when there are an odd number of ‘1’s in the serial character. Odd parity sets the parity bit to 1 when there are an odd number of ‘1’s in the serial character and sets the parity bit to 0 when there are an even number of ‘1’s in the serial character.

18.3.6.2. Diagnostic Mechanisms

The line control register (LCR) also provides mechanisms to force break conditions, as well as parity and framing error conditions. Utilizing these mechanisms while in loopback mode allows diagnostic checking of receiver line logic.

LCR[5] is the Stick Parity bit. Stick Parity causes the transmitter to force the parity bit of a transmitted serial character to 1 or 0 regardless of the calculated parity of the transmitted serial character. Stick Parity also causes the receiver to check the parity bit of a received serial character against either 1 or 0 regardless of the calculated parity of the received serial character. It is the Stick Parity bit in conjunction with LCR[4:3] which determine how the parity bit is affected.

The following table summarizes all possible combinations of parity functionality:

Table 18-6 Summary of All UART Parity Settings

LCR[5]	LCR[4:3]	Type of Parity
0	00	parity disabled, no parity bit generated/checked
0	01	odd parity generated/checked
0	10	parity disabled, no parity bit generated/checked
0	11	even parity generated/checked
1	00	parity disabled, no parity bit generated/checked
1	01	stick parity, generated/checked as logic ‘1’
1	10	parity disabled, no parity bit generated/checked
1	11	stick parity, generated/checked as logic ‘0’

LCR[6] is the Set Break bit. Setting this bit forces the ***SOUT*** low (spacing) condition. The ***SOUT*** line will remain in the spacing condition until this bit is cleared.

18.3.7. Line Status

LSR[0] is the Data Ready (***DR***) bit. This bit is set when an incoming character is completely received into receiver FIFO and it is cleared when the characters in the receiver FIFO has been emptied.

LSR[1] is the overrun error indicator. Overrun errors occur when the receiver FIFO is completely full, the receiver shift register contains a complete character, destroying the character in the receiver shift register. The data in the receiver FIFO is not corrupted.

LSR[2] is the parity error indicator. This bit is set when a received character’s parity (as calculated by the receiver) does not match the value of its received parity bit.

LSR[3] is the framing error indicator. This bit is set when the receiver does not detect a valid stop bit

for an incoming character, i.e. the serial input signal was low during the baud time in which a stop bit (high) was expected.

LSR[4] is the break interrupt indicator. This bit is set whenever the serial input is held in the space (logic 0) state for more time than a complete character transfer (i.e. the time it takes to transmit a start bit, the data bits, the parity bit if enabled and any stop bits).

LSR[5] is the Transmitter Holding Register Empty bit. This bit is set when the transmitter FIFO becomes empty and is cleared after at least one character is loaded into the transmitter FIFO.

LSR[6] is the Transmitter Empty Bit. This bit is set when both the transmitter FIFO and the transmitter shift register are empty. This bit is clear when either the transmitter FIFO contains a character or the transmitter shift register has not completed shifting-out a character being transmitted.

18.3.8. Modem Control

RTSn and CTSn are effective when flow control is enabled (controlled by MCR[5]). For receiver, when water level of RX FIFO equals to or exceed desired value (controlled by FCR[9:8]), RTSn is asserted to high and is cleared to low after RX FIFO is empty. For transmitter, when CTSn is high, the serializer is suspended after the on-going byte (if any) being transferred completedly and continues when CTSn is low.

18.3.9. Loopback Operation

The UART provides loopback operation for transmitter, receiver, and modem status diagnostics. The UART operates in loopback mode when MCR[4] is set.

In loopback mode, the serial data input (***SIN***) is disconnected, and the serial data output (***SOUT***) is driven high (marking state). The transmitter is internally connected to the receiver so that any data transmitted is also received within the UART.

The ***CTSn*** modem inputs is disconnected, and the ***RTSn*** output is driven high (inactive state). The ***RTSn*** is internally connected to the ***CTSn***.

The modem status interrupt can still be generated, although in loopback mode it is controlled through MCR.

For example, writing 0x12 to the MCR (asserting ***rts*** bit and maintaining loopback bit) will cause the ***cts*** bit to go active in the MSR. The “delta” edge detection logic then detects the change from low to high of the ***dcts*** bit in the MSR, therefore causing the modem status interrupt.

18.3.10. DMA Mode

For DMA operation, it's enabled by setting FCR[3]. The TX and RX DMA share one AHB interface by time-sharing. The UART asserts read request when TX FIFO is not full, and asserts write request when its RX FIFO is not empty.

18.4. Registers

- Offset 0x00: Receiver Buffer Register (RBR)

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
rbr	[7:0]	x0	RO	Received Data

- Offset 0x00: Transmitter Holding Register (THR)

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
thr	[7:0]	x0	WO	Data To be transmitted

- Offset 0x04: Interrupt Enable Register (IER)

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
intr_en[6]	[6]	b0	R/W	Enable DMA TX Interrupt
intr_en[5]	[5]	b0	R/W	Enable DMA RX Interrupt
intr_en[4]	[4]	b0	R/W	Enable Received Data timeout Interrupt
intr_en[3]	[3]	b0	R/W	Enable MODEM Status Interrupt
intr_en[2]	[2]	b0	R/W	Enable Receiver Line Status Interrupt
intr_en[1]	[1]	b0	R/W	Enable Transmitter Holding Register Interrupt
intr_en[0]	[0]	b0	R/W	Enable Received Data Available Interrupt

- Offset 0x08: FIFO Control Register (FCR)

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
rts_lvl	[9:8]	b0	R/W	RTS Trigger Level
rxd_lvl	[7:6]	b0	R/W	Receiver Trigger Level
reserved	[5:4]			
dma_en	[3]	b0	R/W	DMA Mode
txf_rst	[2]	b0	WO	Transmit FIFO reset (self-clearing)
rxf_rst	[1]	b0	WO	Receive FIFO reset (self-clearing)
reserved	[0]			

- Offset 0x0C: Line Control Register (LCR)

Signal	Bits	Default	R/W	Description

reserved	[31:7]			
sb	[6]	b0	R/W	Set Break
stp	[5]	b0	R/W	Stick Parity
eps	[4]	b0	R/W	Even Parity Select
pen	[3]	b0	R/W	Parity Enable
stb	[2]	b0	R/W	Number of Stop Bits
wls	[1:0]	b0	R/W	Word Length Select

- Offset 0x10: Modem Control Register (MCR)

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
flow_en	[5]	b0	R/W	Enable CTS and RTS function
lpbk	[4]	b0	R/W	Loopback mode select
reserved	[3:2]			
rts	[1]	b0	R/W	Request to Send (RTS)
reserved	[0]			

- Offset 0x14: Line Status Register (LSR)

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
temt	[6]	b0	RO	Transmitter Empty
thre	[5]	b0	RO	Transmitter Holding Register Empty
bi	[4]	b0	R/W1C	Break Interrupt
fe	[3]	b0	R/W1C	Framing Error
pe	[2]	b0	R/W1C	Parity Error
oe	[1]	b0	R/W1C	Overrun Error
dr	[0]	b0	RO	Data Ready (DR)

- Offset 0x18: Modem Status Register (MSR)

Signal	Bits	Default	R/W	Description
reserved	[31:5]			
cts	[4]	b0	RO	Clear To Send (CTS)
reserved	[3:1]			
dcts	[0]	b0	R/W1C	Delta Clear To Send (DCTS)

- Offset 0x1C: Interrupt Identification Register (IIR)

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
dma_tx_intr	[6]	b0	R/W1C	DMA TX Interrupt
dma_rx_intr	[5]	b0	R/W1C	DMA RX Interrupt
rxto_intr	[4]	b0	R/W1C	Received Data timeout Interrupt
msr_intr	[3]	b0	RO	MODEM Status Interrupt
lsr_intr	[2]	b0	RO	Receiver Line Status Interrupt
thre_intr	[1]	b0	R/W1C	Transmitter Holding Register Interrupt

rda_intr	[0]	b0	R/W1C	Received Data Available Interrupt

- Offset 0x20: Divisor Latch Register (DLX)

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
DLM	[15:8]	x0	R/W	Divisor Latch [15:8]
DLL	[7:0]	x0	R/W	Divisor Latch [7:0]

- Offset 0x24: Fractional Divisor Latch (FDL)

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
frac_dlx	[7:0]	x0	R/W	Fractional Divisor Latch [7:0]

- Offset 0x28: Low Speed Mode

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
lsm	[0]	b0	R/W	Low speed mode. When set, oversampling rate is 4x instead of 8x

- Offset 0x2C: UART_WAKE_EN

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
uart_sleep_wake_en	[0]	b0	R/W	Wake up enable when system is in sleep

- Offset 0x30: UART _EN

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
uart_enable	[0]	b0	R/W	UART enable

- Offset 0x34: DmaRcvStartAdr

Signal	Bits	Default	R/W	Description
dma_rx_base	[31:0]	x0	R/W	Start address of RX buffer in SRAM

- Offset 0x38: DmaRcvByteCnt

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_rx_length	[15:0]	x0	R/W	Transfer byte length of receive buffer in SRAM

- Offset 0x3C: DmaXmtStartAdr

Signal	Bits	Default	R/W	Description
dma_tx_base	[31:0]	x0	R/W	Start address of transmit buffer in SRAM

- Offset 0x40: DmaXmtByteCnt

Signal	Bits	Default	R/W	Description

reserved	[31:16]			
dma_tx_length	[15:0]	x0	R/W	Transfer byte length of transmit buffer in SRAM

- Offset 0x44: DmaRcvRemainCnt

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_rx_remain_len	[15:0]	x0	RO	Remain byte length of receive buffer for DmaRcvByteCnt setting.

- Offset 0x48: DmaXmtRemainCnt

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_tx_remain_len	[15:0]	x0	RO	Remain byte length of transmit buffer for DmaXmtByteCnt setting.

- Offset 0x4C: DmaRcvEnable

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
dma_rx_enable	[0]	x0	R/W	DMA enable for RX start. Write 0 for completion or abort

- Offset 0x50: DmaXmtEnable

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
dma_tx_enable	[0]	x0	R/W	DMA enable for TX start. Write 0 for completion or abort

19. QSPI

The Quad Serial Peripheral Interface module either controls a serial data link as a master or reacts to a serial data link as a slave.

The core operates in various data modes (8-bit, 16-bit or 32-bit). The data is serialized and then transmitted, either LSB or MSB first, using the standard 4-wire SPI bus interface or the Quad mode bus.

19.1. Block Diagram

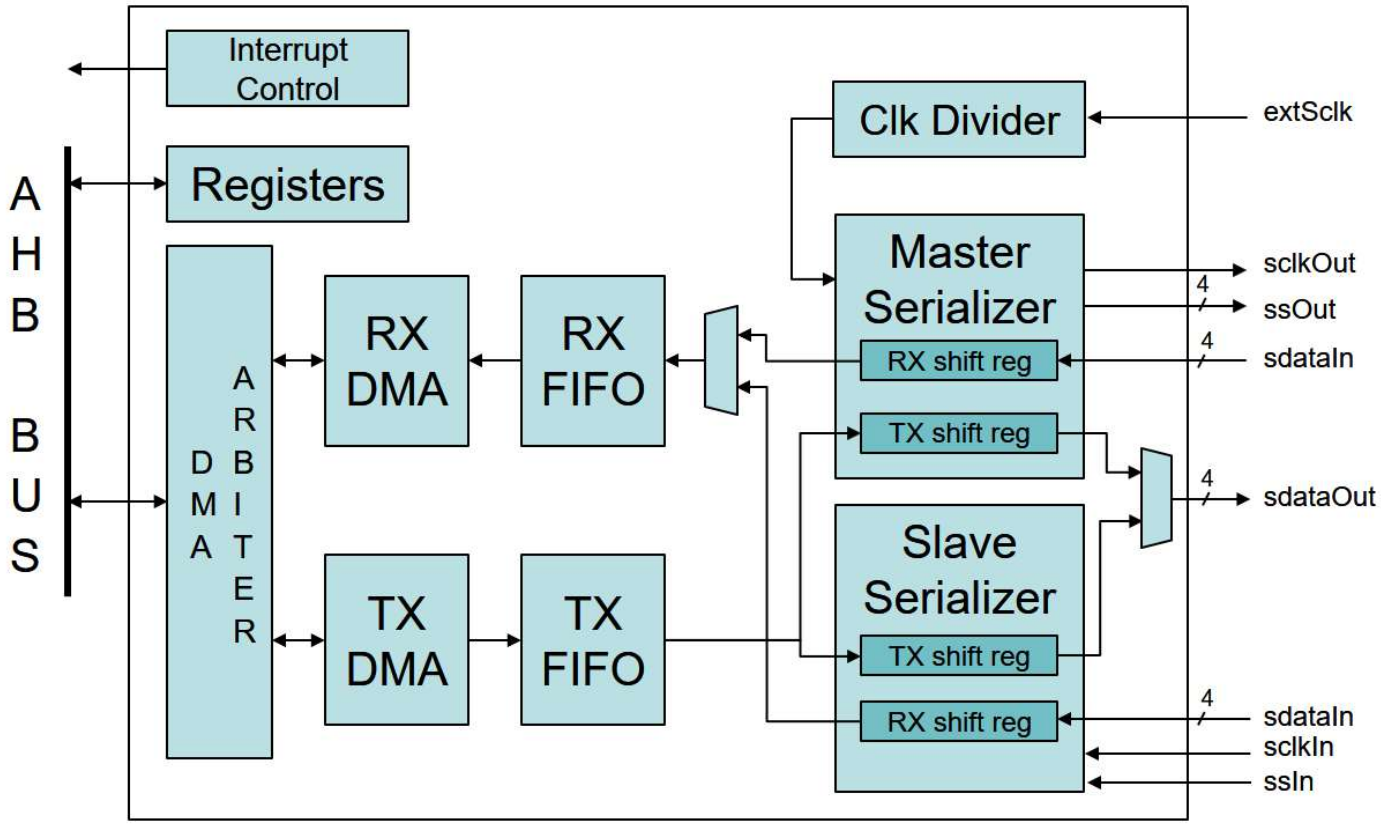


Figure 19-1 Block Diagram of QSPI

19.2. Features

- 8-, 16- or 32-bit serial transmit & receive

- Half duplex operation (Full duplex for standard 4-wire SPI)
- Master or Slave mode
- Quad mode operation
- Dual mode operation
- separate SCLK input for Master Mode
- 32-word Transmit FIFO
- 32-word Receive FIFO
- Interrupt control
- LSB or MSB mode
- Tristate Slave MISO signaling for multiple slaves
- DMA Interface
- Compatible with many industry-standard FLASH devices

19.3. Functional Description

The QSPI can be configured as Master mode or Slave mode. In Master mode, the QSPI core produces the SPI bus clock directly from the External Master SCLK(extSclk), which can be selected between 32M, 16M or RCO1M. The clock is divided into desired speed before use. In Slave mode, the QSPI core receives the SPI bus clock(sclkIn) from another SPI Master. Both Master and Slave Serializer access data through FIFO. Software can access FIFO directly through AHB bus or using DMA interface. The sdataIn and sdataOut are combined as 4 data pins(SDATA). For standard 4-wire SPI, only 2 pins are used and are so-called MOSI(Master Out, Slave In) and MISO(Master In, Slave Out), with opposite transfer direction. For Quad mode, all of them are operating in the same direction.

19.3.1. SPI Data Link

For standard 4-wire SPI, data is transmitted synchronously with MOSI relative to the SCLK generated by the master device. The master also receives data on the MISO signal.

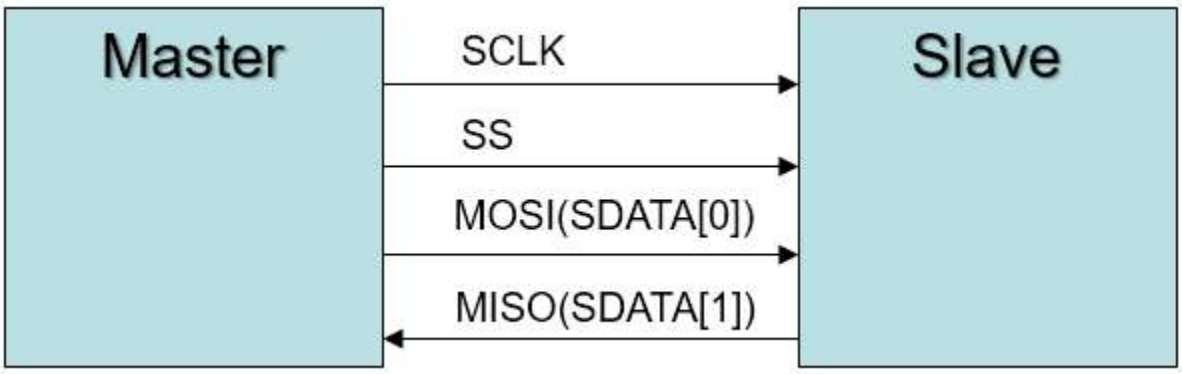


Figure 19-2 Connections of Standard SPI Mode

For Quad mode, the timing is the same as standard 4-wire SPI, except that 4 data pins are used and the data can be half duplex only.

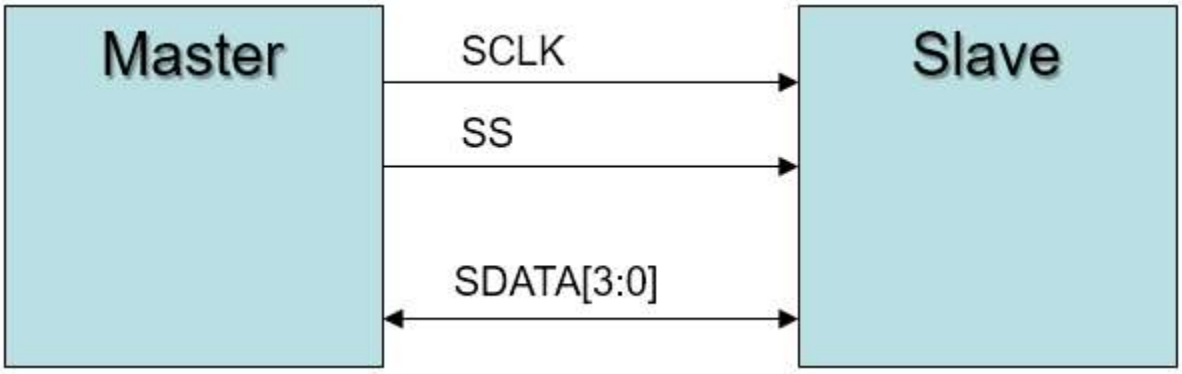


Figure 19-3 Connections of Quad SPI Mode

19.3.2. DMA Operation

QSPI has its native DMA engine. DMA registers (offset 0x60 ~ 0x84) can be set to control TX and RX DMA transactions.

19.4. Registers

- Offset 0x00: Transmit Data (to Tx FIFO)

Signal	Bits	Default	R/W	Description
tx_data	[31:0]	x0	WO	Transmit Data register The data written to this register will be sent to Tx FIFO. The number of bits transmitted is determined by the register “bitsize” (ex: “bitsize” = b011 means only 16 bits out of 32 bits are used). In DMA mode, the register is useless.

- Offset 0x04: Receive Data (from Rx FIFO)

Signal	Bits	Default	R/W	Description
rx_data	[31:0]	x0	RO	Receive Data Register The data received from line bus will be sent to Rx FIFO. Read this register to obtain the data from Rx FIFO. The number of bits stored in FIFO is determined by the register “bitsize” (ex: “bitsize” = b011 means only 16 bits out of 32 bits are used). In DMA mode, this register is useless.

- Offset 0x08: Control Register

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
txf_th	[15:14]	b10	R/W	This register is only useful in DMA mode. When the number of entries in TX FIFO is less than or equal to the corresponding value, the TX DMA continues to access data from SRAM. b00: 4 entries b01: 8 entries b10: 16 entries b11: 24 entries
reserved	[13:11]			
post_delay_en	[10]	b0	R/W	Post delay for Master operation. After data, add delay between ssOut inactive and last SCLK edge. 0=disable delay 1=enable delay.
inter_delay_en	[9]	b0	R/W	This register is only useful in Master mode. It determines whether to add delay between each data (according to “bitsize”). The delay period is determined by register “inter_delay”. 0: disable delay (There is intrinsic 1 SCLK cycle between data) 1: enable delay.
pre_delay_en	[8]	b0	R/W	Pre delay for Master operation. Before data, add delay between ssOut active and first SCLK edge. 0=disable delay 1=enable delay.
reserved	[7]			
ss_cont_en	[6]	b1	R/W	SS Continuous Enable. This register is only useful in Master mode.

				0: data is not continuously transferred 1: data is continuously transferred. ssOut stays active when there is remain data. The behavior of ssOut has some difference according to “ss_cont_mode”. Note that if “inter_delay_en” is 1, only ssOut is continuous but data is not.
msb1st	[5]	b0	R/W	Bit order select for data format. 1: MSB first 0: LSB first
byte_swap	[4]	b1	R/W	Byte swap for data format. It’s only useful for “msb1st” = 1 and “bitsize” = 16/32 bit. When enabled, the byte order is reversed. 0: no swap 1: swap
sdataout0_sel	[3]	b0	R/W	This register is only useful in 4-wire SPI mode (“wire_mode” = b00). It determines the internal connection of data bus SDATA[0] and SDATA[1]. 0: For Master mode use SDATA[0] = sdata_out[0] and sdata_in[0] = SDATA[1] 1: For Slave mode use SDATA[1] = sdata_out[0] and sdata_in[0] = SDATA[0].
cpol	[2]	b1	R/W	Clock polarity of SPI protocol 0: SPI clock rests low 1: SPI clock rests high
cpha	[1]	b0	R/W	Clock phase of SPI protocol 0: first data occurs when SS asserts 1: first data occurs when first SCLK edge occurs after SS asserts
master	[0]	b1	R/W	SPI role mode 0: slave mode 1: master mode

- Offset 0x0C: Auxiliary Control Register

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
ss_cont_mode	[7]	b0	R/W	SS Continuous mode. This register useful when “ss_cont_en” = 1. 0: ssOut become inactive when TX FIFO is empty. 1: ssOut remain active even when TX FIFO is empty.
bitsize	[6:4]	b111	R/W	It determines effective bits of data format of TX/RX FIFO: b001: 8-bit mode (only 8 bits out of 32 bits are used) b011: 16-bit mode (only 16 bits out of 32 bits are used) b111: 32-bit mode Other: reserved
disable_rx	[3]	b0	R/W	If the transfer is only for TX only, set 1 to avoid RX FIFO access. Note that for Quad or Dual wire mode, it’s needed to set 1 for TX and 0 for RX transfer.
disable_tx	[2]	b0	R/W	For Slave mode only. If the transfer is only for RX, set 1 to avoid TX FIFO access.
wire_mode	[1:0]	b00	R/W	SPI wire mode b11: Quad mode (SDATA[3:0] are used) b10: Dual mode (SDATA[1:0] are used) b01: Reserved b00: standard 4-wire SPI (MOSI and MISO are used)

- Offset 0x10: Slave Select Config Register

Note: This register is only useful in Master mode except ss_pol[0].

Signal	Bits	Default	R/W	Description
reserved	[31:28]			
ss_man_en[3]	[27]	b0	R/W	0: ssOut[3] is controlled by HW core 1: ssOut[3] is controlled by ss_man[3]
ss_man_en[2]	[26]	b0	R/W	0: ssOut[2] is controlled by HW core 1: ssOut[2] is controlled by ss_man[2]
ss_man_en[1]	[25]	b0	R/W	0: ssOut[1] is controlled by HW core 1: ssOut[1] is controlled by ss_man[1]
ss_man_en[0]	[24]	b0	R/W	0: ssOut[0] is controlled by HW core 1: ssOut[0] is controlled by ss_man[0]
reserved	[23:20]			
ss_man[3]	[19]	b0	R/W	Valid when ss_man_en[3]=1. 0: ssOut[3] is low 1: ssOut[3] is high
ss_man[2]	[18]	b0	R/W	Valid when ss_man_en[2]=1. 0: ssOut[2] is low 1: ssOut[2] is high
ss_man[1]	[17]	b0	R/W	Valid when ss_man_en[1]=1. 0: ssOut[1] is low 1: ssOut[1] is high
ss_man[0]	[16]	b0	R/W	Valid when ss_man_en[0]=1. 0: ssOut[0] is low 1: ssOut[0] is high
reserved	[15:12]			
sspol[3]	[11]	b0	R/W	This register is only useful in Master mode 0: ssOut[3] is active low 1: ssOut[3] is active high
sspol[2]	[10]	b0	R/W	This register is only useful in Master mode 0: ssOut[2] is active low 1: ssOut[2] is active high
sspol[1]	[9]	b0	R/W	This register is only useful in Master mode 0: ssOut[1] is active low 1: ssOut[1] is active high
sspol[0]	[8]	b0	R/W	0: ssOut[0] (Master mode) or ssIn (Slave mode) is active low 1: ssOut[0] (Master mode) or ssIn (Slave mode) is active high
reserved	[7:4]			
ssout_en[3]	[3]	b0	R/W	0: Disable 1: Active ssOut for slave 3
ssout_en[2]	[2]	b0	R/W	0: Disable 1: Active ssOut for slave 2
ssout_en[1]	[1]	b0	R/W	0: Disable 1: Active ssOut for slave 1
ssout_en[0]	[0]	b0	R/W	0: Disable 1: Active ssOut for slave 0

- Offset 0x14: Mater Clock Divider Register

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
clk_diven	[8]	b0	R/W	Master clock divider enable 0: Master clock is directed from 32M/16M/RCO1M 1: Master clock is divided according to “clk_divsel”.
clk_divsel	[7:0]	b0	R/W	Master clock divider select. This register is only useful when “clk_diven” = 1. The frequency of Master clock is fsrc/2/(clk_divsel+1), where fsrc is the frequency of source clock (32M/16M/RCO1M).

- Offset 0x1C: Delay Register

Signal	Bits	Default	R/W	Description
reserved	[31:24]			
post_delay	[23:24]	x00	R/W	This register is only useful in Master mode and “post_delay_en” = 1. It controls the post-delay for the master serializer between last SCLK and ssOut inactive. The delay is (post_delay+1) SCLK cycles.
inter_delay	[15:8]	x00	R/W	This register is only useful in Master mode and “inter_delay_en” = 1. It controls the inter-transfer delay for the master serializer between each data (according to “bitsize”). The delay is (inter_delay+1) SCLK cycles.
pre_delay	[7:0]	x00	R/W	This register is only useful in Master mode and “pre_delay_en” = 1. It controls the pre-delay for the master serializer between ssOut active and first SCLK. The delay is (pre_delay+1) SCLK cycles.

Registers from offset 0x20 to 0x28 are for interrupt controls and status. There are 6 interrupt sources of the QSPI Control. The bitmap of these interrupt sources is listed in Table 19-1.

Table 19-1 Bitmap of QSPI Interrupts

Bit	Condition	Triggered
[5]	rx_not_empty	RX FIFO becomes not empty
[4]	trx_done	In Master mode, TX serializer becomes IDLE and TX FIFO is empty. In Slave mode, ssIn changes from active to inactive
[3]	rx_full	RX FIFO becomes full
[2]	reserved	
[1]	reserved	
[0]	tx_empty	TX FIFO becomes empty

- Offset 0x20: Interrupt Enable Register

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
intr_en	[5:0]	x00	R/W	Write 1 to enable individual interrupts (see Interrupt Mapping). Read will return status of the enabled interrupts.

- Offset 0x24: Interrupt Status Register

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
intr_status	[5:0]	x00	RO	Interrupt Status 1: interrupt source status 0: no interrupt

- Offset 0x28: Interrupt Clear Register

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
intr_clr	[5:0]	x00	W1C	Clear Interrupt Write 1 to clear the interrupt bit Write 0 has no effect

- Offset 0x2C: Enable Register

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
enable	[0]	b0	R/W	SPI enable. Read back this register to check the real status of HW. 0: Disable SPI. RX and TX FIFOs are reset when this register is from 1 to 0. 1: Enable SPI.

- Offset 0x30: Status Register

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
rx_fifo_full	[7]	b0	RO	RX FIFO full flag
reserved	[6]			
rx_fifo_empty	[5]	b1	RO	RX FIFO empty flag
tx_fifo_full	[4]	b0	RO	TX FIFO full flag
reserved	[3]			
tx_fifo_empty	[2]	b1	RO	TX FIFO empty flag
reserved	[1]			
trx_busy	[0]	b0	RO	TX/RX transfer is busy

- Offset 0x34: TX FIFO Level Register

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
tx_level	[5:0]	x00	RO	The number of entries in TX FIFO (0 ~ 32)

- Offset 0x38: RX FIFO Level Register

Signal	Bits	Default	R/W	Description
reserved	[31:6]			
rx_level	[5:0]	x00	RO	The number of entries in RX FIFO (0 ~ 32).

- Offset 0x40: DMA_RX_BASE

Signal	Bits	Default	R/W	Description
dma_rx_base	[31:0]	x0	R/W	Start address of RX buffer in SRAM for DMA

- Offset 0x44: DMA_RX_LENGTH

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_rx_length	[15:0]	x0	R/W	DMA transfer byte length for RX

- Offset 0x48: DMA_TX_BASE

Signal	Bits	Default	R/W	Description
dma_tx_base	[31:0]	x0	R/W	Start address of TX buffer in SRAM for DMA

- Offset 0x4C: DMA_TX_LENGTH

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_tx_length	[15:0]	x0	R/W	DMA transfer byte length for TX

- Offset 0x50: DMA_RX_REMAIN

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_rx_remain_len	[15:0]	x0	RO	DMA transfer's byte length remaining for RX

- Offset 0x54: DMA_TX_REMAIN

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_tx_remain_len	[15:0]	x0	RO	DMA transfer's byte length remaining for TX

- Offset 0x58: DMA_INTR_EN

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
intr_en_dma_tx	[1]	b0	R/W	Interrupt enable for DMA TX
intr_en_dma_rx	[0]	b0	R/W	Interrupt enable for DMA RX

- Offset 0x5C: DMA_INTR_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
intr_status_dma_tx	[1]	b0	R/W1C	Interrupt status for DMA TX
intr_status_dma_rx	[0]	b0	R/W1C	Interrupt status for DMA RX

- Offset 0x60: DMA_RX_ENABLE

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
dma_tx_dummy_data	[15:8]	x0	R/W	Dummy data for “dma_tx_dummy_en”
reserved	[7:2]			
dma_tx_dummy_en	[1]	b0	R/W	Dummy data enable. This register is only useful in Master mode. When the transaction is for RX only, this register should be 1 before “dma_rx_enable” is set to 1.
dma_rx_enable	[0]	b0	R/W	RX DMA enable. Write 1 to start RX DMA. Write 0 for completion or abort.

- Offset 0x64: DMA_TX_ENABLE

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
dma_tx_enable	[0]	b0	R/W	TX DMA enable. Write 1 to start TX DMA. Write 0 for completion or abort.

20. I²C Master

This chapter describes the I²C Master Interface Peripheral, referred to as I2CM. It provides support for the communications link between integrated circuits in a system. It is a simple two-wire bus with a software-defined protocol for system control, which is used in temperature sensors and voltage level translators to EEPROMs, general-purpose I/O, A/D and D/A converters, CODECs, and many types of microprocessors.

I²C Master supports the following features:

- Clock synchronization
- 7-bit addressing
- Transmit (9x16) and receive (8x16) buffers
- Interrupt or polling mode operation

20.1. Block Diagram

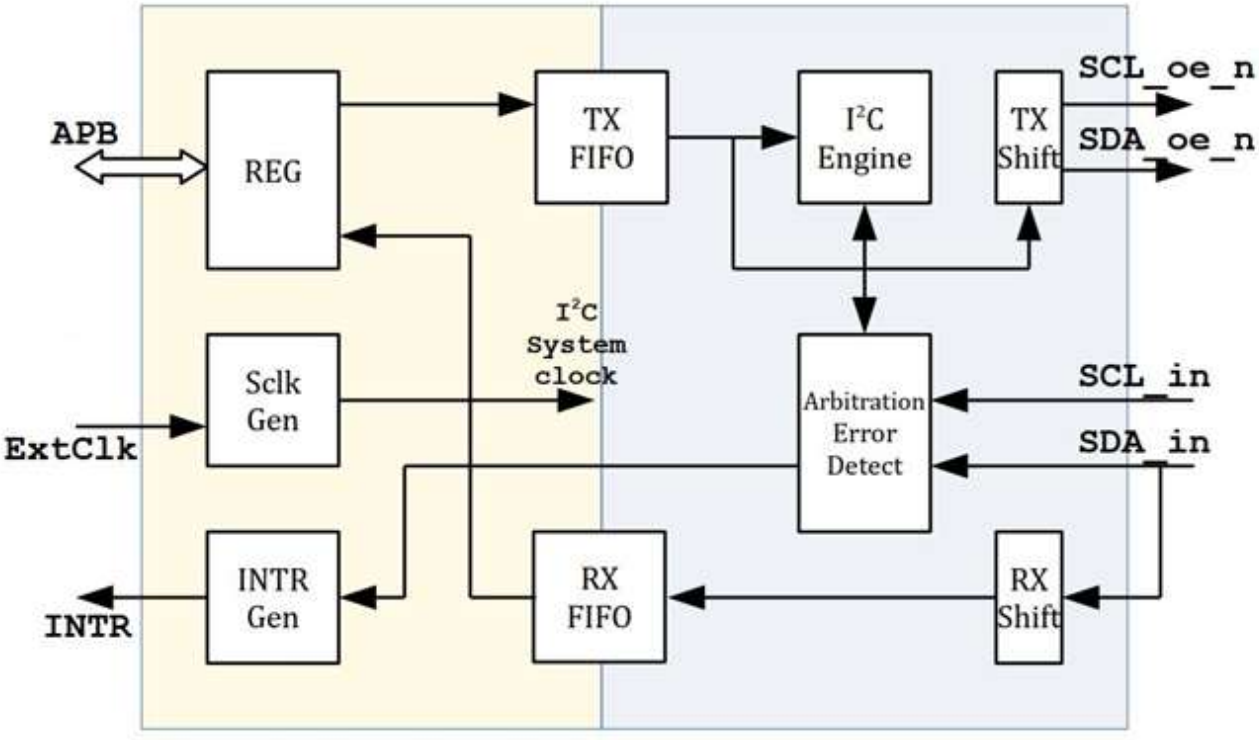


Figure 20-1 Block Diagram of I²C Master

20.2. Functional Descriptions

The I²C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. This circuit is used as a master device.

20.2.1. Clock Divider

The I²C source Clock Generator module can be configured to generate a wide range of frequencies. An external clock source is divided down. The external clock can be chosen from 32M, 16M, or RCO 1M, which is determined by system setting. The clock divider outputs the clock enable of desired frequency to the I²C transmit and receive logic.

The formula below gives the resulting I²C System Clock Frequency as a function of Selected Clock Frequency and Clock Divide Value (from the **SCLK_GEN** Registers):

$$I2CSystemClockFrequency = \frac{SelClockFrequency}{(ClockDivideValue + 1)}$$

Note that the I²C System Clock Frequency is a factor of 4 faster than the nominal I²C Bus Clock (SCL) frequency.

$$I2CBusClockFrequency = \frac{I2CSystemClockFrequency}{4} = \frac{SelClockFrequency}{(ClockDivideValue + 1) \times 4}$$

The additional factor of 4 is necessary for the I²C Engine to generate all phases of the I²C Bus Clock, and to generate start and stop conditions on the I²C Bus that are synchronous to the I²C System Clock.

20.2.2. START and STOP

All transactions begin with a START (S) and are terminated by a STOP (P).

A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition.

A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

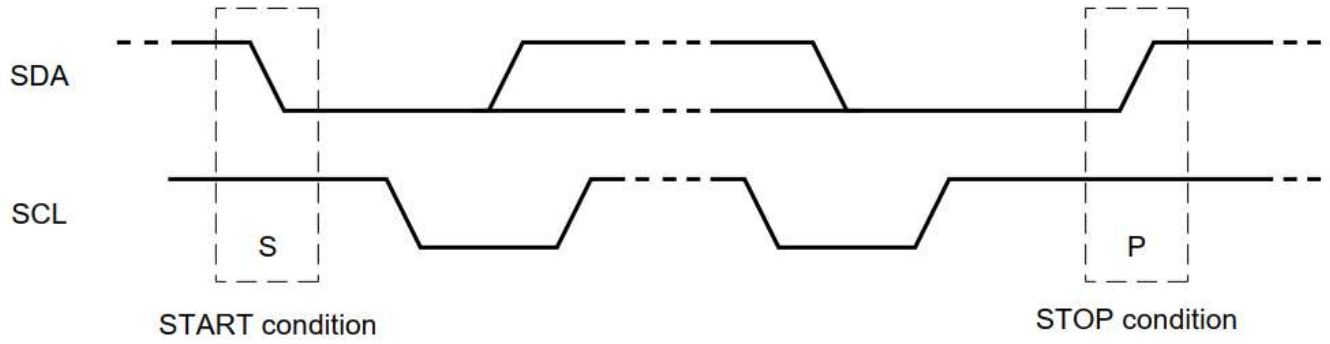


Figure 20-2 START and STOP bits of I²C Interface

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition.

The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical.

20.2.3. Byte Format

Every byte put on the SDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge bit. Data is transferred with the Most Significant Bit (MSB) first.

If a slave cannot receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can stretch the clock line SCL LOW to force the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCL.

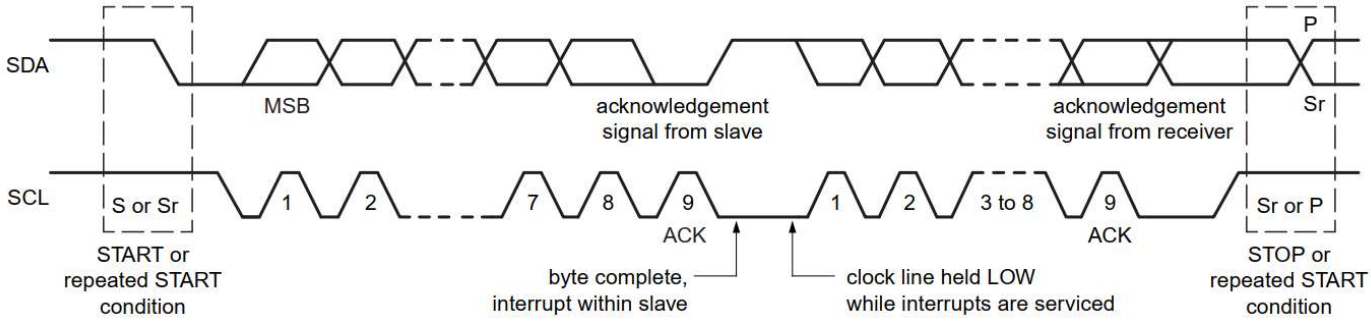


Figure 20-3 The Byte Format of I²C Interface

20.2.4. Clock Synchronization

Clock synchronization is performed using the wired-AND connection of I²C interfaces to the SCL line. The SCL line is held LOW by the master with the longest LOW period. Masters with shorter LOW periods enter a HIGH wait-state during this time.

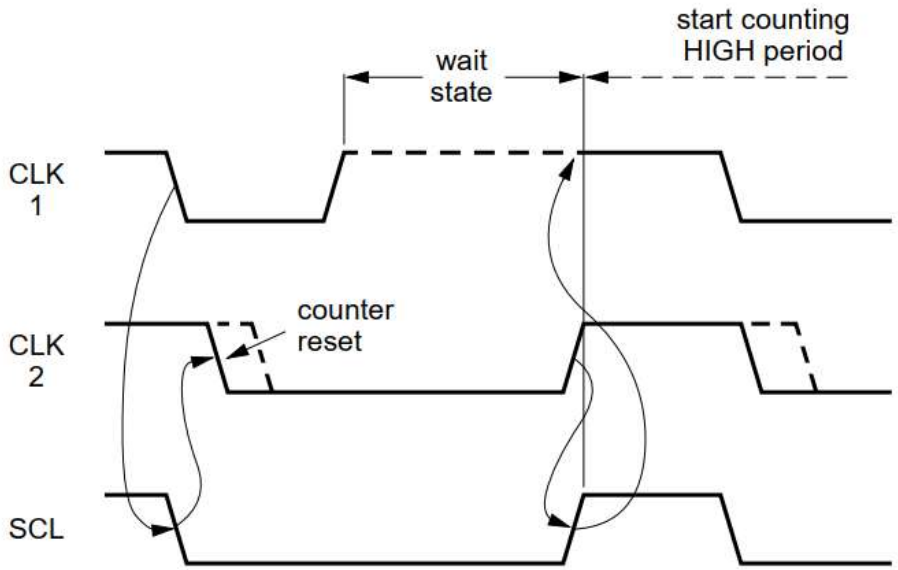


Figure 20-4 The Clock Synchronization of I²C Interface

20.2.5. Arbitration

A master may start a transfer only if the bus is free. Two masters may generate a START condition within the minimum hold time of the START condition which results in a valid START condition on the bus. Arbitration is then required to determine which master will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks to see if the

SDA level matches what it has sent. The first time a master tries to send a HIGH, but detects that the SDA level is LOW, the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transaction.

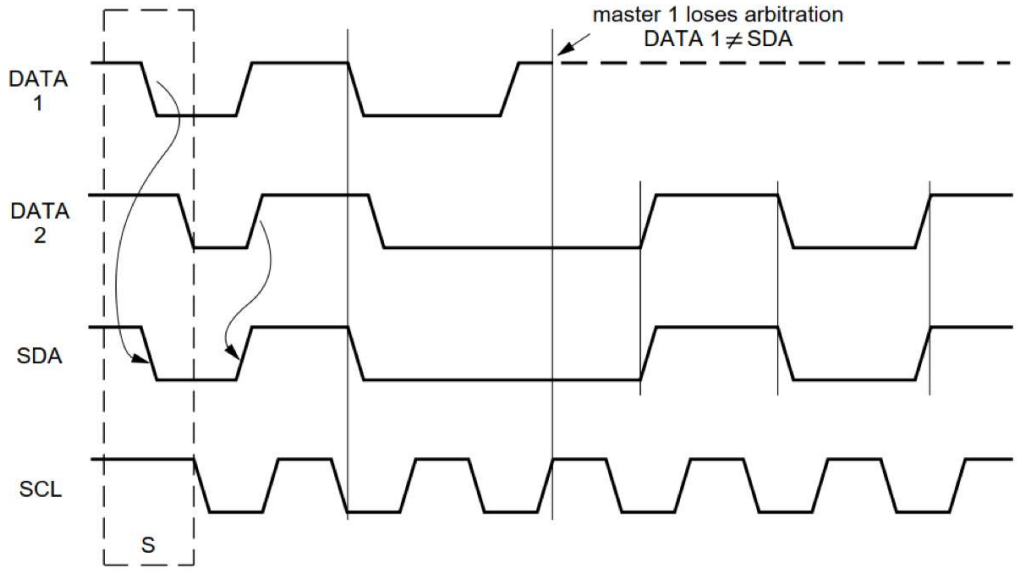


Figure 20-5 Arbitration of I²C Interface

20.3. Programming Sequences

20.3.1. Normal Read/Write Operations

- Disable the I2CM by writing 0 to the **CTRL** register.
- Write to the **SCLK_GEN** register, the source clock of I²C interface clock.
- Write to the **TAR** register, the address of the I²C device to be addressed (bits 6:0).
- Enable restart feature if the target support by writing a '1' in bit 1 (**RESTART_EN**) of the **CTRL** register.
- Enable the I2CM by writing a '1' in bit 0 (**I2CM_ENABLE**) of the **CTRL** register.
- Now write transfer direction and data to be sent to the **BUF** register. If the **BUF** register is written before the **I2CM_ENABLE** is enabled, the data and commands are still written to the buffers.

  This step generates the START condition and the address byte on the I2CM. Once **I2CM_ENABLE** is enabled and there is data in the TX FIFO, I2CM starts reading the TX data.

- I2CM would execute TX commands and handle I²C protocol for read/write interleaved. If the last of TX data is a write command, I²C transaction would stop at ACK bit (9th bit). The event of **TX_EMPTY** occurs. If the last of TX data is a read command, I²C transaction would stop at the last DATA bit (8th bit). The event of **RX_FINISH** occurs.
- Terminate the I²C transaction by writing a '1' in bit 2 (**STOP_EN**) of the **CTRL** register. If the last transaction is a write cycle, the STOP condition issues. If the last transaction is a read cycle, the NACK and STOP condition issue.

20.3.2. Abort for Error Conditions

- If the I²C transaction is aborted for **ABRT_A_NACK**, **ABRT_W_NACK** or **ABRT_LOST_ARB**, the I2CM would issue STOP condition for **ABRT_A_NACK** and **ABRT_W_NACK** and then stop the engine.
- In the ISR for handling the abort interrupt, the TX and RX FIFO should be cleared by writing a '1' in bit 4 (**FIFO_Clear_EN**) of the **CTRL** register.
- If the data line (SDA) is stuck LOW driven by device, force I2CM to send nine clock pulses by writing a '1' in bit 3 (**BUS_Clear_EN**) of the **CTRL** register.
- Disable the I2CM by writing 0 to the **CTRL** register.

20.4. Registers

- Offset 0x00: I2CM_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:5]			
FIFO_Clear_EN	[4]	b0	R/W	If the I²C transaction is abort for some error conditions, we could clear the transmit and receive buffers by setting FIFO_Clear_EN. Write (Set): a value of 1 to force to clear the transmit and receive buffers. Read (Status): a value of 0 to indicate the process of FIFO clear is done.
BUS_Clear_EN	[3]	b0	R/W	If the data line (SDA) is stuck LOW driven by device, enable this bit to send nine clock pulses by the master engine. BUS_Clear_EN is set by software and clear by hardware. Write (Set): a value of 1 to force engine to send nine clock pulses. Read (Status): a value of 0 to indicate the bus clear process is done.
STOP_EN	[2]	b0	R/W	Force to stop I²C transaction after transmit buffer empty.

				STOP_EN is set by software and clear by hardware. Write (Set): a value of 1 to indicate the I²C transaction would be terminated after transmit buffer empty. Read (Status): a value of 0 to indicate all the transaction is done, I²C bus is stop and back to IDLE state.
RESTART_EN	[1]	b0	R/W	Enable "RESTART" function 1: enable 0: disable
I2CM_ENABLE	[0]	b0	R/W	Enable I2CM engine. 1: enable 0: disable

- Offset 0x04: I2CM_TAR

Signal	Bits	Default	R/W	Description
reserved	[31:7]			
TAR	[6:0]	x50	R/W	7-bit I²C Target Address

- Offset 0x08: I2CM_BUF

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
CMD	[8]	b0	WO	This bit controls whether a read or a write is performed. 1: read; 0: write When a command is entered in the TX FIFO, this bit distinguishes the write and read commands.
DAT	[7:0]	b0	R/W	This register contains the data to be transmitted or received on the I²C bus. If you are writing to this register and want to perform a read, bits [7:0] (DAT) are ignored by the I2CM. However, when you read this register, these bits return the value of data received on the I²C interface.

- Offset 0x0C: I2CM_INTR_STAT

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
R_IDLE_STATE	[9]	b0	RO	See INTR_RAW_STAT register for a detailed description of these bits.
R_ABRT_LOST_ARB	[8]	b0	RO	
R_ABRT_W_NACK	[7]	b0	RO	
R_ABRT_A_NACK	[6]	b0	RO	
R_TX_EMPTY	[5]	b0	RO	
R_TX_OVER	[4]	b0	RO	
R_RX_FINISH	[3]	b0	RO	
R_RX_FULL	[2]	b0	RO	
R_RX_OVER	[1]	b0	RO	
R_RX_UNDER	[0]	b0	RO	

- Offset 0x10: I2CM_INTR_ENABLE

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
E_IDLE_STATE	[9]	b0	R/W	These bits enable their corresponding interrupt status bits in the INTR_STAT register.
E_ABRT_LOST_ARB	[8]	b0	R/W	
E_ABRT_W_NACK	[7]	b0	R/W	
E_ABRT_A_NACK	[6]	b0	R/W	
E_TX_EMPTY	[5]	b0	R/W	
E_TX_OVER	[4]	b0	R/W	
E_RX_FINISH	[3]	b0	R/W	
E_RX_FULL	[2]	b0	R/W	
E_RX_OVER	[1]	b0	R/W	
E_RX_UNDER	[0]	b0	R/W	

- Offset 0x14: I2CM_INTR_RAW_STAT

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
IDLE_STATE	[9]	b1	RO	This bit captures I2CM in IDLE state and stays set until it is cleared.
ABRT_LOST_ARB	[8]	b0	RO	This bit is set to 1 when I2CM has lost arbitration.
ABRT_W_NACK	[7]	b0	RO	This bit is set to 1 when I2CM did not receive an acknowledge from the remote slave for the write data.
ABRT_A_NACK	[6]	b0	RO	This bit is set to 1 when the address sent was not acknowledged by any slave.
TX_EMPTY	[5]	b1	RO	This bit is set to 1 when the transmit buffer is empty.
TX_OVER	[4]	b0	RO	Set when the transmit buffer is full and the processor attempts to issue another I^2C command by writing to the BUF register.
RX_FINISH	[3]	b0	RO	Set when the transmit buffer is empty and the last byte of read data is put into the receive buffer.
RX_FULL	[2]	b0	RO	Set when the receive buffer is full.
RX_OVER	[1]	b0	RO	Set if the receive buffer is completely filled and an additional byte is received from an external I^2C device. The I2CM acknowledges this, but any data bytes received after the FIFO is full are lost.
RX_UNDER	[0]	b0	RO	Set if the processor attempts to read the receive buffer when it is empty by reading from the BUF register.

- Offset 0x18: I2CM_INTR_CLR

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
C_ABRT_LOST_ARB	[8]	b0	W1C	Write 1 to clear R_ABRT_LOST_ARB and ABRT_LOST_ARB.
C_ABRT_W_NACK	[7]	b0	W1C	Write 1 to clear R_ABRT_W_NACK and ABRT_W_NACK.
C_ABRT_A_NACK	[6]	b0	W1C	Write 1 to clear R_ABRT_A_NACK and ABRT_A_NACK.
C_TX_EMPTY	[5]	b0	W1C	Write 1 to clear R_TX_EMPTY and TX_ EMPTY.
C_TX_OVER	[4]	b0	W1C	Write 1 to clear R_TX_OVER and TX_OVER.
C_RX_FINISH	[3]	b0	W1C	Write 1 to clear R_RX_FINISH and RX_FINISH.
C_RX_FULL	[2]	b0	W1C	Write 1 to clear R_RX_FULL and RX_FULL.

C_RX_OVER	[1]	b0	W1C	Write 1 to clear R_RX_OVER and RX_OVER.
C_RX_UNDER	[0]	b0	W1C	Write 1 to clear R_RX_UNDER and RX_UNDER.

- Offset 0x1C: I2CM_SCLK_GEN

Signal	Bits	Default	R/W	Description
reserved	[31:14]			
SclkDiv	[13:0]	x0	R/W	I^2C Source Clock Divider.

21. I²C Slave

This chapter describes the I²C Slave Interface Peripheral, referred to as I2CS. It is a simple two-wire bus with a software-defined protocol for system control. This circuit is designed for single byte process of transmission/reception and need software to interfere during every byte.

The I²C Slave supports those features:

- Clock stretching
- 7-bit addressing
- Interrupt or polling mode operation

21.1. Block Diagram

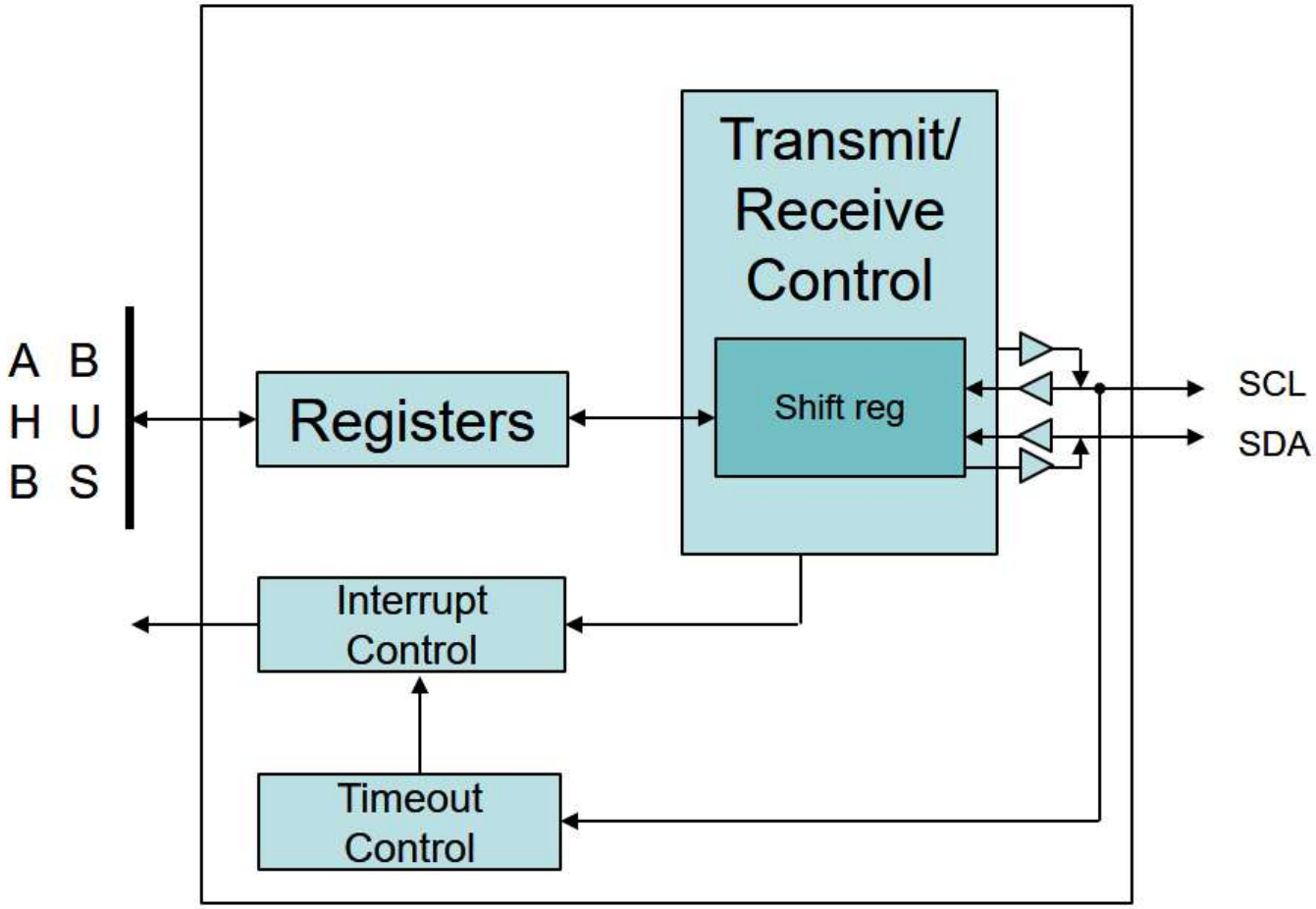


Figure 21-1 Block Diagram of I²C Slave

21.2. Functional Description

The I²C bus is a two-wire serial interface, consisting of a serial data line (SDA) and a serial clock (SCL). These wires carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a “transmitter” or “receiver,” depending on the function of the device. Devices can also be considered as masters or slaves when performing data transfers. This circuit is used as a slave device.

21.2.1. Transmitting and Receiving Control

When a master sends START condition and slave address byte, the controller check the 7-bit value

whether matching value of register “slv_addr” or not. If it matches, the controller responds with ACK and assert interrupt status “addr_match_intr”. Also, the SCL is stretched LOW until software takes corresponding action. The register “rw_flag” indicates this transaction is intended for reading or writing data. Software need to determine to write data to register “tx_data” or read data from register “rx_data” according to “rw_flag”. For the following data bytes, when the byte is for writing, the controller latches the data byte at “rx_data” and responds with ACK. When the byte is for reading, the controller converts “tx_data” to serial bits and send to SDA line.

21.2.2. Timeout

To solve the problem when I^2C bus has some error and bus hangs, a timeout function is provided to inform software. The timeout function is enabled when register “timeout_en” is set and starts to counting time when “bus_busy” is 1. The timer is reset at falling edge of SCL line. If the timer reaches the configured value, the interrupt status “timeout_intr” asserts. The timeout period is calculate by (1+”timeout_val”) * 32 / f_{clk} ms, where f_{clk} is the clock frequency of the internal clock, hclk.

21.2.3. Control Flow

Figure 21-2 shows the full flow for software to interact with this module. The interrupt status can be used in both interrupt or polling mode.

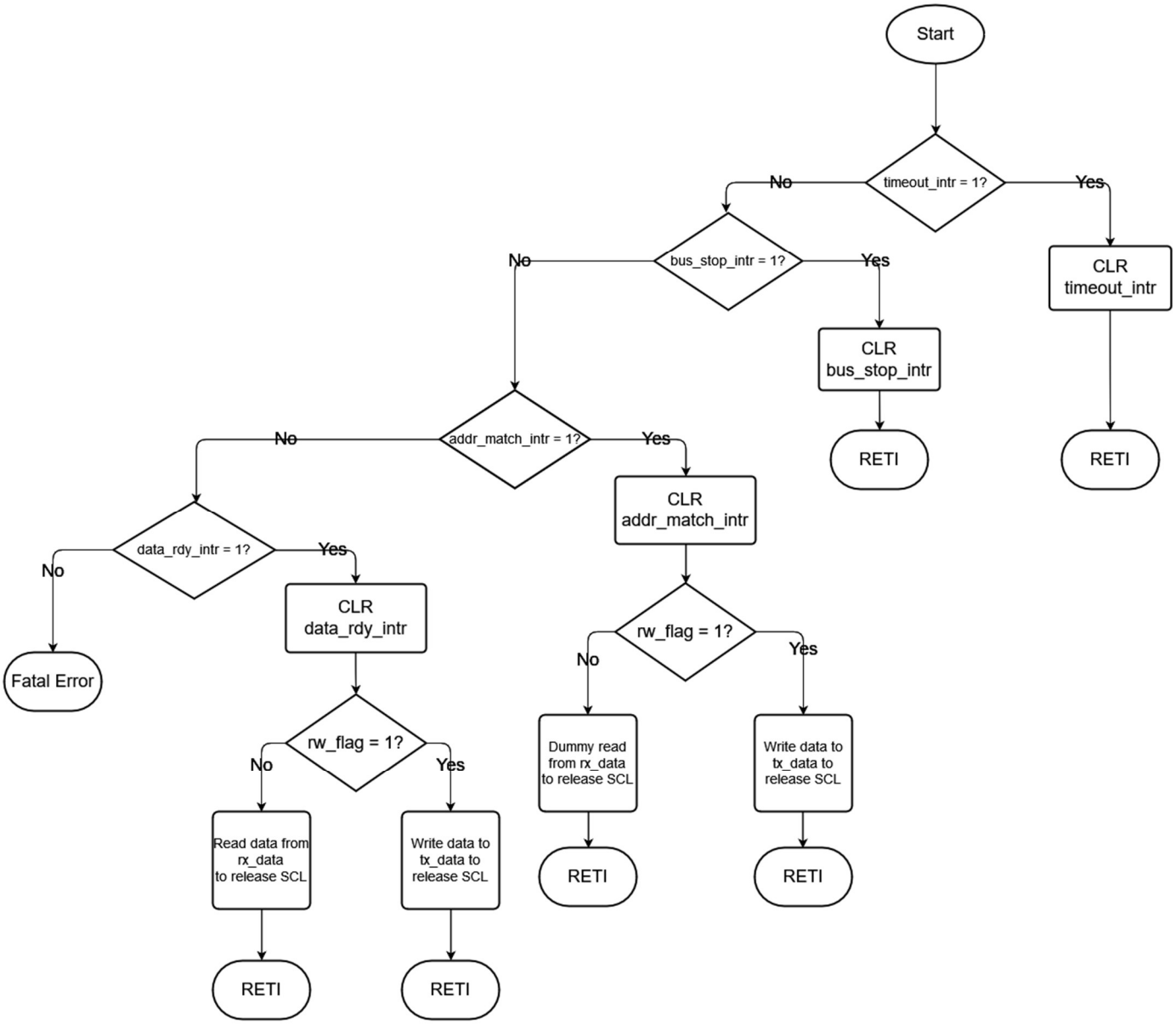


Figure 21-2 Control Flow of I²C Slave

21.3. Registers

- Offset 0x00: I2CS_DATA

Signal	Bits	Default	R/W	Description
reserved	[31:8]			

tx_data	[7:0]	b0	W	Data to be transmitted to master
rx_data	[7:0]	b0	R	Received data byte from master

- Offset 0x04: I2CS_ADDR

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
slv_addr	[7:1]	x0	R/W	7-bit I^2C Target Address
reserved	[0]			

- Offset 0x08: I2CS_IER

Signal	Bits	Default	R/W	Description
reserved	[31:4]			
timeout_intr_en	[3]	b0	R/W	Enable of interrupt timeout_intr
bus_stop_intr_en	[2]	b0	R/W	Enable of interrupt bus_stop_intr.
data_rdy_intr_en	[1]	b0	R/W	Enable of interrupt data_rdy_intr
addr_match_intr_en	[0]	b0	R/W	Enable of interrupt addr_match_intr

IER determines whether ISR issues interrupts to MCU or not. Note that the state of ISR works normally even when IER is disabled.

- Offset 0x0C: I2CS_ISR

Signal	Bits	Default	R/W	Description
reserved	[31:4]			
timeout_intr	[3]	b0	R/W1C	When timeout occurs, this bit asserts. Write 1 to clear it.
bus_stop_intr	[2]	b0	R/W1C	When STOP condition is detected, this bit asserts. Write 1 to clear it.
data_rdy_intr	[1]	b0	R/W1C	When data byte is received or transmitted, this bit asserts. Write 1 to clear it.
addr_match_intr	[0]	b0	R/W1C	When slave address byte is detected with value matching slv_addr, this bit asserts. Write 1 to clear it.

- Offset 0x10: I2CS_TIMEOUT

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
timeout_val	[15:8]	b0	R/W	Timeout period. The period is calculated by (1+"timeout_val") * 32 / fclk ms, where fclk is the clock frequency of PCLK.
reserved	[7:1]			
timeout_en	[0]	b0	R/W	Enable timeout function when SCL being not changed for certain period

-

- Offset 0x14: I2CS_ENABLE

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
i2c_enable	[0]	b0	R/W	Enable I2C slave module. Set this bit after all above setting is ready.

- Offset 0x18: I2CS_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
rw_flag	[1]	b0	R	Flag to indicate the following byte is for reading or writing. 0: write 1: read
bus_busy	[0]	b0	R	The bus busy status. This bit asserts when START condition is detected and de-asserts when STOP condition is detected.

22. PWM

The PWM module generates pulse width modulated signals and drives the assigned GPIOs. It supports a pulse generator with up & up-and-down counting modes. Five PWM modules can provide up to 5 PWM channels and each channel have its own individual frequency control. A read-type xDMA is also included in the PWM module to transfer frequency control between memory and the PWM module without CPU intervention.

22.1. Block Diagram

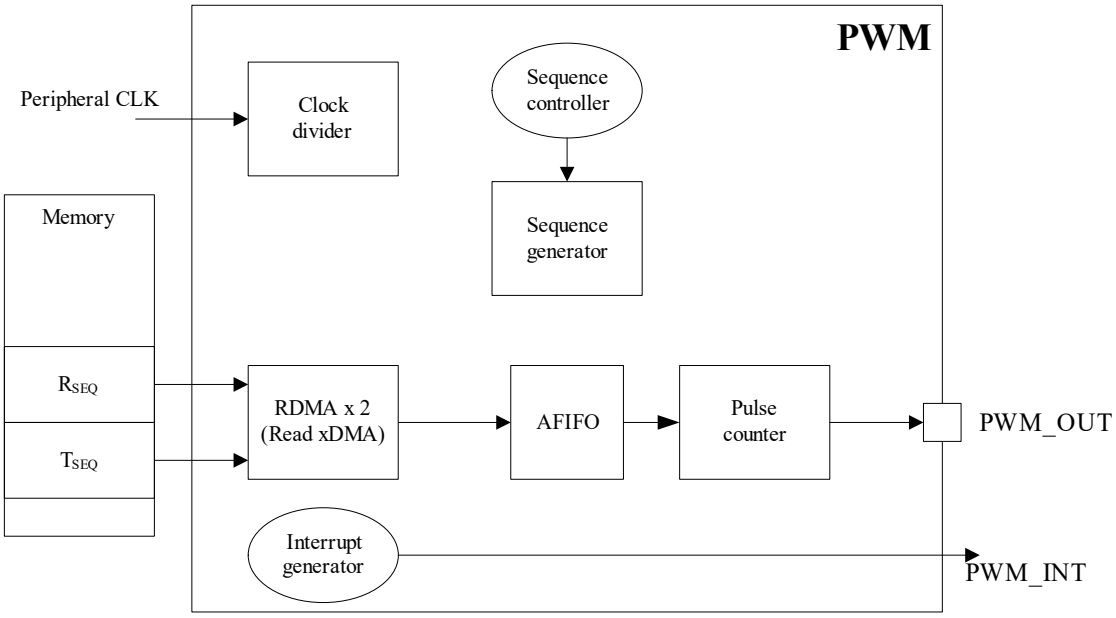


Figure 22-1 Block Diagram of PWM Control

22.2. Features

- Programmable clock divider for PWM module
- 5 sets of PWM module and up to five channels for PWM
- PWM pulse generator with up & up-and-down counting modes
- Programmable duty-cycle sequence defined in the memory
- Duty-cycle sequence can be repeated or loop controlled.

- Register mode for low power mode PWM application

22.3. Functional Description

For simplicity, PWMx means PWM0, PWM1, PWM2, PWM3 or PWM4. Some examples use PWM0 as an introduction since all five PWM hardware are the same.

22.3.1. Pulse Generator

The pulse generator consists a counter with up and up-and-down counting modes. The internal logic compares the threshold (THD) with the counter value and generates a defined pulse duration to the PWM output. Phase (PHA) indicates the polarity of pulse generator output when the counter starts to count from zero. Once the counter exceeds the THD it will toggle the PWM output. This is illustrated in Figure 22-2.

For the Up counting mode, the counter increases automatically one by one. The counter will wrap around to zero whenever the counter reaches counter end value (CNT_END).

For Up-and-down counting mode, the counter increases automatically one by one. The counter will stop increasing and decrease one by one whenever the counter reaches counter end value (CNT_END).

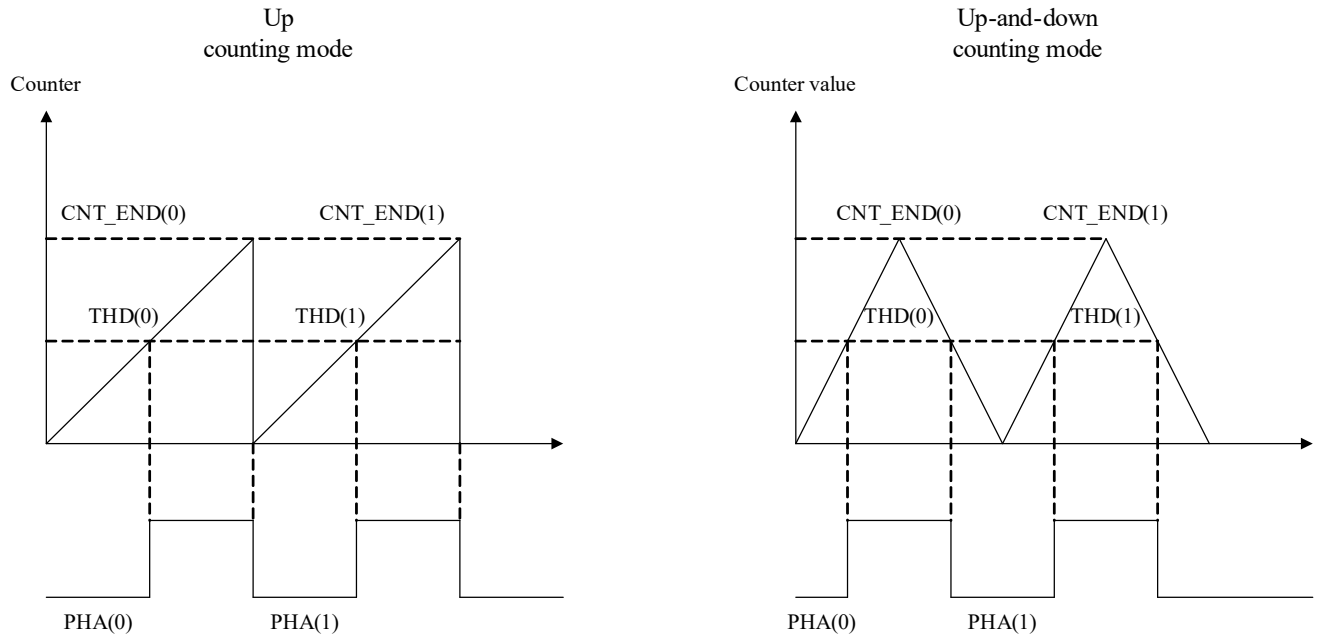


Figure 22-2 Pulse Generator Schemes of PWM Control

22.3.2. Data Format of xDMA

Each PWM module has two xDMA with read types RDMA0 and RDMA1. Each xDMA addressing mode is word (4-bytes) alignment.

For RDMA0, the segment size and block size are configured from PWMx_RDMA0_SET0. The start address for memory reading is configured from PWMx_RDMA0_SET1. The PWMx_RDMA0_CTL0 is used to trigger RDMA for memory reading. The sample sequence stored in memory is configured from PWMx_SET0.

For RDMA1, the segment size and block size are configured from PWMx_RDMA1_SET0. The start address point for memory reading is configured from PWMx_RDMA1_SET1. The PWMx_RDMA0_CTL0 triggers RDMA to start memory reading. The sample sequence stored in memory is configured from PWMx_SET0.

The RDMA0 reads sequence R_{SEQ} from a memory range that is defined by PWMx_RDMA0_SET0~1. The RDMA1 reads sequence T_{SEQ} from a memory range that is defined by PWMx_RDMA1_SET0~1. The samples of both sequences stored in the memory have two formats.

Format-0 is each 32-bit memory data representing two samples with phase and threshold information. The 32-bits consist of two of 1-bit PHA and two 15-bit THD values. The counter end value depends on the PWMx_SET1 register.

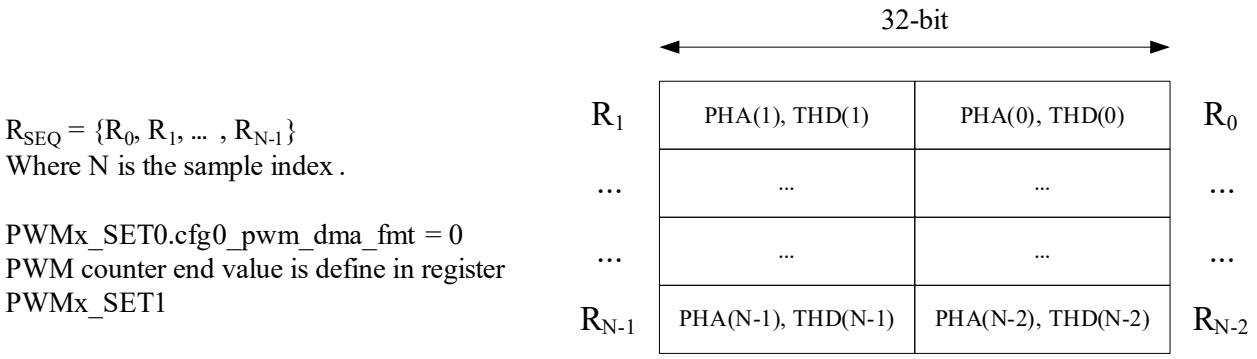


Figure 22-3 xDMA Format-0 of PWM Control

Format-1 is each 32-bit of memory data representing one sample with counter end value, phase and threshold information. The 32-bits consist of a 16-bit CNT_END, 1-bit PHA and 15-bit THD value.

$R_{SEQ} = \{R_0, R_1, \ldots, R_{N-1}\}$
Where N is the sample index .

PWMx_SET0.cfg0_pwm_dma_fmt = 1
PWM counter end value equals to
CNT_END(1), … , CNT_END(N)

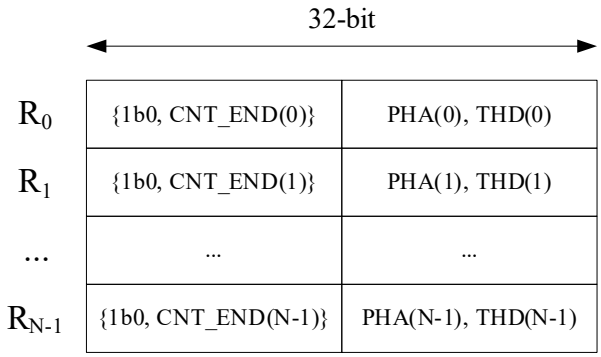


Figure 22-4 xDMA Format-1 of PWM Control

When R_{SEQ} sample format is defined, the T_{SEQ} sample format of memory content is as same as R_{SEQ}.

22.3.3. Sequence Controller

The sequence controller for R_{SEQ} and T_{SEQ} has two modes: single sequence mode and two sequence mode. Each sample of the sequence can also be manipulated by the sequence controller. It provides repeat and delay of each sample of the sequence.

Below is an example of single sequence mode for R_{SEQ} that has 5 samples stored in the memory, repeat 1-time and end sequence with 2-sample delay by repeating last sample.

Table 22-1 Configuration Summary of Single Sequence Mode

Singe sequence mode, R_{SEQ} is 5-samples, repeat 1-time and end with delay 2-sample.	
Sequence controller configurations	PWM0_SET0. cfg0_seq_order = 0 (play R_{SEQ} 1st) PWM0_SET1.cfg0_pwm_cnt_end = 100 (counter ends at 99) PWM0_SET2.cfg0_seqx_pcnt = 0 (single sequence)
R_{SEQ} register configurations	PWM0_SET3.cfg0_seq0_num=5 (number of sample) PWM0_SET4.cfg0_seq0_rpt=1 (repeat number of sample) PWM0_SET5.cfg0_seq0_dly=2 (delay 2-sample at last)
R_{SEQ}	$\{R_0, R_1, \ldots, R_4\}$, stored in the memory
PWM output	$\{R_0, R_0, R_1, R_1, \ldots, R_4, R_4, R_{4(D)}, R_{4(D)}\}$ $R_{4(D)}$ are sample delay by repeating last sample

Another example of a two sequence mode for R_{SEQ} that has 5 samples stored in the memory, repeat 1-time and end sequence with 2-sample delay by repeating last sample. T_{SEQ} that has 5 samples stored in the memory, repeat 0-time and end sequence with 1-sample delay by repeating last sample. The two sequence can play in a loop several times according to PWM0_SET2.cfg0_seqx_pcnt.

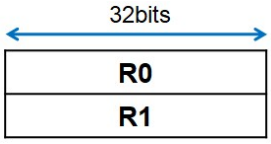
Table 22-2 Configuration Summary of Two Sequence Mode

Two sequence mode, R_{SEQ} is 5-samples, repeat 1-time and end with delay 2-sample. T_{SEQ} is 5-samples, repeat 0-time and end with delay 1-sample.	
Sequence controller configurations	PWM0_SET0. cfg0_seq_order = 0 (play R_{SEQ} 1st, then play T_{SEQ}) PWM0_SET1.cfg0_pwm_cnt_end = 100 (counter ends at 99) PWM0_SET2.cfg0_seqx_pcnt = 4 (two sequences with loop play by 4)
R_{SEQ} register configurations	PWM0_SET3.cfg0_seq0_num=5 (number of sample) PWM0_SET4.cfg0_seq0_rpt=1 (repeat number of sample) PWM0_SET5.cfg0_seq0_dly=2 (delay 2-sample at last)
T_{SEQ} register configurations	PWM0_SET3.cfg0_seq1_num=5 (number of sample) PWM0_SET4.cfg0_seq1_rpt=0 (repeat number of sample) PWM0_SET5.cfg0_seq1_dly=2 (delay 2-sample at last)
R_{SEQ}	{R_0, R_1, …, R_4}, stored in the memory
T_{SEQ}	{T_0, T_1, …, T_4}, stored in the memory
PWM output	{R_0, R_0, R_1, R_1, ..., R_4, R_4, $R_{4(D)}$, $R_{4(D)}$, T_0, T_1, …, T_4, $T_{4(D)}$}, {R_0, R_0, R_1, R_1, ..., R_4, R_4, $R_{4(D)}$, $R_{4(D)}$, T_0, T_1, …, T_4, $T_{4(D)}$}, {R_0, R_0, R_1, R_1, ..., R_4, R_4, $R_{4(D)}$, $R_{4(D)}$, T_0, T_1, …, T_4, $T_{4(D)}$}, {R_0, R_0, R_1, R_1, ..., R_4, R_4, $R_{4(D)}$, $R_{4(D)}$, T_0, T_1, …, T_4, $T_{4(D)}$}. $R_{4(D)}$ represent sample delay by repeating last sample $T_{4(D)}$ represent sample delay by repeating last sample

22.3.4. Register Mode

The PWM support register mode for some power saving application. The PWM register mode is designed for working when only peripheral clock exist. Due to the DMA can't work without system clock, the PWM register mode generate pulse by data put in the register. The PWM register mode shared control register with normal PWM, but has independent interrupt and data registers.

The picture below shows different sequence in register mode. The PWM control is exactly the same with normal mode PWM, the only difference is the data source from register and the DMA control is removed.



Non-Continuous+1 set : {R0}
Non-Continuous+2 set : {R0、R1}
Continuous+1 set : {R0、R0、….、R0…}
Continuous+2 set : {R0、R1、….、R0、R1、…}

Figure 22-5 Register Mode of PWM Control

22.4. Registers

There are five independent PWMs, PWM0~PWM4, modules and each of them has the same register sets with different offsets. Offsets of each PWM module are listed in Table 22-3.

Table 22-3 Address Offset of PWM Modules

Module	Address Offset
PWM0	0x000
PWM1	0x100
PWM2	0x200
PWM3	0x300
PWM4	0x400

The exact address of a register is then become the sum of PWM base address, PWMx offset and the register offset. Below lists PWM0 registers only.

- Offset 0x00: PWM0_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:2]			
cfg0_ck_ena	[1]	b0	R/W	Enable pwm gated clock 0: disable 1: enable
cfg0_pwm_ena	[0]	b0	R/W	Enable pwm 0: disable 1: enable

- Offset 0x04: PWM0_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg0_pwm_rst	[0]	b0	W1C	Reset pwm 1: reset

- Offset 0x08: PWM0_SET0

Signal	Bits	Default	R/W	Description
cfg0_clk_div	[31:24]	d0	R/W	PWM gated clock divider value, only valid when >0
reserved	[23:21]			
cfg0_pwm_play_num	[20]	d0	R/W	PWM register mode data play number, used in register mode 0: 1 data 1: 2 data
reserved	[19:15]			
cfg0_pwm_ena_trig	[14:12]	d7	R/W	PWM counter enable trigger by other PWM 0: trigger whenever PWM0 sequence finish 1: trigger whenever PWM1 sequence finish

				2: trigger whenever PWM2 sequence finish 3: trigger whenever PWM3 sequence finish 4: trigger whenever PWM4 sequence finish others: self-trigger Note: if cfgX_seqx_cnt is 65535, trigger whenever PWMX 1st sequence finishes where X is 0~4.
cfg0_ck_div	[11:8]	d0	R/W	PWM gated clock divider value 0: clock divided by 1 1: clock divided by 2 2: clock divided by 4 3: clock divided by 8 4: clock divided by 16 5: clock divided by 32 6: clock divided by 64 7: clock divided by 128 8: clock divided by 256 otherwise: clock divided by 256
cfg0_reg_mode	[7]	d0	R/W	PWM register mode, used in register mode 0:DMA mode 1:register mode
cfg0_dma_auto	[6]	d1	R/W	Auto Trigger start DMA when the amount of playback 2-sequence play does not reach to cfgx_seqx_pcnt 0: No auto trigger 1: Auto trigger
cfg0_pwm_cnt_trig	[5]	d0	R/W	PWM counter trigger , set 1 in register mode 0: trigger by cfg_pwm_ena 1: trigger by cfg_pwm_ena & FIFO not empty
cfg0_pwm_cnt_mode	[4]	d0	R/W	PWM counter mode , used in register mode 0: Up counter 1: Up and down counter
cfg0_pwm_dma_fmt	[3]	d0	R/W	PWM dma sample format , used in register mode 0: Format-0, 32-bit is PHA(N), THD(N), PHA(N-1), THD(N-1) 1: Format-1, 32-bit is 1'b0, CNT_END(N), PHA(N), THD(N)
cfg0_seq_mode	[2]	d0	R/W	PWM sequence play mode , used in register mode 0: non-continuous play 1: continuous play
cfg0_seq_two_sel	[1]	d0	R/W	Two Sequence selection 0: One sequence 1: Two sequence
cfg0_seq_order	[0]	d0	R/W	PWM sequence play order 0: RSEQ 1st 1: TSEQ 1st

- Offset 0x0C: PWM0_SET1

Signal	Bits	Default	R/W	Description
reserved	[31:15]			
cfg0_pwm_cnt_end	[14:0]	d0	R/W	PWM counter end value, ignored when

				cfg0_pwm_dma_fmt is at Format-1 value: 3…32767

- Offset 0x10: PWM0_SET2

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg0_seqx_pcnt	[15:0]	d0	R/W	Amount of sequence play 0: Play one/two sequence 1 time N: Play one/two sequence N-times 65535: Play one/two sequence infinitely

- Offset 0x14: PWM0_SET3

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg0_seq0_num	[15:0]	d0	R/W	Number of element in RSEQ

- Offset 0x18: PWM0_SET4

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg0_seq0_rpt	[15:0]	d0	R/W	Number of repeat for each element in RSEQ

- Offset 0x1C: PWM0_SET5

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg0_seq0_dly	[15:0]	d0	R/W	Number of delay after RSEQ is play finish

- Offset 0x20: PWM0_SET6

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg0_seq1_num	[15:0]	d0	R/W	Number of element in TSEQ

- Offset 0x24: PWM0_SET7

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg0_seq1_rpt	[15:0]	d0	R/W	Number of repeat for each element in TSEQ

- Offset 0x28: PWM0_SET8

Signal	Bits	Default	R/W	Description
reserved	[31:16]			

cfg0_seq1_dly	[15:0]	d0	R/W	Number of delay after TSEQ is play finish

- Offset 0x40: PWM0_RDMA0_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg0_pwm_rdma0_ctl0	[0]	d0	W1C	0 Start xdma for memory access 1: enable

- Offset 0x44: PWM0_RDMA0_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg0_pwm_rdma0_ctl1	[0]	d0	W1C	Software reset xdma 1: Reset

- Offset 0x48: PWM0_RDMA0_SET0

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma0_set0	[31:0]	d0	R/W	15:0 Segment size, 4-byte aligned 0: Illegal value Others: 1…65535 (Block size < Segment size) 31:16 Block size, 4-byte aligned 0: Single-shot mode Others: 1…65535 (Block size < Segment size)

- Offset 0x4C: PWM0_RDMA0_SET1

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma0_set1	[31:0]	d0	R/W	Start address point, 4-byte aligned The 2-LSB bit will be ignored

- Offset 0x50: PWM0_RDMA0_SET2

Signal	Bits	Default	R/W	Description
reserved				
cfg0_pwm_rdma0_set2	[7:0]	d0	R/W	[0] Set initial address 0: when xdma start, initial address is loaded from ptr 1: when xdma reset, initial address is loaded from ptr [3:1] Reserved [5:4] DMA data format 0: word, 4-bytes aligned 1: halfword, 2-bytes aligned 2/3: 1-byte aligned [7:6] Reserved

- Offset 0x58: PWM0_RDMA0_R0

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma0_r0	[31:0]	d0	R	Status of xdma next pointer address

- Offset 0x5C: PWM_RDMA0_R1

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma0_r1	[31:0]	d0	R	[0] xdma interrupt status [1] xdma error interrupt status [2] OR(xdma interrupt status, xdma error interrupt status) [9:8] xdma counter status, used for ping-pong buffer to check buffer status [16] xdma enable [17] xdma_done [19:18] xdma_status

- Offset 0x60: PWM0_RDMA1_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg0_pwm_rdma1_ctl0	[0]	d0	W1C	0 Start xdma for memory access 1: enable

- Offset 0x64: PWM0_RDMA1_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg0_pwm_rdma1_ctl1	[0]	d0	W1C	Software reset xdma 1: Reset

- Offset 0x68: PWM0_RDMA1_SET0

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma1_set0	[31:0]	d0	R/W	15:0 Segment size, 4-byte aligned 0: Illegal value Others: 1…65535 (Block size < Segment size) 31:16 Block size, 4-byte aligned 0: Single-shot mode Others: 1…65535 (Block size < Segment size)

- Offset 0x6C: PWM0_RDMA1_SET1

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma1_set1	[31:0]	d0	R/W	Start address point, 4-byte aligned The 2-LSB bit will be ignored

- Offset 0x70: PWM0_RDMA1_SET2

Signal	Bits	Default	R/W	Description
reserved				
cfg0_pwm_rdma1_set2	[7:0]	d0	R/W	[0] Set initial address 0: when xdma start, initial address is loaded from ptr 1: when xdma reset, initial address is loaded from ptr

				[3:1] Reserved [5:4] DMA data format 0: word, 4-bytes aligned 1: halfword, 2-bytes aligned 2/3: 1-byte aligned [7:6] Reserved

- Offset 0x78: PWM0_RDMA1_R0

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma1_r0	[31:0]	d0	R	Status of xdma next pointer address

- Offset 0x7C: PWM0_RDMA1_R1

Signal	Bits	Default	R/W	Description
cfg0_pwm_rdma1_r1	[31:0]	d0	R	[0] xdma interrupt status [1] xdma error interrupt status [2] OR(xdma interrupt status, xdma error interrupt status) [9:8] xdma counter status, used for ping-pong buffer to check buffer status [16] xdma enable [17] xdma_done [19:18] xdma_status

- Offset 0xA0: PWM0_INT_CLEAR

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
cfg0_pwm_int_clear	[7:0]	d0	W1C	0 Clear RDMA0 interrupt; 1: clear 1 Reserved 2 Clear RDMA1 interrupt; 1: clear 3 Reserved 4 Clear RSEQ done interrupt; 1: clear 5 Clear TSEQ done interrupt; 1: clear 6 Clear one/two sequence*cfgx_seqx_pcnt done interrupt; 1: clear 7 Clear register mode interrupt 1:clear

- Offset 0xA4: PWM0_INT_MASK

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
cfg0_pwm_int_mask	[7:0]	b1110000	R/W	0 Mask RDMA0 interrupt;

	0: No mask 1: Mask 1 Reserved 2 Mask RDMA1 interrupt; 0: No mask 1: Mask 3 Reserved 4 Mask RSEQ done interrupt; 0: No mask 1: Mask 5 Mask TSEQ done interrupt; 0: No mask 1: Mask 6 Mask one/two sequence*cfgx_seqx_pcnt done interrupt; 0: No mask 1: Mask 7 Mask register mode interrupt 0: No mask 1: Mask

- Offset 0xA8: PWM0_INT_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
cfg0_pwm_int_status	[7:0]	d0	R/W	0 Status of RDMA0 interrupt 1 Reserved 2 Status of RDMA1 interrupt 3 Reserved 4 Status of RSEQ done interrupt 5 Status of TSEQ done interrupt 6 Status of one/two sequence*cfgx_seqx_pcnt done interrupt 7 Status of register mode interrupt

- Offset 0xAC: PWM0_REG_DATA0

Signal	Bits	Default	R/W	Description
cfg0_pwm_reg_data0	[31:0]	d0	R/W	PWM register mode data0

- Offset 0xB0: PWM0_REG_DATA1

Signal	Bits	Default	R/W	Description
cfg0_pwm_reg_data1	[31:0]	d0	R/W	PWM register mode data1

23. I²S

The I²S is the most common interface to exchange audio data between chips. The I²S interface of CZ20 supports various sample rates and data formats which is applicable to most applications. It also supports the DMA function to transfer audio samples between memory and the I²S module without CPU intervention.

23.1. Block Diagram

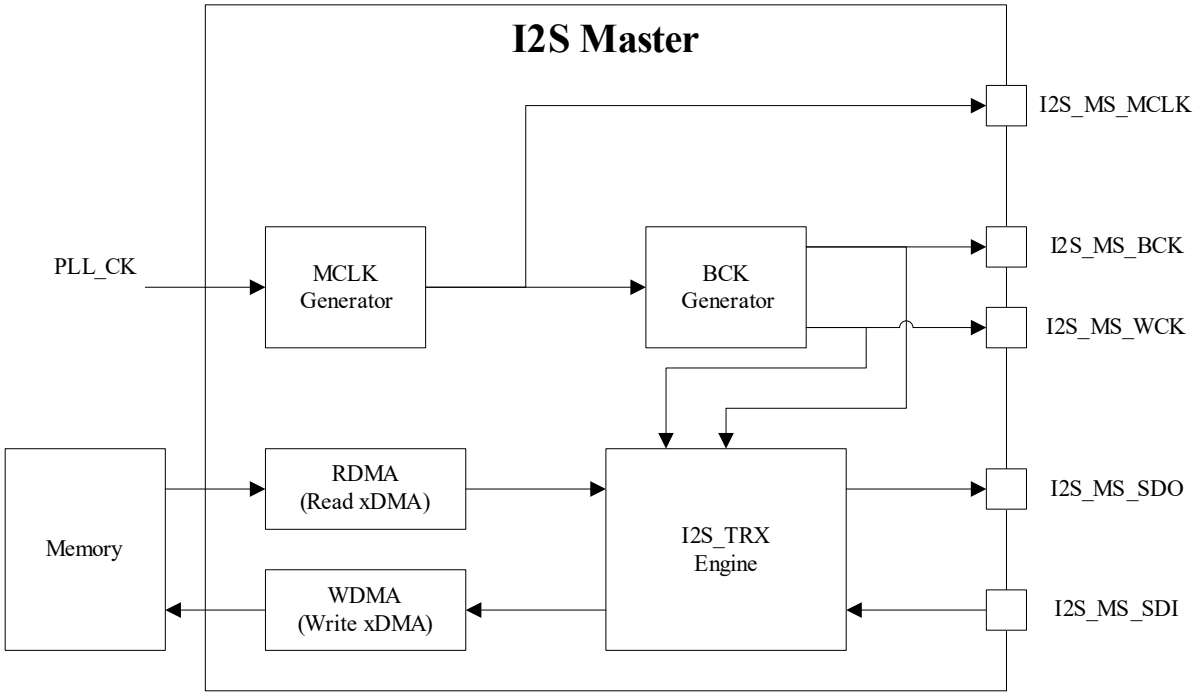


Figure 23-1 Block Diagram of I²S

23.2. Features

- Support master mode only
- Support MCLK
- Support I2S, Left Justified (LJ), and Right Justified (RJ) formats
- Support 8, 16, 32 and 48 KHz sample rates
- Support 16, 24 or 32-bit data length

23.3. Sample Rate Settings

The main parameters of I^{2}S sample rate configurations are:

- I2S_BCK = 2 * I2S_WCK * (Bit-Length of I2S_BCK per-channel)
- I2S_MCLK = I2S_BCK * Ratio = iMCLK / iMCLK_Divisor
  - iMCLK is the internal MCLK clock rate in the MCLK generator
  - I2S_MCLK is generated from iMCLK divided by iMCLK_Divisor
  - Ratio is selected from 2, 4 and 8

The above settings are summarized through Table 23-1 to Table 23-4.

Table 23-1 Suggestion Setting of I2S MCLK

I2S_MCLK_SET0. cfg_mck_isel	iMCLK	Comment
0	12.288M	PLL_CK = 32M, I2S_WCK = 48K
1	8.192M	PLL_CK = 32M, I2S_WCK = 8/16/32K
2	12.288M	PLL_CK = 48M, I2S_WCK = 48K
3	8.192M	PLL_CK = 48M, I2S_WCK = 8/16/32K
4	24.576M	PLL_CK = 64M, I2S_WCK = 48K
5	16.384M	PLL_CK = 64M, I2S_WCK = 8/16/32K

Table 23-2 The Divider Setting of I2S MCLK

I2S_MCLK_SET1. cfg_mck_div	iMCLK_Divisor	Comment
0	1	I2S_MCLK = iMCLK / 1
1	2	I2S_MCLK = iMCLK / 2
2	4	I2S_MCLK = iMCLK / 4
3	8	I2S_MCLK = iMCLK / 8
4	16	I2S_MCLK = iMCLK / 16
5	32	I2S_MCLK = iMCLK / 32

Table 23-3 The BCK to MCK Ratio Setting of I2S Interface

I2S_MS_SET0. cfg_bck_osr	Ratio	Comment
0	2	I2S_MCLK = I2S_BCK * Ratio
1	4	
2	8	

Table 23-4 Data Length Setting of I2S Interface

I2S_MS_SET0. cfg_bck_len	Bit-Length of BCK per channel	Comment
0	16	L/R with 16-BCLK
2	32	L/R with 32-BCLK

23.4. Interface Modes

The I²S supports three interface formats: I²S, LJ and RJ. Each I²S format can be configured to 16 or 32 BCK cycles per channel. The supported audio sample widths are 16, 24, or 32 bits.

If the audio sample width is less than the number of BCK cycles, the audio sample will insert zeros to adapt the number of BCK cycles. For example, to transfer the 24-bit audio data, BCK cycles shall be set to 32 and 8-bit zeros shall be padded.

If the audio sample width is larger than the number of BCK cycles, the audio sample width will truncate the LSB to adapt the number of BCK cycles

The interface waveforms of those formats are illustrated in Figure 23-2 to Figure 23-4.

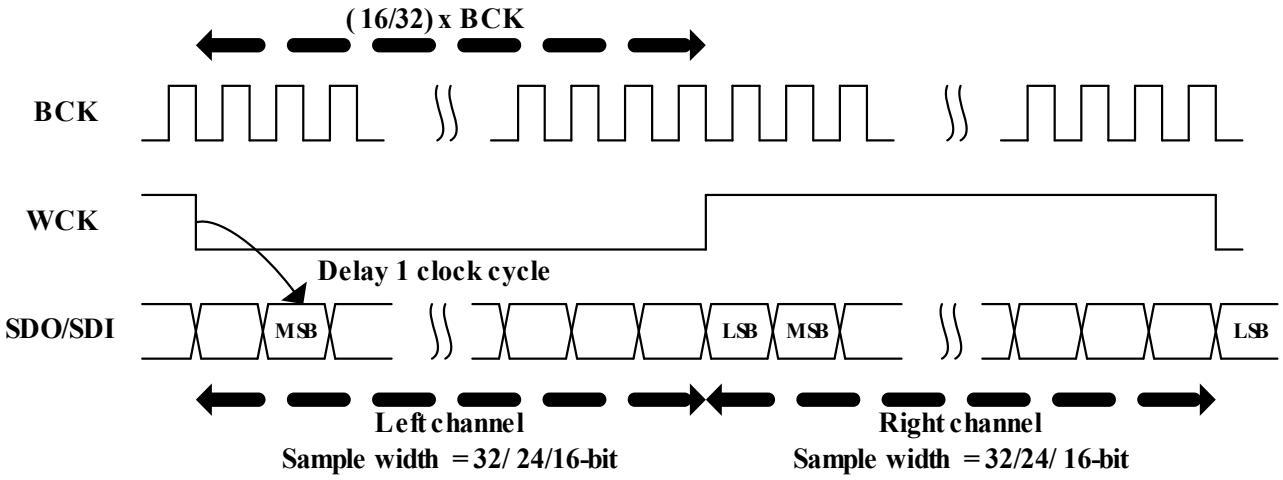


Figure 23-2 I²S Interface Waveform of I²S mode

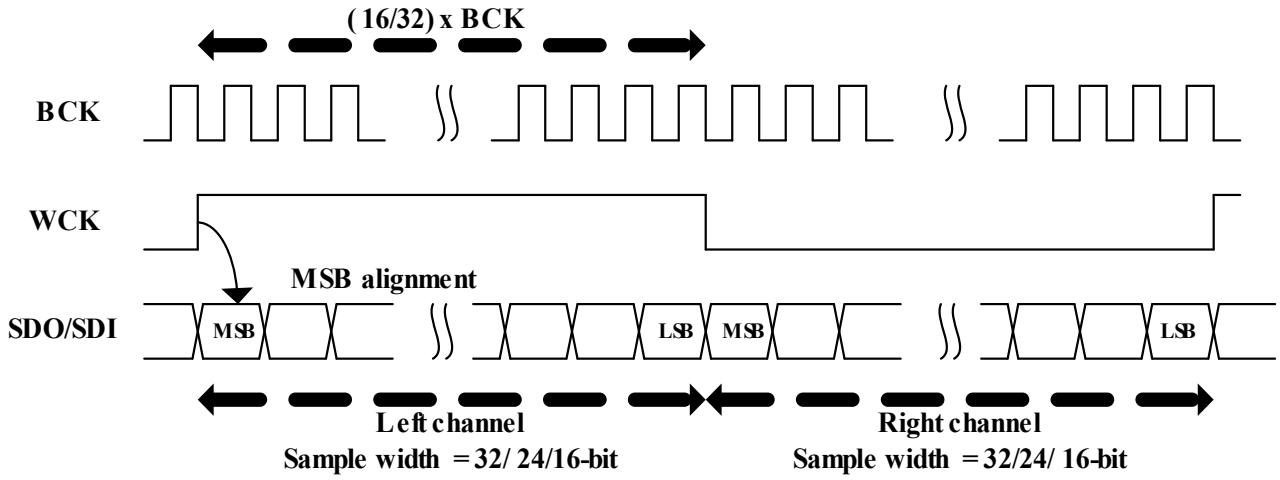


Figure 23-3 I²S Interface Waveform of Left Justified Mode

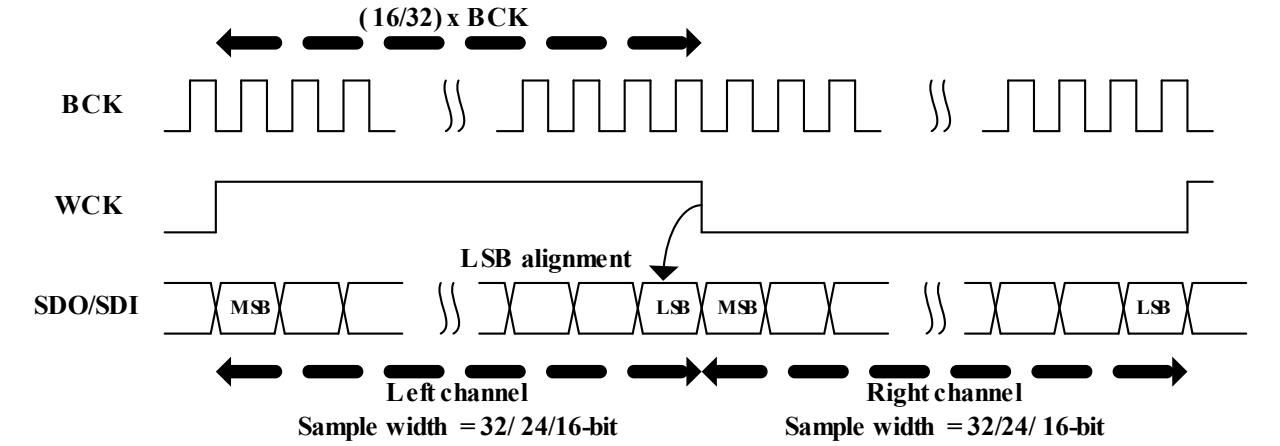


Figure 23-4 I²S Interface Waveform of Right Justified Mode

23.5. xDMA Supports

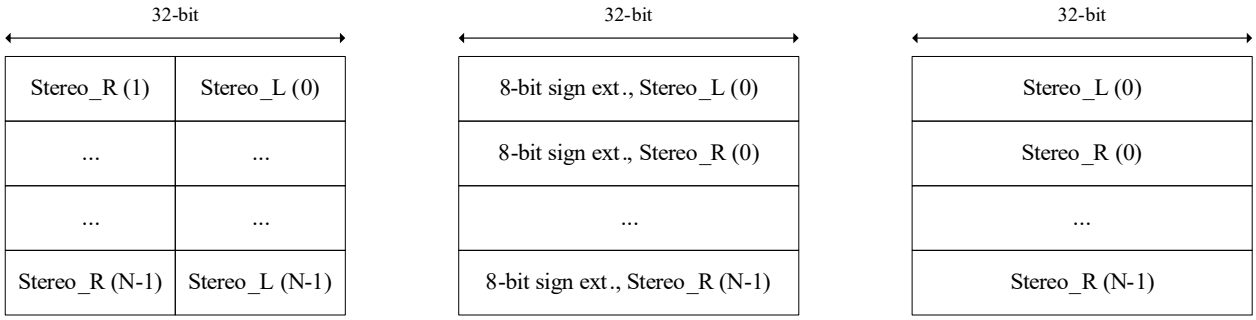
The I²S module has two xDMA with read type RDMA and two with write type WDMA. Each xDMA addressing mode shall be 4-byte align.

For the RDMA, the segment size and block size are configured from I2S_RDMA_SET0. The start address pointer for memory reading is configured from I2S_RDMA_SET1. The I2S_RDMA_CTL0 is used to trigger the start of RDMA for memory reading. The audio sample width and the channel definition stored in memory are configured from I2S_MS_SET0.

For the WDMA, the segment size and block size are configured from I2S_WDMA_SET0. The start address pointer for memory reading is configured from I2S_WDMA_SET1. The I2S_WDMA_CTL0 is used to trigger WDMA to start memory writing. The audio sample width and the channel definition stored in memory are configured from I2S_MS_SET0.

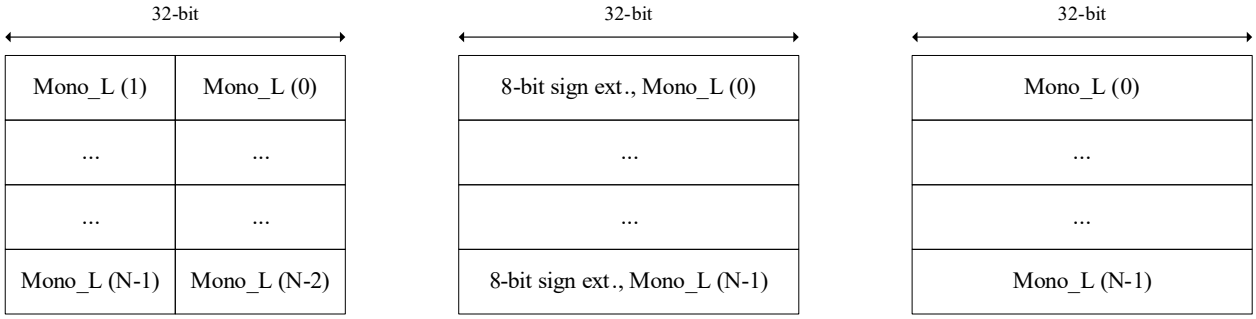
The audio sample format can have 16, 24 or 32-bits in the memory content. Each audio sample can also be defined as Stereo, Mono-L (Left) and Mono-R (Right) for I^2S Left and Right channel respectively.

Possible data structures in memory for stereo and mono audio samples are illustrated in Figure 23-5 and Figure 23-6 respectively.



- I2S_MS_SET0.cfg_tx/rx_wid = 0, 1, and 2 (16, 24 and 32-bit)
- I2S_MS_SET0.cfg_tx/rx_chn = 0 (Stereo audio sample)

Figure 23-5 Possible Data Structure in Memory of Stereo Audio Samples



- I2S_MS_SET0.cfg_tx/rx_wid = 0, 1, and 2 (16, 24 and 32-bit)
- I2S_MS_SET0.cfg_tx/rx_chn = 1 (Stereo audio sample)

Figure 23-6 Possible Data Structure in Memory of Mono Audio Samples

23.6. Registers

- Offset 0x00: I2S_MS_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
cfg_i2s_ck_free	[9:8]	b00	R/W	[8]: Enable i2s register clock free run 0: Auto gated 1: Free run [9]: Enable i2s engine clock free run 0: Auto gated 1: Free run
Reserved	[7:3]			
cfg_pdm_tx_en_mux	[2]	b0	R/W	PDM tx en mux
cfg_mck_ena	[1]	b0	R/W	Enable MCLK 0: disable 1: enable
cfg_i2s_ena	[0]	b0	R/W	Enable I2S 0: disable 1: enable

- Offset 0x04: I2S_MS_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_i2s_rst	[0]	b0	W1C	Reset I2S 1: reset

- Offset 0x08: I2S_MCLK_SET0

Signal	Bits	Default	R/W	Description
reserved	[31:3]			
cfg_mck_isel	[2:0]	d0	R/W	Internal MCLK rate selection 0: average 12.288M when PLL is 32M 1: average 8.192M when PLL is 32M 2: average 12.288M when PLL is 48M 3: average 8.192M when PLL is 48M 4: average 24.576M when PLL is 64M 5: average 16.384M when PLL is 64M

- Offset 0x0C: I2S_MCLK_SET1

Signal	Bits	Default	R/W	Description
reserved	[31:4]			
cfg_mck_div	[3:0]	d0	R/W	Output MCLK divider 0: internal MCLK divided by 1

	1: internal MCLK divided by 2 2: internal MCLK divided by 4 3: internal MCLK divided by 8 4: internal MCLK divided by 16 5: internal MCLK divided by 32 others: internal MCLK divided by 32

- Offset 0x10: I2S_MCLK_SET2

Signal	Bits	Default	R/W	Description
reserved	[31:24]			
cfg_mck_int	[23:16]	0x02	R/W	MCLK divider integer value
cfg_mck_fra	[15:0]	0xCAAA	R/W	MCLK divider fractional value

- Offset 0x14: I2S_MS_SET0

Signal	Bits	Default	R/W	Description
reserved	[31:16]			
cfg_rxd_chn	[15:14]	0x0	R/W	Rx sample format in xdma 0: stereo, L/R 1: mono, L 2: mono, R 3: reserved
cfg_txd_chn	[13:12]	0x0	R/W	Tx sample format in xdma 0: stereo, L/R 1: mono, L 2: mono, R 3: reserved
cfg_rxd_wid	[11:10]	0x0	R/W	Sample width for I2S received sample 0: 16-bit 1: 24-bit 2: 32-bit 3: 32-bit
cfg_txd_wid	[9:8]	0x0	R/W	Sample width for I2S transmitted sample 0: 16-bit 1: 24-bit 2: 32-bit 3: 32-bit
cfg_bck_len	[7:6]	0x0	R/W	Bit Length of BCLK per channel 0: 16 1: 24 (proprietary I2S) 2: 32 3: 32
cfg_i2s_fmt	[5:4]	0x0	R/W	I2S format 0: LJ mode 1: RJ mode 2: I2S mode 3: I2S mode
cfg_i2s_mod	[3:2]	0x0	R/W	I2S transceiver mode 0: TxRx 1: Tx only

				2: Rx only 3: TxRx
cfg_bck_osr	[1:0]	0x0	R/W	Ratio of MCLK and BCLK 0: 2 1: 4 2: 8 3: 8

- Offset 0x40: I2S_RDMA_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_i2s_rdma_ctl0	[0]	d0	W1C	[0] Start xdma for memory access 1: enable

- Offset 0x44: I2S_RDMA_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_i2s_rdma_ctl1	[0]	d0	W1C	Software reset xdma 1: Reset

- Offset 0x48: I2S_RDMA_SET0

Signal	Bits	Default	R/W	Description
cfg_i2s_rdma_set0	[31:0]	d0	R/W	[15:0] Segment size, 4-byte aligned 0: illegal value 1...32768: normal [31:16] Block size, 4-byte aligned 0: single mode 1..32768: normal

- Offset 0x4C: I2S_RDMA_SET1

Signal	Bits	Default	R/W	Description
cfg_i2s_rdma_set1	[31:0]	d0	R/W	Start address point, word (4-byte) aligned The 2-LSB bit will be ignored

- Offset 0x50: I2S_RDMA_SET2

Signal	Bits	Default	R/W	Description
reserved	[31:8]			

cfg_i2s_rdma_set2	[7:0]	d0	R/W	[0] Set initial address 0: when xdma start, initial address is loaded from ptr 1: when xdma reset, initial address is loaded from ptr [3:1] Reserved [5:4] DMA data format 0: word, 4-bytes aligned 1: halfword, 2-bytes aligned 2,3: 1-byte aligned [7:6] Reserved

- Offset 0x58: I2S_RDMA_R0

Signal	Bits	Default	R/W	Description
cfg_i2s_rdma_r0	[31:0]	d0	R	Status of xdma next pointer address

- Offset 0x5C: I2S_RDMA_R1

Signal	Bits	Default	R/W	Description
cfg_i2s_rdma_r1	[31:0]	d0	R	[0] xdma interrupt status [1] xdma error interrupt status [2] OR(xdma interrupt status, xdma error interrupt status) [9:8] xdma counter status, used for ping-pong buffer to check buffer status [16] xdma enable [17] xdma_done [19:18] xdma_status

- Offset 0x60: I2S_WDMA_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_i2s_wdma_ctl0	[0]	d0	W1C	[0] Start xdma for memory access 1: enable

- Offset 0x64: I2S_WDMA_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_i2s_wdma_ctl1	[0]	d0	W1C	Software reset xdma 1: Reset

- Offset 0x68: I2S_WDMA_SET0

Signal	Bits	Default	R/W	Description
cfg_i2s_wdma_set0	[31:0]	d0	R/W	[15:0] Segment size, 4-byte aligned 0: illegal value 1...32768: normal [31:16] Block size, 4-byte aligned 0: single mode 1..32768: normal

- Offset 0x6C: I2S_WDMA_SET1

Signal	Bits	Default	R/W	Description
cfg_i2s_wdma_set1	[31:0]	d0	R/W	Start address point, word (4-byte) aligned The 2-LSB bit will be ignored

- Offset 0x70: I2S_WDMA_SET2

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
cfg_i2s_wdma_set2	[7:0]	d0	R/W	[0] Set initial address 0: when xdma start, initial address is loaded from ptr 1: when xdma reset, initial address is loaded from ptr [3:1] Reserved [5:4] DMA data format 0: word, 4-bytes aligned 1: halfword, 2-bytes aligned 2,3: 1-byte aligned [7:6] Reserved

- Offset 0x78: I2S_WDMA_R0

Signal	Bits	Default	R/W	Description
cfg_i2s_wdma_r0	[31:0]	d0	R	Status of xdma next pointer address

- Offset 0x7C: I2S_WDMA_R1

Signal	Bits	Default	R/W	Description
cfg_i2s_wdma_r1	[31:0]	d0	R	[0] xdma interrupt status [1] xdma error interrupt status [2] OR(xdma interrupt status, xdma error interrupt status) [9:8] xdma counter status, used for ping-pong buffer to check buffer status [16] xdma enable [17] xdma_done [19:18] xdma_status

- Offset 0xA0: I2S_INT_CLEAR

Signal	Bits	Default	R/W	Description
reserved	[31:4]			
cfg_i2s_int_clear	[3:0]	d0	W1C	[0] Clear I2S_RDMA interrupt 1: clear [1] Reserved

				[2] Clear I2S_WDMA interrupt 1: clear [3] Reserved

- Offset 0xA4: I2S_INT_MASK

Signal	Bits	Default	R/W	Description
reserved	[31:0]			
cfg_i2s_int_mask	[3:0]	d0	R/W	[0] Mask I2S_RDMA interrupt 0: No mask, 1: Mask [1] Reserved [2] Mask I2S_WDMA interrupt 0: No mask, 1: Mask [3] Reserved

- Offset 0xA8: I2S_INT_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:0]			
cfg_i2s_int_status	[3:0]	d0	R/W	[0] Status of I2S_RDMA interrupt [1] Reserved [2] Status of I2S_WDMA interrupt [3] Reserved

24. IRM

The infrared remote, simple and low-cost technology is very often used in today's homes. The controller provides several of the most frequently used infrared protocols. It is a dedicated peripheral that allows the generation of infrared waveforms for transmitting infrared signals with minimal software overheads.

24.1. Block Diagram

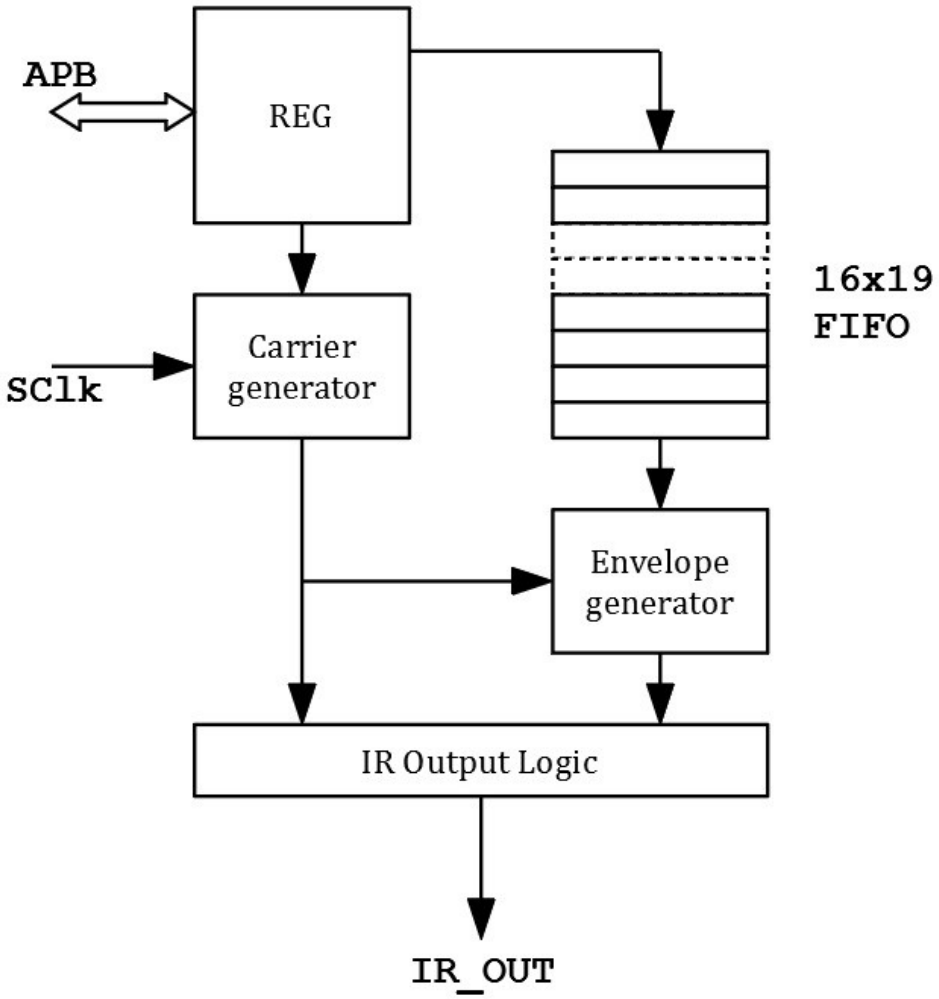


Figure 24-1 Block Diagram of Infra-red Modulator

24.2. Features

- Support NEC, RC5, RC6, RCMM and SIRC protocols and all other existing protocols for common TV or STB appliances
- Configurable carrier generator module
- Various modes available for generation of different infrared modes

24.3. Functional Description

The InfraRed Modulator provides the specific pulse waveforms that are applied to an InfraRed diode. This allows transmission of data over an InfraRed link to another system or device. InfraRed (IR) technology employs the use of a carrier waveform that is composed of short pulses, usually with a duty cycle of less than 50%. This reduces the "ON" time of the InfraRed diode and, thus, lowers the overall system battery drain.

The IR Modulator offers the advantage that the end user may then control all of his appliances with a single remote-controller. It provides a variety of programming options that support all existing protocols.

The InfraRed Modulator (IR Modulator) generates specific waveforms that are applied to an IR diode. It supports NEC, RC5, RC6, RCMM and SIRC protocols. Generated waveforms are the result of an amplitude Modulator of two signals:

- the carrier: this is a square signal with programmable frequency and duty cycle. It doesn't carry any information. It reduces power consumption by reducing "ON" time of the InfraRed diode. This signal is filtered out by receiver.
- the envelope: this signal carries information. It consists of a succession of high and low level of different duration.

The IR Modulator produces repetitive waveform with predefined frequency and duty cycle with little involvement from software. A commonly used IR signal can be described as an AM (amplitude modulated) signal, where an envelope timing is defined as a multiple of a base time unit, which is generally different for every brand of video and audio equipment.

A carrier frequency is defined by settings in the IR Modulator carrier configuration register (**CARRIER**). Two fields are used to determine the relative "high" and "low" time of the carrier.

$$Carrier_{freq} = \frac{IR\_SCLK_{freq}}{Base_{cnt} \times (LOW_{cnt} + HIGH_{cnt} + 2)}$$

Equation 24-1 Calculation of IRM Carrier Frequency

A duty cycle (typically selected between 25% and 50%) is given as the ratio:

$$DutyCycle = \frac{HIGH_{cnt} + 1}{LOW_{cnt} + HIGH_{cnt} + 2}$$

Equation 24-2 Calculation of IRM Duty Cycle

The envelope signal is controlled by the content of FIFO. FIFO size is 16x19 with the content of {ENV_LAST, ENV_INT, ENV_MARK, ENV_CNT[15:0]}. Before IR Modulator starts, fill the envelope timer count in ENV_CNT[15:0] with envelope level indicator in ENV_MARK for each period and mark ENV_LAST bit for the last envelope.

The IRM can be used in Normal Mode or Auto Mode according to the register setting, **OP_MODE**.

24.3.1. IRM Normal Mode

In this mode, if IR Modulator encounters a data in FIFO with the bit ENV_LAST=1, then it stops right after transmitting the corresponding envelope even if the FIFO is not empty. This is illustrated in Figure 24-2. The software can restart IR Modulator to make it transmit the following data present in FIFO, by writing 1 to the **IR_START** register.

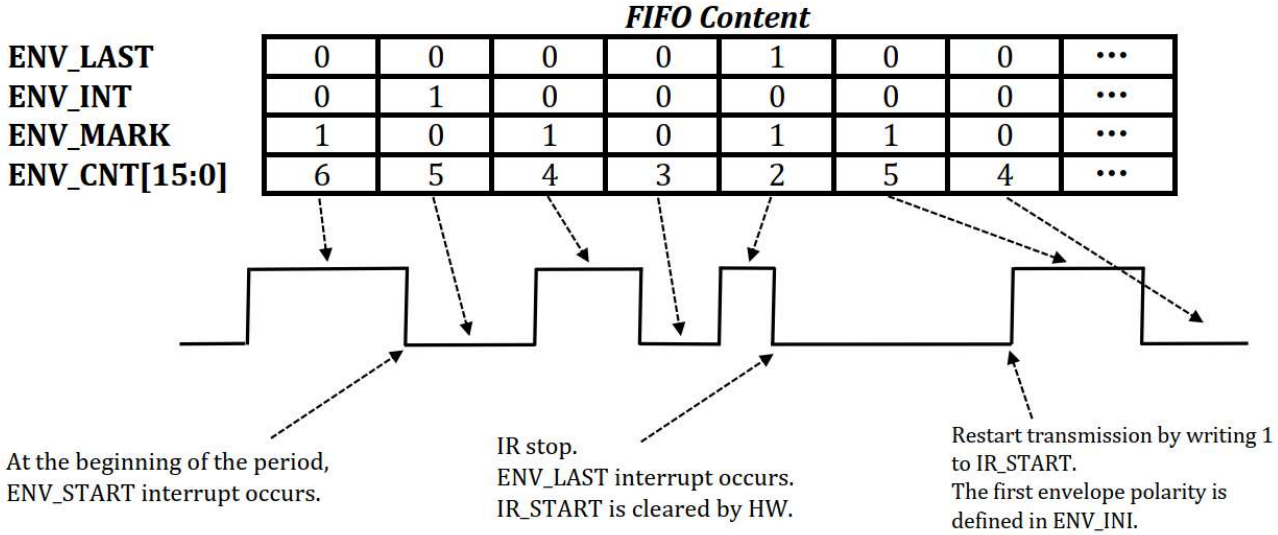


Figure 24-2 Normal Mode Operation of IRM

Before restarting, it is possible to add data in FIFO. When IR Modulator is stopped, envelope signal is always 0.

If there is not data in FIFO with ENV_LAST=1, IR Modulator transmits all data present in FIFO and stop when the FIFO becomes empty (this event generates **FIFO_UFL** interrupt).

24.3.2. IRM Auto Mode

In this mode, IR Modulator transmit all the data present in FIFO regardless of the ENV_LAST bit field. It will stop only when FIFO becomes empty (this event may generate **FIFO_UFL** interrupt). This is shown in Figure 24-3.

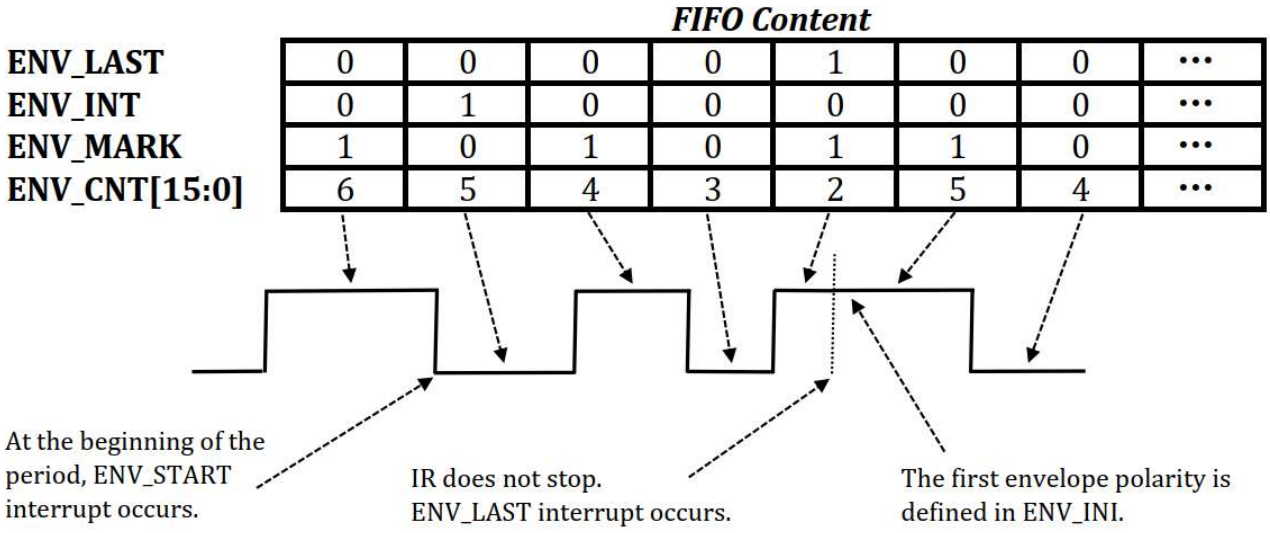


Figure 24-3 Auto Mode Operation of IRM

24.3.3. Programming Examples

In this example, the carrier frequency is set to 38KHz.

Enable sequences:

- Set IR_ENA = 1 to enable the IR Modulator.
- Set INT_ENA = 1 to enable the interrupt if the interrupt is preferred.

- Configure OP_MODE.
- Configure OUT_MODE. Uses the "AND" if the LED does not invert the signal (LED is on when signal is at high level), otherwise uses "NAND" function.

Program the carrier:

- Set CAR_LOW_CNT = 1 and CAR_HIGH_CNT = 0.
- Set CAR_BASE_CNT = 280. This value is calculated from Equation 24-1.

Fill the FIFO and start the transmission:

- Fill the ENV_CNT with the envelope counter value and ENV_MARK with the envelope level according to the carrier period.
- Set ENV_LAST bit for the last envelope if any.
- Set ENV_INT bit to get the interrupt of ENV_START_INT for the right time of filling more envelopes.
- Set IR_START=1 to start the transmission

When ENV_START_INT interrupt occurs:

- Fill the FIFO with remaining envelopes.
- Set ENV_LAST bit for the last envelope if any.

When ENV_LAST interrupt occurs:

- Software driver has to consider the mute time of the IR protocol and then refill the FIFO with new data if any.

24.4. Registers

- Offset 0x00: IR_CONF

Signal	Bits	Default	R/W	Description
reserved	[31:]6			
CAR_INI	[5]	0	R/W	Initial Carrier Value 0b - Carrier starts with '0' (the carrier is low during CAR_LOW_CNT and high during CAR_HIGH_CNT) 1b - Carrier starts with '1' (the carrier is high during CAR_LOW_CNT

				and low during CAR_HIGH_CNT)
NO_CAR	[4]	0	R/W	No Carrier 0b - Normal. IR Modulator output = envelope + carrier 1b - Carrier is inhibited. IR Modulator output = envelope only
OUT_MODE	[3:2]	0	R/W	Output Logic Function 00b - envelope AND carrier 01b - envelope OR carrier 10b - envelope NAND carrier 11b - envelope NOR carrier
OP_MODE	[1]	0	R/W	Operation mode 0b - Normal Mode. IR Modulator will stop when it encounters an envelope with FIFO_IN[ENV_LAST]= 1. 1b - Auto Mode. IR Modulator will transmit all envelopes and stop only when FIFO becomes empty. FIFO_IN[ENV_LAST] bit only generates an interrupt but doesn't stop.
reserved	[0]			

- Offset 0x04: IR_CARRIER

Signal	Bits	Default	R/W	Description
reserved	[31:24]			
CAR_HIGH_CNT	[23:20]	0	R/W	Carrier high level duration = (CAR_HIGH_CNT + 1) * CAR_BASE_CNT.
CAR_LOW_CNT	[19:16]	0	R/W	Carrier low level duration = (CAR_LOW_CNT + 1) * CAR_BASE_CNT.
CAR_BASE_CNT	[15:0]	0x127	R/W	The divisor for target carrier frequency from IR source clock (32 MHz). Value 0x0 is equivalent to 0x1.

- Offset 0x08: IR_FIFO_IN

Signal	Bits	Default	R/W	Description
reserved	[31:15]			
ENV_LAST	[18]	0	W	Last Envelope 0b - IR Modulator loads the next envelope when this envelope finishes. 1b - IR Modulator stops and generates an interrupt when this envelope is completely transmitted. If IR_CONF[OP_MODE] = 1, IR Modulator only generates an interrupt.
ENV_INT	[17]	0	W	Envelope Interrupt Generate an interrupt when starting emission of the envelope. 0b - Not generate interrupt 1b - Generate interrupt
ENV_MARK	[16]	0	W	Envelope Mark 0b - the envelope period is a space 1b - the envelope period is a mark
ENV_CNT	[15:0]	0	W	Envelope Duration Envelope duration expressed in carrier period number. Value 0x0000 has the same behavior as the value 0x0001.

- Offset 0x0C: IR_FIFO_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:7[			
FIFO_EMPTY	[6]	1b	R	IR Modulator FIFO Empty Flag 0b - FIFO is not empty 1b - FIFO is empty (FIFO level = 000000)
FIFO_FULL	[5]	0b	R	IR Modulator FIFO Full Flag 0b - FIFO is not full 1b - FIFO is full (FIFO level = 100000)
FIFO_LVL	[4:0]	0	R	Current IR Modulator FIFO Level 00000b - Empty 00001b - 1 entry ..... 10000b - 16 entries/FIFO is full 10001b-11111b - Not valid

- Offset 0x10: IR_CMD

Signal	Bits	Default	R/W	Description
Reserved	[31:4[			
FIFO_RST	[3]	0	R/W	Reset IR Modulator FIFO.This bit is self-clearing. 0b - No effect 1b - Reset FIFO. IR Modulator FIFO is completely re-initialized, all data in the FIFO are erased.
IR_START	[2]	0	R/W	Start IR Modulator. This bit is self-clearing. 0b - No effect 1b - Start transmission. Before setting this field, the Modulator must be enabled (the bit field IR_ENA must be set first).
IR_DIS	[1]	0	R/W	Disable IR Modulator. This bit is self-clearing. 0b - No effect 1b - Disable IR Modulator. The transmission of envelopes is immediately stopped. The FIFO is not reinitialized (the content of the FIFO is conserved).
IR_ENA	[0]	0	R/W	Enable IR Modulator Unit. This bit is self-clearing. 0b - No effect 1b - Enable IR Modulator

- Offset 0x14: IR_INT_STATUS

Signal	Bits	Default	R/W	Description
Reserved	[31:3]			
FIFO_UFL_INT	[2]	0	R	IR Modulator FIFO Underflow (FIFO_UFL) Interrupt This interrupt occurs when IR Modulator tries to transmit data but the FIFO is empty. 0b - Interrupt is not pending 1b - Interrupt is pending
ENV_LAST_INT	[1]	0	R	Envelop Last (ENV_LAST) Interrupt The interrupt occurs when IR Modulator has finished transmitting an envelope with FIFO_IN[ENV_LAST] =1. 0b - Interrupt is not pending

				1b - Interrupt is pending
ENV_START_INT	[0]	0	R	Envelop Start (ENV_START) Interrupt This interrupt occurs when IR Modulator has started to transmit an envelope with FIFO_IN[ENV_INT] = 1 0b - Interrupt is not pending 1b - Interrupt is pending

- Offset 0x18: IR_INT_ENA

Signal	Bits	Default	R/W	Description
Reserved	[31:3]			
FIFO_UFL_ENA	[2]	0	R/W	Enable/Disable FIFO_UFL Interrupt 0b - Disable FIFO_UFL interrupt 1b - Enable FIFO_UFL interrupt
ENV_LAST_ENA	[1]	0	R/W	Enable/Disable ENV_LAST Interrupt 0b - Disable ENV_ LAST interrupt 1b - Enable ENV_ LAST interrupt
ENV_START_ENA	[0]	0	R/W	Enable/Disable ENV_START Interrupt 0b - Disable ENV_START interrupt 1b - Enable ENV_START interrupt

- Offset 0x1C: IR_INT_CLR

Signal	Bits	Default	R/W	Description
Reserved	[31:3]			
FIFO_UFL_ CLR	[2]	0	W1C	Clear FIFO_UFL interrupt This bit is self-clearing. 0b - No effect 1b - Clear FIFO_UFL interrupt
ENV_LAST_ CLR	[1]	0	W1C	Clear ENV_LAST interrupt This bit is self-clearing. 0b - No effect 1b - Clear ENV_ LAST interrupt
ENV_START_CLR	[0]	0	W1C	Clear ENV_START interrupt This bit is self-clearing. 0b - No effect 1b - Clear ENV_START interrupt

25. AUX ADC

The AUX ADC is a successive approximation (SAR) analog to digital converter. It can be used to measure voltages from external analog inputs (AIOs) or from the internal signals like the temperature sensor and the battery. The AIOs are shared with GPIOs and can be configured as single or differential inputs to the AUX ADC. Various sampling modes are provided and the DMA mode is also supported via the xDMA.

25.1. Block Diagram

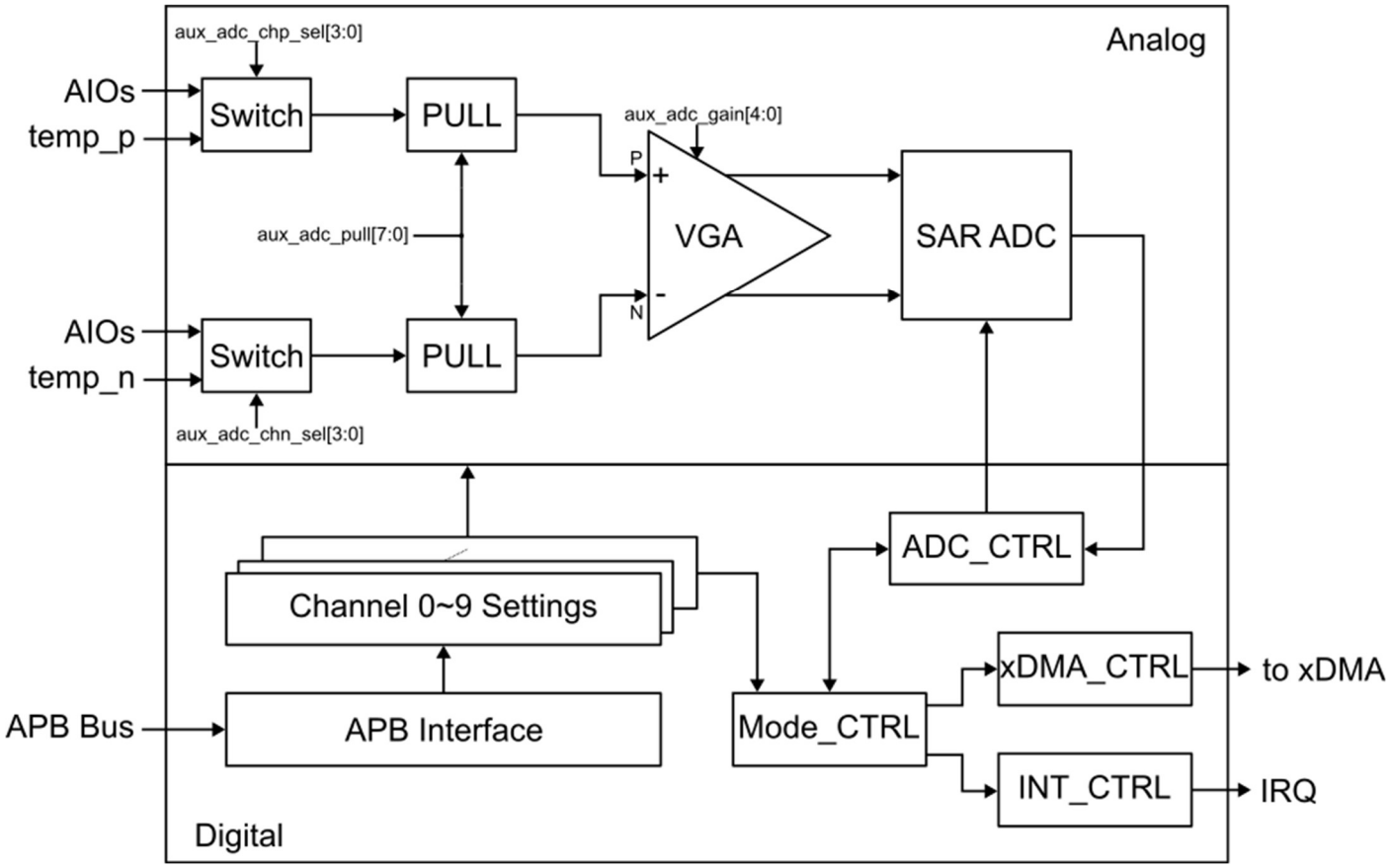


Figure 25-1 Block Diagram of AUX ADC

25.2. Features

- Support VGA.
- Support 8,10,12-bit resolutions.
- Support the 14-bit resolution with oversampling.
- Support single or differential AIO inputs.
- Support the one-shot, timer, and scan modes.
- Support up to 10 logical channels
- Support xDMA.

25.3. Functional Description

The AUX ADC consists both the analog and the digital parts as shown in Figure 25-1.

In the analog part, the input switch selects from AIOs or temperature sensor inputs (temp_p and temp_n) and passes them to the VGA. The VGA can amplify the input signal to the proper operating ranges of the SAR ADC.

Two PULL controls are placed before the input ports of VGA, P and N and are used to measure voltages of Battery or Ground.

The digital part generates control signals to the analog circuits with the correct timing and send the ADC output data to the memory via xDMA. It supports up to 10 logical channels. Each logical channel has its own register sets for both analog and digital parts and can be hooked to any physical channel like AIOs or the temperature sensor. If more than one logical channels are enabled, they will be served in sequence.

The mode controller, Mode_CTRL, defines the operation mode for different usage scenarios. Three modes are supported and they are the one-shot, the timer and the scan mode. The ADC results can be configured to 8, 10, 12 bits. A higher resolution of 14 bits with the oversampling is also supported. Results are all converted to the 16-bit unsigned integer with extended zeros to the MSB.

25.3.1. One-shot Mode

One shot mode requires that only one channel is enabled. Configure register SADC_SET0.[0] to set the trigger source from the control register. The MCU enables the controller via SADC_CTL0 and

triggers the mode controller via SADC_CTRL2 by single shot. It sends a start request to the digital controller. Whenever the digital controller receives start request, it begins sampling the ADC results according to the channel configurations.

25.3.2. Timer Mode

Timer mode also requires that only one channel is enabled. Configure the register SADC_SET0.[0] to set the trigger source from the timer. The MCU enables the controller via SADC_CTL0 and triggers periodically by the timer. It sends a start request to the digital controller periodically. Whenever the digital controller receives a start request, it begins sampling the ADC resultss according to the channel configurations.

25.3.3. Scan Mode

The scan mode is used when more than one channel is enabled. Configure the register SADC_SET0.[0] to set the trigger source from the control register. The MCU enables the controller via SADC_CTL0 and triggers the mode controller SADC_CTRL2 by one-shot. It sends a start request to the digital controller for enabling the channels one by one according to the highest priority from CH0 to CH9. Whenever the digital controller receives a start request, it begins to sample the ADC results according to the channel gain, acquisition time and end of delay time.

25.3.4. Input Switch

From Figure 25-1 we can see that the input switches are controlled by aux_adc_chp_sel[3:0] and aux_adc_chn_sel[3:0] while the PULL control are controlled by aux_adc_pull[7:0]. Those control signals are come from the SADC_PNSEL_CHx register where x is 0~9 depending on which channel is served. The mapping of those control signals are listed in Table 25-1.

Table 25-1 Mapping of Input Control Signals of AUX ADC

Control Signals	Register	Description
aux_adc_chp_sel[3:0]	SADC_PNSEL_CHx[3:0]	Select P channel inputs of AUX ADC. Depends on the package, some AIOs may not be available. 0000b Select AIO0 0001b Select AIO1 0010b Select AIO2 0011b Select AIO3 0100b Select AIO4 0101b Select AIO5 0110b Select AIO6 0111b Select AIO7 1000b Select Temperature Sensor P output (temp_p) 1001b Reserved.

		1010b Select Battery voltage. Others Disabled.
aux_adc_chn_sel[3:0]	SADC_PNSEL_CHx[7:4]	Select N channel inputs of AUX ADC. Depends on the package, some AIOs may not be available. 0000b Select AIO0 0001b Select AIO1 0010b Select AIO2 0011b Select AIO3 0100b Select AIO4 0101b Select AIO5 0110b Select AIO6 0111b Select AIO7 1000b Select Temperature Sensor N output (temp_n) 1001b Reserved. 1010b Select Battery voltage. Others Disabled.
aux_adc_pull[7:0]	SADC_PNSEL_CHx[23:16]	aux_adc_pull[0] : Pull the P input to high aux_adc_pull[1] : Pull the P input to low aux_adc_pull[2] : Pull the N input to high aux_adc_pull[3] : Pull the N input to low aux_adc_pull[4] : Short the P input to VDD aux_adc_pull[5] : Short the P input to GROUND aux_adc_pull[6] : Short the N input to VDD aux_adc_pull[7] : Short the N input to GROUND

To understand these more clearly, Table 25-2 lists several common AUX ADC scenarios and their settings. Each scenario does not restrict the use of logical channels and you can use anyone if available.

Table 25-2 AUX ADC Scenarios and Their Input Settings

Scenarios	Description	Register Settings
Single AIO	Logical channel 1 is used. AIO0 connected to the P input.	SADC_PNSEL_CH1[3:0] = 0000b SADC_PNSEL_CH1[7:4] = 1111b SADC_PNSEL_CH1[23:16] = 00000000b
Differential AIO	Logical channel 0 is used. AIO2 connected to the P input. AIO4 connected to the N input.	SADC_PNSEL_CH0[3:0] = 0010b SADC_PNSEL_CH0[7:4] = 0100b SADC_PNSEL_CH0[23:16] = 00000000b
Temperature Sensor	Logical channel 9 is used.	SADC_PNSEL_CH9[3:0] = 1000b SADC_PNSEL_CH9[7:4] = 1000b SADC_PNSEL_CH9[23:16] = 00000000b
Battery Voltage	Logical channel 3 is used.	SADC_PNSEL_CH3[3:0] = 1010b SADC_PNSEL_CH3[7:4] = 1010b SADC_PNSEL_CH3[23:16] = 00010001b

25.3.5. Oversampling

The raw data from the ADC has the resolution of 12 bits. It can achieve a higher 14-bit resolution with

oversampling. By setting SADC_SET1[11:8] to a non-zero value, the ADC_CTRL will sample ADC results by OSR times and average by the OSR value. The OSR value ranges from 2 to 256. The SADC_SET1[3:0] controls the bit resolution sending to xDMA.

25.3.6. Result Monitor

Each channel has its own monitor for value detection and is defined in SADC_THD_CH0~9. The lower part of SADC_THD_CH0[13:0] defines the low threshold value and if the SADC result is lower than this value, the monitor will send an interrupt. The higher bits of SADC_THD_CH0[29:16] define the high threshold value and if the ADC result is higher than this value, the monitor will send an interrupt. The two interrupts are different.

25.3.7. xDMA Supports

The AUX ADC supports WDMA only because the data is always from the AUX ADC to the memory. Regardless of the resolution settings, ADC data are all converted to the 16-bit unsigned integer with extended zeros to the MSB before writing to the memory. Thus, the block size, segment size and address of WDMA shall always be 2-byte align.

25.3.8. Interrupts

The AUX ADC has several interrupt sources. Table 25-3 lists the source of each interrupt status bit.

Table 25-3 Interrupt Sources of AUX ADC

Interrupt status bit fields	Description
SADC_INT_STATUS[0]	Status of WDMA interrupt, issued whenever the sample written to memory exceeds the block size
SADC_INT_STATUS[1]	Used for WDMA error
SADC_INT_STATUS[2]	When sadc_dig_ctrl finishes one SADC sampling, it will issue a one-time interrupt. If oversampling is enabled, a number of interrupts are asserted according to the oversampling ratio.
SADC_INT_STATUS[3]	When sadc_dig_ctrl finishes one SADC sampling or SADC sampling with oversampling, it will issue one-time interrupt.
SADC_INT_STATUS[4]	For one-shot mode, it will issue a one-time interrupt whenever the mode operation for one channel is finished. For timer mode, it will issue interrupts periodically whenever the mode operation for one channel is finished and triggered by the timer for the next time. For scan mode, it will issue a one-time interrupt whenever the mode operation for two channels or more are finished.
SADC_INT_STATUS[17:8]	From LSB to MSB is status of the monitor low threshold value interrupt for CH0 to CH9.
SADC_INT_STATUS[27:18]	From LSB to MSB is status of the monitor high threshold value interrupt for CH0 to CH9.

25.4. Registers

25.4.1. Common Registers

- Offset 0x000: SADC_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_sadc_ck_free	[9:8]	b00		[8]: Enable SADC register clock free run 0: Auto gated 1: Free run [9]: Enable SADC engine clock free run 0: Auto gated 1: Free run
reserved	[7:3]			
cfg_sadc_ldo_ena	[2]	b0	R/W	Enable SADC LDO, only work when cfg_sadc_tst[2] = 1 0: disable 1: enable
cfg_sadc_vga_ena	[1]	b0	R/W	Enable SADC VGA, only work when cfg_sadc_tst[2] = 1 0: disable 1: enable
cfg_sadc_ena	[0]	b0	R/W	Enable SADC 0: disable 1: enable

- Offset 0x004: SADC_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:9]			
cfg_sadc_afifo_rst	[8]	b0	W1C	Reset SADC AFIFO (Default is no reset) 1: Reset
reserved	[7:1]			
cfg_sadc_rst	[0]	b0	W1C	Reset SADC 1: reset

- Offset 0x008: SADC_CTL2

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_sadc_start	[0]	b0	W1C	Software start SADC 1: start

- Offset 0x00C: SADC_SET0

Signal	Bits	Default	R/W	Description
cfg_sadc_tmr_ckdiv	[31:16]	d0	R/W	Timer rate configuration Timer rate = timer clock / (cfg_sadc_ckdiv+1) where cfg_sadc_ckdiv = 2…65535

reserved	[15:7]			
cfg_sadc_dbg_sel	[6:3]	d0	R/W	Debug monitor selection
cfg_sadc_afifo_ckpsel	[2]	b0	R/W	AFIFO Clock phase selection 0: Write AFIFO at rising edge of clock 1: Write AFIFO at falling edge of clock
cfg_sadc_tmr_cksel	[1]	b0	R/W	Timer clock source selection 0: System clock 1: Slow clock
cfg_sadc_smp_mode	[0]	b0	R/W	Sample rate mode 0: Sample rate depends on software start SADC, cfg_sadc_start 1: Sample rate depends on timer rate1: start

- Offset 0x010: SADC_SET1

Signal	Bits	Default	R/W	Description
reserved	[31:28]			
cfg_sadc_val_tst	[27:16]	d0	R/W	Set SADC adjust value, used for calibration Data format is s1.11.0
cfg_sadc_tst	[15:12]	d0	R/W	[0] SADC value bypass 0: sadc_i = sadc_i + cfg_sadc_val_tst, used for calibration 1: sadc_i = cfg_sadc_val_tst, used for bypass [1] SADC MSB bit inversion 0: disable 1: bit reverse [2] SADC control mode 1 0: normal mode, control by digital 1: manual mode, control by cfg_sadc_chx_sel [3] SADC control mode 2 0: normal mode, control by digital 1: manual mode, control by cfg_sadc_ena, cfg_sadc_vga_ena, cfg_sadc_ldo_ena respectively.
cfg_sadc_osr	[11:8]	d0	R/W	Oversample rate selection 0: No oversample 1: Oversample by 2 2: Oversample by 4 3: Oversample by 8 4: Oversample by 16 5: Oversample by 32 6: Oversample by 64 7: Oversample by 128 8: Oversample by 256 Others: reserved
cfg_sadc_chx_sel	[7:4]	d0	R/W	At test mode, select from CH0 to CH9
cfg_sadc_bit	[3:0]	d2	R/W	SAC output Resolution 0: 8-bit 1: 10-bit 2: 12-bit

3: 14-bit (oversample)

- Offset 0x100: SADC_WDMA_CTL0

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_sadc_wdma_ctl0	[0]	d0	W1C	[0] Start xdma for memory access 1: enable

- Offset 0x104: SADC_WDMA_CTL1

Signal	Bits	Default	R/W	Description
reserved	[31:1]			
cfg_sadc_wdma_ctl1	[0]	d0	W1C	Software reset xdma 1: Reset

- Offset 0x108: SADC_WDMA_SET0

Signal	Bits	Default	R/W	Description
cfg_sadc_wdma_set0	[31:0]	d0	R/W	[15:0] Segment size, 2-byte aligned 0: illegal value 1...32768: normal [31:16] Block size, 2-byte aligned 0: single mode 1..32768: normal

- Offset 0x10C: SADC_WDMA_SET1

Signal	Bits	Default	R/W	Description
cfg_sadc_wdma_set1	[31:0]	d0	R/W	Start address point, word (2-byte) aligned The 1-LSB bit will be ignored

- Offset 0x110: SADC_WDMA_SET2

Signal	Bits	Default	R/W	Description
reserved	[31:8]			
cfg_sadc_wdma_set2	[7:0]	d16	R/W	[0] Set initial address 0: when xdma start, initial address is loaded from ptr 1: when xdma reset, initial address is loaded from ptr [3:1] Reserved [5:4] DMA data format 0: word, 4-bytes aligned 1: halfword, 2-bytes aligned 2/3: 1-byte aligned [7:6] Reserved

- Offset 0x114: SADC_WDMA_R0

Signal	Bits	Default	R/W	Description
cfg_sadc_wdma_r0	[31:0]	d0	R	Status of xdma next pointer address

- Offset 0x118: SADC_WDMA_R1

Signal	Bits	Default	R/W	Description
cfg_sadc_wdma_r1	[31:0]	d0	R	[0] xdma interrupt status [1] xdma error interrupt status [2] OR(xdma interrupt status, xdma error interrupt status) [9:8] xdma counter status, used for ping-pong buffer to check buffer status [16] xdma enable [17] xdma_done [19:18] xdma_status

- Offset 0x120: SADC_INT_CLEAR

Signal	Bits	Default	R/W	Description
reserved	[31:28]			
cfg_sadc_int_clear	[27:0]	d0	W1C	[0] Clear SADC_WDMA interrupt 1: clear [1] Reserved [2] Clear SADC Done interrupt 1: clear [3] Clear SADC Valid interrupt 1: clear [4] Clear SADC Mode Done interrupt 1: clear [7:5] NoUse [17:8] Clear SADC monitor Low interrupt for CH9:0 1: clear [27:18] Clear SADC monitor High interrupt for CH9:0 1: clear

- Offset 0x124: SADC_INT_MASK

Signal	Bits	Default	R/W	Description
reserved	[31:28]			
cfg_sadc_int_mask	[27:0]	d0	R/W	[0] Mask SADC_WDMA interrupt [1] Reserved [2] Mask SADC Done interrupt [3] Mask SADC Valid interrupt [4] Mask SADC Mode Done interrupt [7:5] NoUse [17:8] Mask SADC monitor Low interrupt for CH9:0 [27:18] Mask SADC monitor High interrupt for CH9:0

- Offset 0x128: SADC_INT_STATUS

Signal	Bits	Default	R/W	Description
reserved	[31:0]			
cfg_sadc_int_status	[27:0]	d0	R	[0] Status of SADC_WDMA interrupt [1] Reserved [2] Status of SADC Done interrupt

	[3] Status of SADC Valid interrupt [4] Status os SADC Mode Done interrupt [7:5] NoUse [17:8] Status of SADC monitor Low interrupt for CH9:0 [27:18] Status of SADC monitor High interrupt for CH9:0

- Offset 0x12C: SADC_R0

Signal	Bits	Default	R/W	Description
reserved	[31:20]			
sadc_o_chx	[19:16]	b0	R	SADC channel index
	[15:14]			
sadc_o	[13:0]	d0	R	SADC output value after digital control and oversampling, data format is u0.14.0;

- Offset 0x130: SADC_R1

Signal	Bits	Default	R/W	Description
sadc_num_res	[31:16]	d0	R	Number of SADC result write into WDMA since last sadc_start trigger
reserved	[15:12]			
sadc_i_12b	[11:0]	d0	R	SADC result from analog, data format is u0.12.0

- Offset 0x134: SADC_R2

Signal	Bits	Default	R/W	Description
reserved	[31:18]			
sadc_ana_ena	[17]	d0	R	SADC analog enable
sadc_busy	[16]	d0	R	SADC mode control busy 0: idle 1: busy
reserved	[15:12]			
sadc_i_syn	[11:0]	d0	R	SADC result via AFIFO data format is u0.12.0

25.4.2. Registers for Each Channel

Each logical has the identical register sets with the different offset which is listed in below table.

Table 25-4 The Start Offset of Each Channel

Logical Channel (CHx)	Channel_Start_Offset
0	0x020
1	0x030
2	0x040
3	0x050
4	0x060
5	0x070

6	0x080
7	0x090
8	0x0A0
9	0x0B0

- Offset Channel_Start_Offset + 0x000: SADC_PNSEL_CHx

Signal	Bits	Default	R/W	Description
reserved	[31]			
cfg_sadc_edly_chx	[30:28]	d0	R/W	ADC end delay time. 0: 0.3μs 1: 1μs 2: 2μs 3: 3μs 4: 4μs 5: 8μs 6: 12μs 7: 16μs
reserved	[27]			
cfg_sadc_tacq_chx	[26:24]	d0	R/W	ADC result acquisition time. 0: 0.3μs 1: 1μs 2: 2μs 3: 3μs 4: 4μs 5: 8μs 6: 12μs 7: 16μs
cfg_sadc_pull_chx	[23:16]	d0	R/W	cfg_sadc_pull_chx[0] : Pull the P input to high cfg_sadc_pull_chx[1] : Pull the P input to low cfg_sadc_pull_chx[2] : Pull the N input to high cfg_sadc_pull_chx[3] : Pull the N input to low cfg_sadc_pull_chx[4] : Short the P input to VDD cfg_sadc_pull_chx[5] : Short the P input to GROUND cfg_sadc_pull_chx[6] : Short the N input to VDD cfg_sadc_pull_chx[7] : Short the N input to GROUND
cfg_sadc_ref_in_chx	[15:14]			Select P/N channel VGA reference 00b: both signal in 01b: P signal, N OP_vdd 10b: P OP_vdd, N signal 11b: both OP_vdd
cfg_sadc_gain_chx	[13:8]	d0	R/W	[11:8] VGA gain selection, +3dB/step [13:12] VGA gain selection, +6dB/step
cfg_sadc_nsel_chx	[7:4]	D15	R/W	Select N channel inputs of AUX ADC. Depends on the package, some AIOs may not be available. 0000b Select AIO0 0001b Select AIO1 0010b Select AIO2 0011b Select AIO3

				0100b Select AIO4 0101b Select AIO5 0110b Select AIO6 0111b Select AIO7 1000b Select Temperature Sensor N output (temp_n) 1001b Reserved. 1010b Select Battery voltage. Others Disabled.
cfg_sadc_psel_chx	[3:0]	d15	R/W	Select P channel inputs of AUX ADC. Depends on the package, some AIOs may not be available. 0000b Select AIO0 0001b Select AIO1 0010b Select AIO2 0011b Select AIO3 0100b Select AIO4 0101b Select AIO5 0110b Select AIO6 0111b Select AIO7 1000b Select Temperature Sensor P output (temp_p) 1001b Reserved. 1010b Select Battery voltage. Others Disabled.

- Offset Channel_Start_Offset + 0x004: SADC_SET_CHx

Signal	Bits	Default	R/W	Description
cfg_sadc_burst_chx	[31]	d1	R/W	SADC Burst mode selection 0: Disable burst mode 1: Select burst mode, SADC takes 2^Oversample number of samples and sends the average to Memory
reserved	[30:0]			

- Offset Channel_Start_Offset + 0x008: SADC_THD_CHx

Signal	Bits	Default	R/W	Description
reserved	[31:30]			
cfg_sadc_hthd_chx	[29:16]	d0	R/W	SADC high threshold, u0.14.0
reserved	[15:14]			
cfg_sadc_lthd_chx	[13:0]	d0	R/W	SADC low threshold, u0.14.0

26. AUX Comparator

The AUX Comparator is an analog comparator that compares two voltages, the input voltage V_{in} and the reference voltage V_{ref}. The comparator will output a logical high if V_{in} is greater than V_{ref}, otherwise it will output a logical low. Both input and reference voltages can be selected among AIOs and can be used to monitor the external analog signal.

26.1. Block Diagram

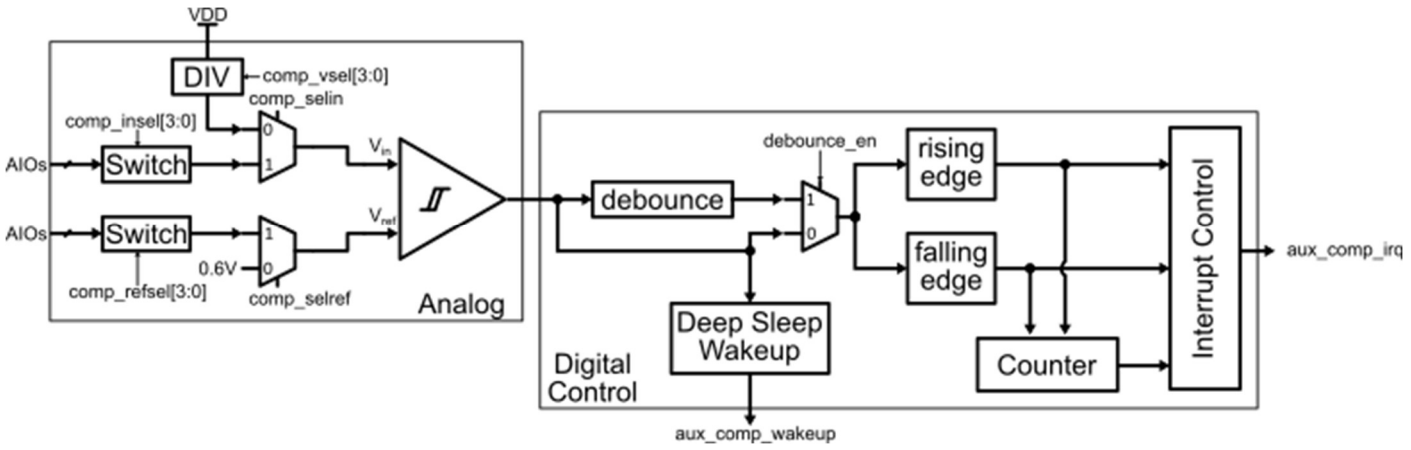


Figure 26-1 Block Diagram of AUX Comparator

26.2. Functional Descriptions

The AUX Comparator is divided into an analog part and a digital control as illustrated in Figure 26-1. The analog part includes the input switches and an analog comparator. The input voltage can be selected among AIOs or from the divided VDD. The reference voltage can also be selected among AIOs or from a bandgap reference of 0.6V. The analog comparator has a built-in hysteresis to prevent the output from damping.

The output of the analog comparator is then processed by the Digital Control. The output can be de-bounced and an interrupt will be generated on the rising, falling, or both edges.

If Counter Mode is enabled, the counter will count the rising, falling, or both edges of the output of the analog comparator. The interrupt won't be generated until the counter value exceeds the pre-defined value. Be aware to reset the counter value after the interrupt is triggered.

In battery-powered applications, the VDD of AUX Comparator is connected to the battery and the AUX Comparator can be used to monitor the battery voltage and provides the alarm interrupt when it drops below the threshold. In such case, the input and reference voltages shall be selected from the divided VDD and the bandgap reference respectively. The threshold voltage of VDD can be set from 1.8V to 2.55V with 50mV per step by the comp_vsel register. The comparator will output a logical high when the voltage of VDD is above the threshold and a logical low when the voltage of VDD is below the threshold.

26.3. Registers

- Offset 0x00: AUXCOMP_ANA_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:20]			
comp_en_start	[25:24]	0x1	R/W	AUX bandgap start-up control. Reserved for analog tuning.
reserved	[23:21]			
comp_tc	[20]	0x0	R/W	Empty register. Reserved for future use.
comp_insel	[19:16]	0x0	R/W	Vin selection from AIOs. 0x0 AIO0 0x1 AIO1 0x2 AIO2 0x3 AIO3 0x4 AIO4 0x5 AIO5 0x6 AIO6 0x7 AIO7 Other reserved
comp_refsel	[15:12]	0x0	R/W	Vref selection from AIOs. 0x0 AIO0 0x1 AIO1 0x2 AIO2 0x3 AIO3 0x4 AIO4 0x5 AIO5 0x6 AIO6 0x7 AIO7 Other reserved
comp_vsel	[11:8]	0x0	R/W	Select the VDD trigger threshold from 1.8V (0x0) to 2.55V (0xF), 50mV per step.
comp_psrr	[7]	0x0	R/W	AUX buffer source control. Reserved for analog tuning.
comp_swdiv	[6]	0x1	R/W	Empty register. Reserved for future use.
comp_selhys	[5:4]	0x0	R/W	AUX hysteresis control. Reserved for analog tuning.
comp_pw	[3:2]	0x3	R/W	AUX current step control. Reserved for analog tuning.
comp_selin	[1]	0x0	R/W	Select the Vin source. 1b From one of AIOs selected by comp_insel[3:0].

				0b From voltage divider output.
comp_selref	[0]	0x0	R/W	Select the Vref source. 1b From one of AIOs selected by comp_refsel[3:0]. 0b From the bandgap reference, 0.6V.

- Offset 0x04: AUXCOMP_DIG_CTRL0

Signal	Bits	Default	R/W	Description
counter_trigger_th	[31:16]	0xF	R/W	Set the trigger threshold of the counter mode. The interrupt will be triggered when counter_cnt > counter_trigger_th.
reserved	[15:14]			
ds_wakeup_pol	[13]	0	R/W	Set the wakeup polarity in Deep Sleep when ds_wakeup_en is enabled. 0b level low 1b level high
ds_wakeup_en	[12]	0	R/W	Enable wakeup in Deep Sleep. 0b disable 1b enable
counter_mode_edge	[11:10]	0	R/W	Select the voltage divider output from 1.8V (0x0) to 2.55V (0xF), 50mV per step.
reserved	[9]			
counter_mode_en	[8]	0	R/W	Enable counter mode. When counter mode is enabled, every comparator event will let the counter count up until it reaches the trigger count. If counter mode will be used in Deep Sleep, slow_clk must be also enabled. 0b disable 1b enable
debounce_sel	[7:6]	0	R/W	Debounce time selection. 00b 2 slow_clk 01b 4 slow_clk 1xb 8 slow_clk
reserved	[5]			
debounce_en	[4]	0	R/W	Enable debounce. 0b disable 1b enable
reserved	[3]			
comp_en_ds	[2]	0	R/W	Enable AUX Comparator in Deep Sleep Mode. 0b disable 1b enable
comp_en_sp	[1]	0	R/W	Enable AUX Comparator in Sleep Mode. 0b disable 1b enable
comp_en_nm	[0]	0	R/W	Enable AUX Comparator in Normal Mode. 0b disable 1b enable

- Offset 0x08: AUXCOMP_DIG_CTRL1

Signal	Bits	Default	R/W	Description
reserved	[31:21]			
comp_settle_time	[20:16]	0x8	R/W	AUX comparator settling time. Reserved for analog tuning.
reserved	[15:12]			
clr_counter	[11]	0	W1C	Write 1 to clear the counter value.
clr_intr_counter	[10]	0	W1C	Write 1 to clear the counter-triggered interrupt status
clr_intr_falling	[9]	0	W1C	Write 1 to clear the falling-edge interrupt status
clr_intr_rising	[8]	0	W1C	Write 1 to clear the rising-edge interrupt status
reserved	[7:3]			
en_intr_counter	[2]	0	R/W	Enable the counter mode interrupt of AUX Comparator. 0b disable 1b enable
en_intr_falling	[1]	0	R/W	Enable the falling-edge interrupt of AUX Comparator. 0b disable 1b enable
en_intr_rising	[0]	0	R/W	Enable the rising-edge interrupt of AUX Comparator. 0b disable 1b enable

- Offset 0x0C: AUXCOMP_DIG_CTRL2

Signal	Bits	Default	R/W	Description
counter_cnt	[31:16]	0	R	The current counter value if Counter Mode is enabled.
reserved	[15:9]			
comp_out	[8]	0	R	The output of AUX Comparator
reserved	[7:3]			
sta_intr_counter	[2]	0	R	Counter mode interrupt status.
sta_intr_falling	[1]	0	R	Falling-edge interrupt status.
sta_intr_rising	[0]	0	R	Rising-edge interrupt status

27. BOD Comparator

The BOD Comparator is dedicatedly used as the brown-out detector. It is similar to the AUX Comparator but its two inputs are fixed to the divided VDD and the bandgap reference.

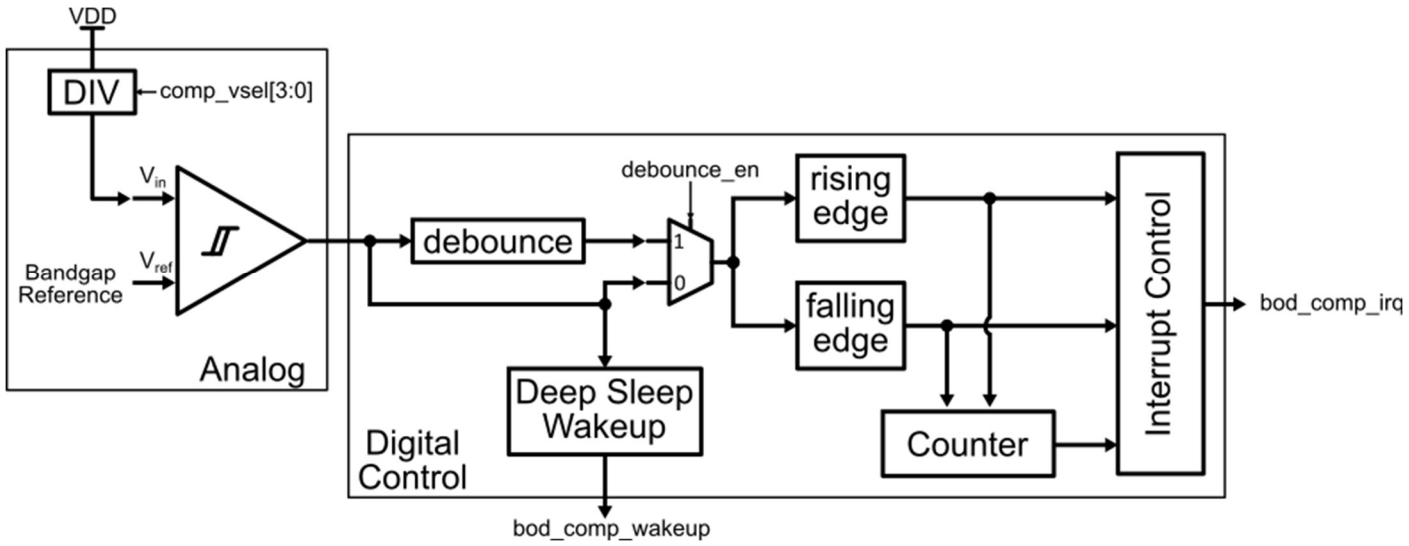


Figure 27-1 Block Diagram of BOD Comparator

The threshold voltage of VDD can be set from 1.8V to 2.55V with 50mV per step by the comp_vsel register. The comparator will output a logical high when the voltage of VDD is above the threshold and a logical low when the voltage of VDD is below the threshold.

27.1. Registers

- Offset 0x00: BODCOMP_ANA_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:12]			
comp_vsel	[11:8]	0x0	R/W	Select the VDD trigger threshold from 1.8V (0x0) to 2.55V (0xF), 50mV per step.
reserved	[7:2]			

- Offset 0x04: BODCOMP_DIG_CTRL0

Signal	Bits	Default	R/W	Description
counter_trigger_th	[31:16]	0xF	R/W	Set the trigger threshold of the counter mode. The interrupt will be triggered when counter_cnt > counter_trigger_th.

reserved	[15:14]			
ds_wakeup_pol	[13]	0	R/W	Set the wakeup polarity in Deep Sleep when ds_wakeup_en is enabled. 0b level low 1b level high
ds_wakeup_en	[12]	0	R/W	Enable wakeup in Deep Sleep. 0b disable 1b enable
counter_mode_edge	[11:10]	0	R/W	Select the voltage divider output from 1.8V (0x0) to 2.55V (0xF), 50mV per step.
reserved	[9]			
counter_mode_en	[8]	0	R/W	Enable counter mode. When counter mode is enabled, every comparator event will let the counter count up until it reaches the trigger count. If counter mode will be used in Deep Sleep, slow_clk must be also enabled. 0b disable 1b enable
debounce_sel	[7:6]	0	R/W	Debounce time selection. 00b 2 slow_clk 01b 4 slow_clk 1xb 8 slow_clk
reserved	[5]			
debounce_en	[4]	0	R/W	Enable debounce. 0b disable 1b enable
reserved	[3]			
comp_en_ds	[2]	0	R/W	Enable BOD Comparator in Deep Sleep Mode. 0b disable 1b enable
comp_en_sp	[1]	0	R/W	Enable BOD Comparator in Sleep Mode. 0b disable 1b enable
comp_en_nm	[0]	0	R/W	Enable BOD Comparator in Normal Mode. 0b disable 1b enable

- Offset 0x08: BODCOMP_DIG_CTRL1

Signal	Bits	Default	R/W	Description
reserved	[31:12]			
clr_counter	[11]	0	W1C	Write 1 to clear the counter value.
clr_intr_counter	[10]	0	W1C	Write 1 to clear the counter-triggered interrupt status
clr_intr_falling	[9]	0	W1C	Write 1 to clear the falling-edge interrupt status
clr_intr_rising	[8]	0	W1C	Write 1 to clear the rising-edge interrupt status
reserved	[7:3]			
en_intr_counter	[2]	0	R/W	Enable the counter mode interrupt of BOD Comparator. 0b disable 1b enable

en_intr_falling	[1]	0	R/W	Enable the falling-edge interrupt of BOD Comparator. 0b disable 1b enable
en_intr_rising	[0]	0	R/W	Enable the rising-edge interrupt of BOD Comparator. 0b disable 1b enable

- Offset 0x0C: BODCOMP_DIG_CTRL2

Signal	Bits	Default	R/W	Description
counter_cnt	[31:16]	0	R	The current counter value if Counter Mode is enabled.
reserved	[15:9]			
comp_out	[8]	0	R	The output of BOD Comparator
reserved	[7:3]			
sta_intr_counter	[2]	0	R	Counter mode interrupt status.
sta_intr_falling	[1]	0	R	Falling-edge interrupt status.
sta_intr_rising	[0]	0	R	Rising-edge interrupt status

28. Crypto Engine

To execute security applications faster, a crypto accelerator is supplied for various algorithms. This module is essentially a processor which accepts binary codes to execute.

28.1. Block Diagram

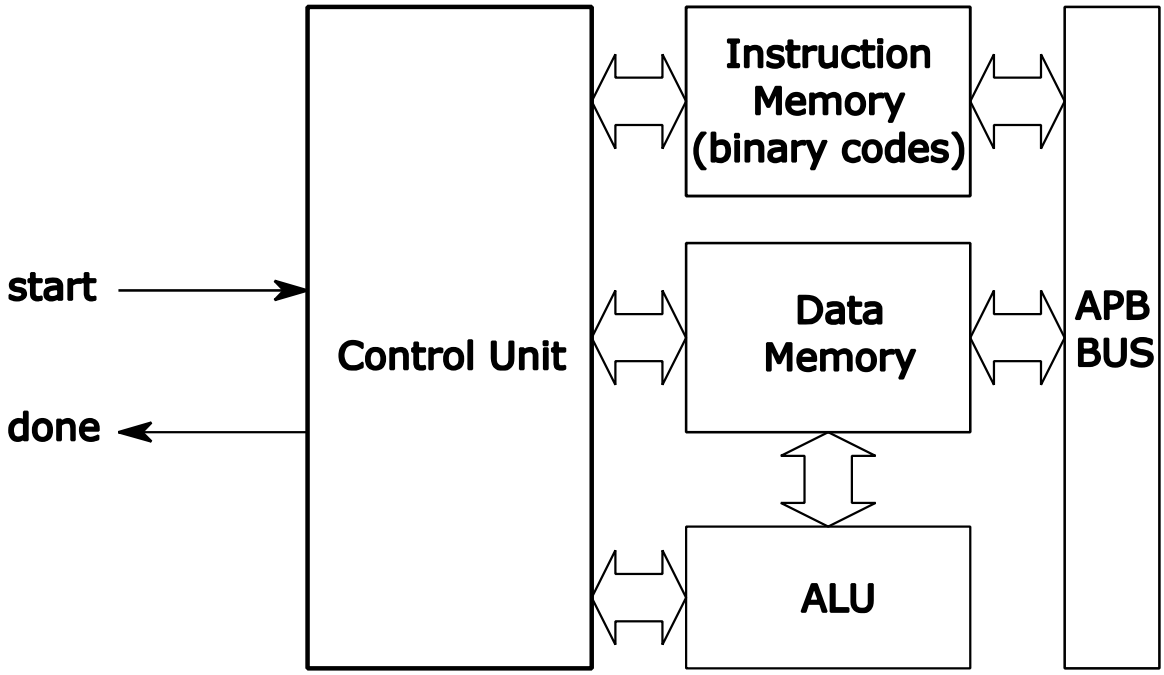


Figure 28-1 Block Diagram of Crypto Engine

28.2. Features

The crypto engine supports following cryptographic schemes:

- AES
  - 128-bit, 192-bit and 256-bit key length encryption/decryption
  - ECB, CBC, CTR, CMAC, CCM cipher modes
- SHA256
  - Can be extended to HMAC, HMAC DRBG or other
- ECC
  - 192-bit and 256-bit key length

- Both prime field GF(p) and binary field GF(2m)
- NIST P-192 curve (also knowns as SECP192R1)
- NIST P-256 curve (also knowns as SECP256R1 or Prime256v1)
- NIST B-163 curve (also knowns as SECT163R2)
- Curve 25519
- ECDH and ECDSA protocol

28.3. Functional Description

There are internal memories in the crypto accelerator: instruction memory and data memory. The instruction memory needs binary code pre-loaded before starting. Due to the requirements of various binary codes according to different crypto algorithms, refer to the SDK document for the usage of this module.

29. Cache Controller

The cache controller is designed to manage the cache memory, which serves as an intermediary between the CPU and the Flash memory. Its primary role is to enhance system performance by reducing the time the CPU spends waiting for data from the slower Flash memory. The cache controller accomplishes this by storing frequently accessed data and instructions from the Flash memory in the cache, allowing the CPU to retrieve them quickly.

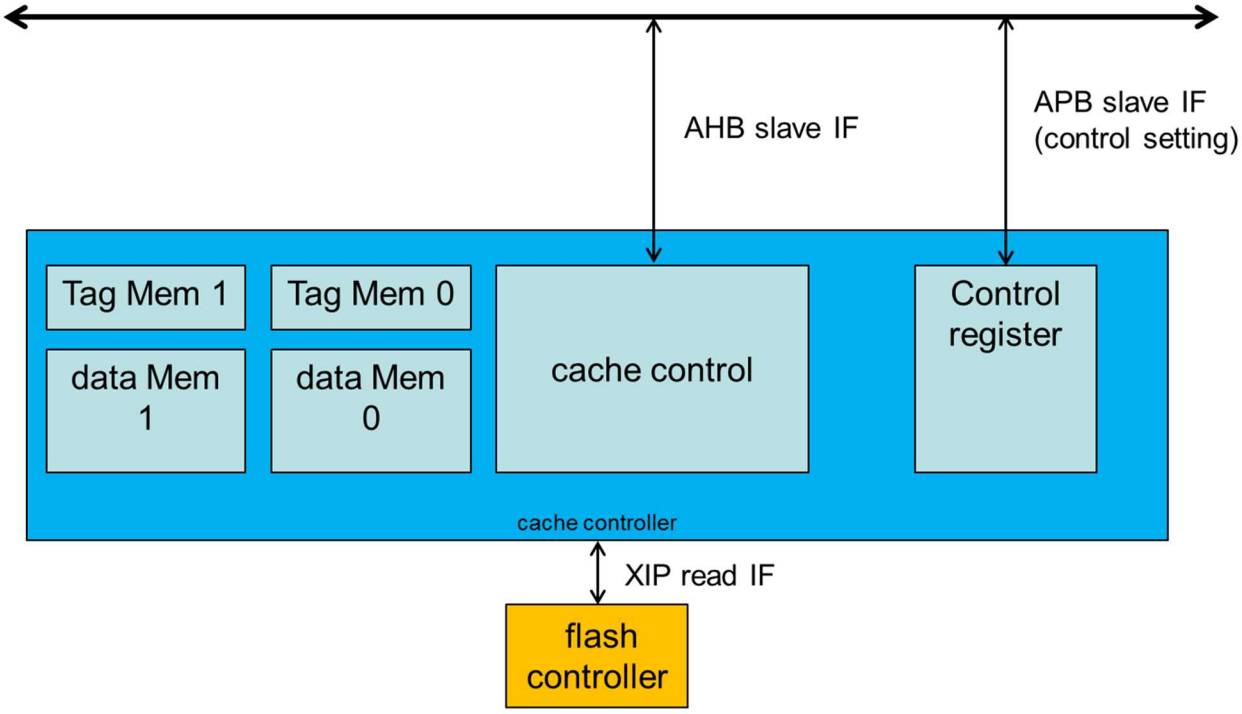


Figure 29-1 Block Diagram of Cache Controller

The block diagram of the cache controller is illustrated in Figure 29-1. The 2-way set associative scheme is implemented. It is possible to trade off the system performance against the power consumption by disabling the way-1 cache via the register setting.

Be aware that the cache controller handles only the READ path from the Flash memory. Writing data to the Flash memory shall follow the flash write procedures of Flash Controller.

29.1. Registers

Registers of Cache Controller are resident in the System Control space whose base address is 0x50000000 for the secure partition or 0x40000000 for the non-secure partition.

- Offset 0x48: CACHE_CTRL

Signal	Bits	Default	R/W	Description
reserved	[31:10]			
cache_way_1_clr	[9]	0	W1C	Write 1 to this bit to clear the way-1 cache.
cache_way_0_clr	[8]	0	W1C	Write 1 to this bit to clear the way-0 cache.
reserved	[7:2]			
cache_way_1_en	[1]		R/W	Cache way-1 enable. 0b Disable way-1 cache. 1b Enable way-1 cache.
cache_en	[0]		R/W	Cache enable. 0b Disable cache. Both way-0 and way-1 cache are disabled. 1b Enable cache. The way-1 cache can be further disabled by setting cache_way_1_en = 0. The way-9 cache is always enabled once cache_en = 1.

30. Reference Manual Revision History

Revision	Date	Description
1	2025-03-31	• Beta Release
2	2025-10-28	GA Release • Added section 5.7 registers • Added values to 10.3.11 GPIO_DRV_CTRL0 (offset: 0x30)
3	2026-03-19	• Fixed wrong selections of comp_selin and comp_selref in AUX comparator • Add analog control registers for AUX Comparator • Add GPIO and Peripheral MUX

31. Contact Us

Contact Information	
Contact	www.TridentIoT.com
Sales	sales@tridentiot.com
Technical Support	support@tridentiot.com

31.1. Support

Informational Links	
Support Documents	https://tridentiot.com/products/
SDK Manuals	https://tridentiot.github.io/tridentiot-sdk-docs/
Hardware Documents	https://tridentiot.com/technology/hardware/