PANCHIP

PAN101x series

User Manual

V1.2 May. 2025

Panchip Microelectronics Co., Ltd.

BLE SoC Transceiver

General Description

PAN101x series integrates BLE5.3 and 2.4GHz dual-mode wireless SoC transceiver. The transceiver works in the 2.400-2.483GHz universal ISM frequency band. There is a 256KB Flash program memory and a 16KB SRAM memory. In addition, PAN101x series is equipped with a wealth of peripherals, including up to 12 GPIOs, 8-channel PWM, one 32-bit timer, two 24-bit timers, one I2C, two UARTs, two SPIs, four external channels ADC, WDT, WWDT, USB2.0(Full Speed), 32K RC etc. PAN101x series is suitable for application of wireless mouse, smart home and electronic shelf label.

Key Features

- **MCU**
  - 32-bit MCU core running up to 48MHz
- **Memory**
  - 256KB flash supporting deep power-down mode
  - 16KB SRAM
  - 128B eFuse
  - 4KB I-cache
- **Low Power**
  - Active mode RX: 2.5mA@1Mbps(DCDC)
  - Active mode TX at 0dBm: 5.06mA(DCDC)
  - Standby mode : 0.28uA
  - Standby mode(16K SRAM retention):
  - 1.88uA (GPIO, XTL, RCL can wake up)
  - Deep sleep mode: 3.37uA (All Logic Retention, GPIO, XTL, RCL can wake up)
- **Clock**
  - 32MHz RC
  - 32MHz XTAL
  - 32kHz RC
  - 32.768kHz XTAL
  - DPLL(48MHz)
- **RF**
  - Mode
    - BLE5.3 modes: 1Mbps, 2Mbps, 500kbps, 125kbps
    - 2.4G private protocol: 1Mbps, 2Mbps 500kbps, 250kbps, 125kbps, supporting hardware ACK
  - Output power: up to 8.5dBm
  - Receiver
    - -98dBm@125kbps
    - -98dBm@500kbps
    - -95dBm@1Mbps
    - -92dBm@2Mbps
  - RSSI
    - Resolution: 0.25dB
    - Accuracy: ±2dB
    - Range: -90 to -15dBm
  - Single antenna supported
  - Safety regulations: BQB / ETSI / FCC
- **Peripheral**
  - Up to 12 GPIOs
  - 8-channel PWM one 32-bit timer, two 24-bit timers
  - One I2C
  - Two UARTs
  - Two SPIs
  - 2-channel DMA
  - 9-channel ADC (4 ext, VBG_1P2, VBG_VT, 1/4VDD, VBG_0P6, Temp2)
  - WDT / WWDT
  - IO / BOD / POR / LVR / System reset
  - FMC(Support IAP, support the boot loader with address 0x0)
  - Clock measurement and clock calibration
  - USB2.0(Full_speed)
  - Flash data encryption
- **Temperature sensor**
  - Support temperature sensor
  - Test range: -40 to 85℃
- **Power Management**
  - Integrated voltage regulator
  - Operating voltage: 1.8 to 3.6V (DCDC ON: 2 to 3.6V)
- **Package**
  - SSOP24 / MSOP10
- **Operating Condition**
  - Operating temperature: -40 to 85℃
  - Storage temperature: -60 to 150℃

Typical Applications

- Electronic Shelf Label
- Wireless mouse
- LED light control

Bluetooth Features

Bluetooth Low Energy Controller

The PAN101x series Bluetooth Low Energy Controller supports all low-energy features required by Bluetooth specification version 5.3. The controller supports the following:

- **Support 1M PHY, 2M PHY and Coded PHY (s2 and s8)**
- **Support Advertising, and Connection and Peripheral role**
- **Up to 2 Link Layer state machines concurrently:**
  - Advertising
  - Connection
- **Support LE Features:**
  - LL Encryption
  - LE Data Packet Length Extension
  - Channel Selection Algorithm #2
  - Constant Tone Extension
- **Support Update Channel Statistics**

Bluetooth Host

- **Generic Access Profile (GAP) with all possible LE roles**
  - Peripheral
  - Broadcaster
- **GATT (Generic Attribute Profile)**
  - Server (to be a sensor)
  - Client (to connect to sensors)
- **Pairing support, including the Secure Connections feature from Bluetooth 4.2**
- **Non-volatile storage support for permanent storage of Bluetooth-specific settings and data**
- **Clean HCI driver abstraction**
  - 3-Wire (H5) & 5-Wire (H4) UART
  - SPI
  - Local controller support as a virtual HCI driver

Proprietary Radio 2.4G Features

- Support 250K, 1M and 2M PHY
- XN297L, PAN1026 protocol compliant
- Support No Acknowledge, Acknowledge and Acknowledge with payload
- Support CRC8, CRC16 and CRC24
- Support whitening
- Compatible with Bluetooth frame structure, can simulate Bluetooth broadcast and scanning
- Compatible with Bluetooth Coded PHY S2/S8
- Supports 2-byte address
- Support the same spread spectrum function as the BLE protocol

Content

List of Tables

List of Figures

1 Block Diagram

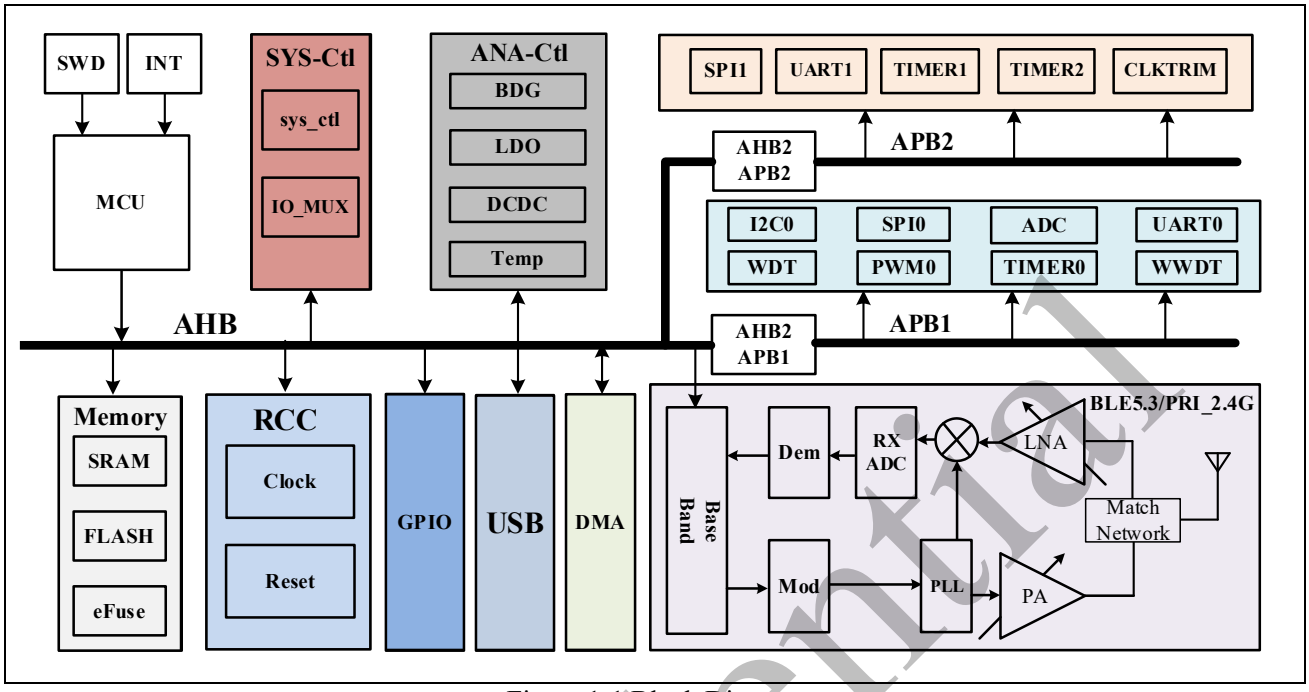


Figure 1-1 Block Diagram

2 Pin Information

2.1 Pin Diagram

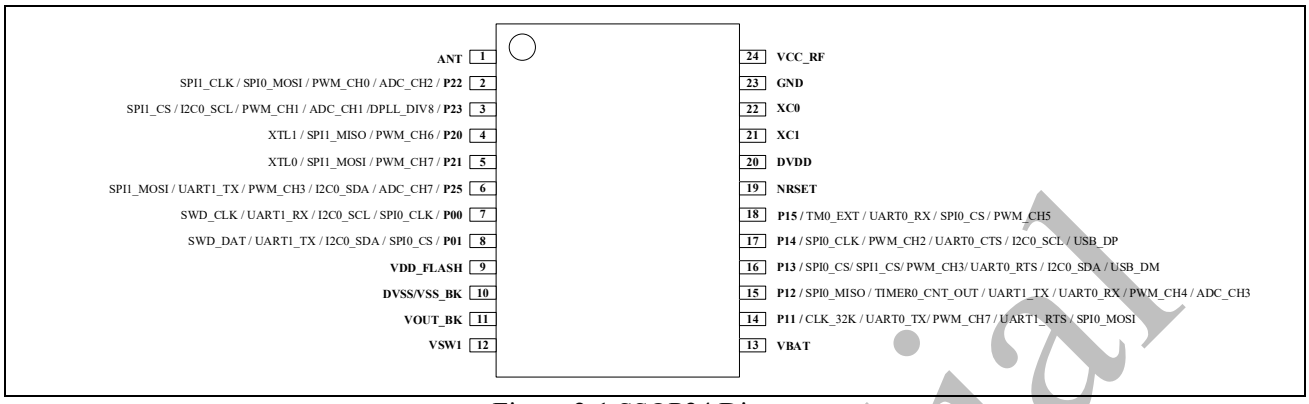


Figure 2-1 SSOP24 Diagram

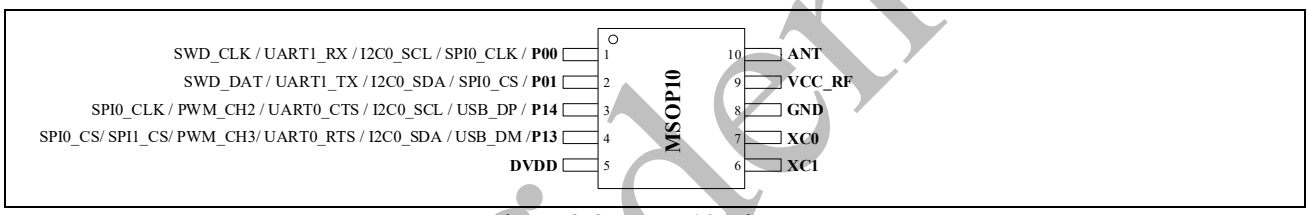


Figure 2-2 MSOP10 Diagram

2.2 Pin Descriptions

Detail pin descriptions see Table 2-1.

Table 2-1 Pin Descriptions

Pin No.		Pin Name	Pin Type	Description
SSOP24	MSOP10			
1	10	ANT	AI/AO	RF antenna , an external antenna is required for use
2	-	P22	I/O	General-purpose digital input and output
		SPI1_CLK	I/O	SPI1 clock
		PWM_CH0	O	Channel 0 PWM output
		SPI0_MOSI	I/O	SPI0 MOSI
		ADC_CH2	AI	Channel 2 ADC input
3	-	P23	I/O	General-purpose digital input and output
		SPI1_CS	I/O	SPI1 CS
		PWM_CH1	O	Channel 1 PWM output
		DPLL_DIV8	O	DPLL_DIV8 output
		I2C0_SCL	I/O	I2C0 SCL
		ADC_CH1	AI	Channel 1 ADC input
4	-	P20	I/O	General-purpose digital input and output
		XTL1	AO	External 32.768kHz clock source output
		SPI1_MISO	I/O	SPI1 MISO
		PWM_CH6	O	Channel 6 PWM output
5	-	P21	I/O	General-purpose digital input and output
		XTL0	AI	External 32.768kHz clock source input
		SPI1_MOSI	I/O	SPI1 MOSI
		PWM_CH7	O	Channel 7 PWM output
6	-	P25	I/O	General-purpose digital input and output
		UART1_TX	O	UART1 TX
		SPI1_MOSI	I/O	SPI1 MOSI
		PWM_CH3	O	Channel 3 PWM output
		I2C0_SDA	I/O	I2C0 SDA
		ADC_CH7	AI	Channel 7 ADC input
7	1	P00	I/O	General-purpose digital input and output
		SWD_CLK	I	SWD clock input

		UART1_RX	I	UART1 RX
		I2C0_SCL	I/O	I2C0 SCL
		SPI0_CLK	I/O	SPI0 clock
8	2	P01	I/O	General-purpose digital input and output
		SWD_DAT	I/O	SWD data input and output
		UART1_TX	O	UART1 TX
		I2C0_SDA	I/O	I2C0 SDA
		SPI0_CS	I/O	SPI0 CS
9	-	VDD_FLASH	P	FLASH power input
10	-	DVSS/VSS_BK	P	Common ground terminal of DCDC power supply, independent power ground
11	-	VOUT_BK	P	DCDC voltage output
12	-	VSW1	P	DCDC internal power switch (switching frequency is about 650kHz), an external inductor is required when using
13	-	VBAT	P	Power input (VDD)
14	-	P11	I/O	General-purpose digital input and output
		UART1_RTS	O	UART1 RTS
		SPI0_MOSI	I/O	SPI0 MOSI
		PWM_CH7	O	Channel 7 PWM output
		CLK_32K	O	CLK_32K output
		UART0_TX	O	UART0 TX
15	-	P12	I/O	General-purpose digital input and output
		UART0_RX	I	UART0 RX
		TIMER0_CNT_OUT	O	Timer0 output
		UART1_TX	O	UART1 TX
		SPI0_MISO	I/O	SPI0 MISO
		PWM_CH4	O	Channel 4 PWM output
		ADC_CH3	AI	Channel 3 ADC input
16	4	P13	I/O	General-purpose digital input and output
		UART0_RTS	O	UART0 RTS
		I2C0_SDA	I/O	I2C0 SDA
		PWM_CH3	O	Channel 3 PWM output
		SPI1_CS	I/O	SPI1 CS
		SPI0_CS	I/O	SPI0 CS

		USB_DM	AI/AO	USB DM
17	3	P14	I/O	General-purpose digital input and output
		UART0_CTS	I	UART0 CTS
		I2C0_SCL	I/O	I2C0 SCL
		PWM_CH2	O	Channel 2 PWM output
		SPI0_CLK	I/O	SPI0 clock
		USB_DP	AI/AO	USB DP
18	-	P15	I/O	General-purpose digital input and output
		SPI0_CS	I/O	SPI0 CS
		PWM_CH5	O	Channel 5 PWM output
		TM0_EXT	I	Timer0 external input
		UART0_RX	I	UART0 RX
19	-	NRSET	I	Reset pin
20	5	DVDD	P	HLDO output , typical value 1.2V
21	6	XC1	AO	External 32MHz clock source output
22	7	XC0	AI	External 32MHz clock source input
23	8	GND	P	Ground
24	9	VCC_RF	P	RF power input, can be directly connected to VOUT_BK

3 Function Description

3.1 System Manager

3.1.1 Overview

System management includes the following sections:

- System Memory Map and description
- System Timer (SysTick)
- Nested Vectored Interrupt Controller (NVIC)
- System Control registers (SCB)

3.1.2 Memory Organization

3.1.2.1 Overview

The PAN101x series provides 4G-byte addressing space. The addressing space assigned to each on-chip controllers is shown in the Table 3-1. The detailed register definition, addressing space, and programming details will be described in the following sections for each on-chip peripheral. The PAN101x series only supports little-endian data format.

3.1.2.2 System Memory Map

The memory locations assigned to each on-chip controllers are shown in the Table 3-1.

Address	Peripheral	BUS	
0xE0000000-0xE00FFFFF	M0		
0x50020000-0x5003FFFF	BLE5.3	AHB	
0x400A0000-0x400AFFFF	USB		
0x40090000-0x4009FFFF	Reserved		
0x40080000-0x4008FFFF	eFuse	AHB	
0x40070000-0x4007FFFF	ANACTL		
0x40060000-0x4006FFFF	DMA		
0x40050000-0x4005FFFF	FLASH(control, accelerator)		
0x40040000-0x4004FFFF	CLKS, RESET, BOD, LVR		
0x40030000-0x4003FFFF	SYSTEM（IOMUX）		
0x40020000-0x4002FFFF	GPIO		
0x4001A000-0x4001AFFF	Reserved	AHB	APB2
0x40019000-0x40019FFF	Reserved		
0x40018000-0x40018FFF	Reserved		
0x40017000-0x40017FFF	Reserved		
0x40016000-0x40016FFF	CLKTRIM		
0x40015000-0x40015FFF	TIMER2		
0x40014000-0x40014FFF	TIMER1		

0x40013000-0x40013FFF	UART1		
0x40012000-0x40012FFF	Reserved		
0x40011000-0x40011FFF	SPI1		
0x40010000-0x40010FFF	Reserved		
0x4000C000-0x4000CFFF	Reserved		
0x4000B000-0x4000BFFF	Reserved		APB1
0x4000A000-0x4000AFFF	Reserved		
0x40009000-0x40009FFF	Reserved		
0x40008000-0x40008FFF	TIMER0		
0x40007000-0x40007FFF	WWDT		
0x40006000-0x40006FFF	WDT		
0x40005000-0x40005FFF	ADC		
0x40004000-0x40004FFF	PWM0		
0x40003000-0x40003FFF	UART0		
0x40002000-0x40002FFF	Reserved		
0x40001000-0x40001FFF	SPI0		
0x40000000-0x40000FFF	I2C0		
0x20000000-0x2000FFFF	SRAM		
0x00800000-0x00FFFFFF	ROM		
0x00000000-0x007FFFFF	FLASH		

Table 3-1 Address Space Assignments for On-Chip Modules

3.1.3 System Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SYS Base Address: SYS_BA = 0x4003_0000				
SYS_P0_MFP	SYS_BA+0x00	R/W	P0 Multiple Function and Input Type Control Register	0x0000_0003
SYS_P1_MFP	SYS_BA+0x04	R/W	P1 Multiple Function and Input Type Control Register	0x0000_0000
SYS_P2_MFP	SYS_BA+0x08	R/W	P2 Multiple Function and Input Type Control Register	0x0000_0000
SYS_REGLCTL	SYS_BA+0x40	R/W	Register Write-Protection Control Register	0x0000_0000
SYS_STATUS	SYS_BA+0x44	RO	ROM Status Register	0x0000_0000
SYS_CTRL0	SYS_BA+0x48	R/W	System Control0 Register	0x0000_0000
SYS_CTRL1	SYS_BA+0x4C	R/W	System Control1 Register	0x0004_0004

3.1.4 System Register Description

3.1.4.1 Multiple Function Port0 Control Register (SYS_P0_MFP)

Register	Offset	R/W	Description	Reset Value
SYS_P0_MFP	SYS_BA+0x00	R/W	P0 Multiple Function and Input Type Control Register	0x0000_0003

Bits	Description	
[31:24]	Reserved	Reserved.
[23:16]	EXT[7:0]	P0[7:0] Alternate Function Selection Extension The pin function of P0 depends on EXT, MFP and ALT.
[15:8]	ALT[7:0]	P0[7:0] Alternate Function Select Bit The pin function of P0 depends on EXT, MFP and ALT.
[7:2]	Reserved	Reserved.
[1]	MFP [1]	P0.1 Alternate Function Select Bit Bits EXT[1] (SYS_P0_MFP[17]), ALT[1] (SYS_P0_MFP[9]), and MFP[1] (SYS_P0_MFP[1]) determine the P0.1 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = swd_dat function is selected. (0, 1, 0) = uart1_tx function is selected. (0, 1, 1) = i2c0_sda function is selected. (1, 0, 0) = spi0_cs function is selected. (1, 1, 1) = Reserved.
[0]	MFP [0]	P0.0 Alternate Function Select Bit Bits EXT[0] (SYS_P0_MFP[16]), ALT[0] (SYS_P0_MFP[8]), and MFP[0] (SYS_P0_MFP[0]) determine the P0.0 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = swd_clk function is selected. (0, 1, 0) = uart1_rx function is selected. (0, 1, 1) = i2c0_scl function is selected. (1, 0, 0) = spi0_clk function is selected. (1, 1, 1) = Reserved.

3.1.4.2 Multiple Function Port1 Control Register (SYS_P1_MFP)

Register	Offset	R/W	Description	Reset Value
SYS_P1_MFP	SYS_BA+0x04	R/W	P1 Multiple Function and Input Type Control Register	0x0000_0000

Bits	Description	
[31:24]	Reserved	Reserved.
[23:16]	EXT[7:0]	P1[7:0] Alternate Function Selection Extension The pin function of P1 depends on EXT, MFP and ALT.
[15:8]	ALT[7:0]	P1[7:0] Alternate Function Select Bit The pin function of P1 depends on EXT, MFP and ALT.
[7:6]	Reserved	Reserved.
[5]	MFP [5]	P1.5 Alternate Function Select Bit Bits EXT[5] (SYS_P1_MFP[21]), ALT[5] (SYS_P1_MFP[13]), and MFP[5] (SYS_P1_MFP[5]) determine the P1.5 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_cs function is selected. (0, 1, 0) = pwm_ch5 function is selected. (0, 1, 1) = tm0_ext function is selected. (1, 0, 0) = uart0_rx function is selected. (1, 1, 1) = Reserved.
[4]	MFP [4]	P1.4 Alternate Function Select Bit Bits EXT[4] (SYS_P1_MFP[20]), ALT[4] (SYS_P1_MFP[12]), and MFP[4] (SYS_P1_MFP[4]) determine the P1.4 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_cts function is selected. (0, 1, 0) = i2c0_scl function is selected. (0, 1, 1) = pwm_ch2 function is selected. (1, 1, 0) = spi0_clk function is selected. (1, 1, 1) = Reserved.
[3]	MFP [3]	P1.3 Alternate Function Select Bit Bits EXT[3] (SYS_P1_MFP[19]), ALT[3] (SYS_P1_MFP[11]), and MFP[3] (SYS_P1_MFP[3]) determine the P1.3 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_rts function is selected. (0, 1, 0) = i2c0_sda function is selected. (0, 1, 1) = pwm_ch3 function is selected. (1, 1, 0) = spi1_cs function is selected. (1, 1, 1) = spi0_cs function is selected.
[2]	MFP [2]	P1.2 Alternate Function Select Bit Bits EXT[2] (SYS_P1_MFP[18]), ALT[5] (SYS_P1_MFP[10]), and MFP[2] (SYS_P1_MFP[2]) determine the P1.2 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_rx function is selected. (0, 1, 0) = pwm_ch4 function is selected. (0, 1, 1) = timer0_cnt_out function is selected.

		(1, 0, 0) = uart1_tx function is selected. (1, 1, 1) = spi0_miso function is selected.
[1]	MFP [1]	P1.1 Alternate Function Select Bit Bits EXT[1] (SYS_P1_MFP[17]), ALT[1] (SYS_P1_MFP[9]), and MFP[1] (SYS_P1_MFP[1]) determine the P1.1 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart1_rts function is selected. (0, 1, 0) = spi0_mosi function is selected. (0, 1, 1) = pwm_ch7 function is selected. (1, 0, 0) = clk_32k function is selected. (1, 1, 1) = uart0_tx function is selected.
[0]	Reserved	Reserved.

3.1.4.3 Multiple Function Port2 Control Register (SYS_P2_MFP)

Register	Offset	R/W	Description	Reset Value
SYS_P2_MFP	SYS_BA+0x08	R/W	P2 Multiple Function and Input Type Control Register	0x0000_0000

Bits	Description	
[31:24]	Reserved	Reserved.
[23:16]	EXT[7:0]	P2[7:0] Alternate Function Selection Extension The pin function of P2 depends on EXT, MFP and ALT.
[15:8]	ALT[7:0]	P2[7:0] Alternate Function Select Bit The pin function of P2 depends on EXT, MFP and ALT.
[7:6]	Reserved	Reserved.
[5]	MFP [5]	P2.5 Alternate Function Select Bit Bits EXT[5] (SYS_P2_MFP[21]), ALT[5] (SYS_P2_MFP[13]), and MFP[5] (SYS_P2_MFP[5]) determine the P2.5 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_mosi function is selected. (0, 1, 0) = uart1_tx function is selected. (0, 1, 1) = pwm_ch3 function is selected. (1, 0, 1) = Reserved. (1, 1, 0) = i2c0_sda function is selected. (1, 1, 1) = Reserved.
[4]	Reserved	Reserved.
[3]	MFP [3]	P2.3 Alternate Function Select Bit Bits EXT[3] (SYS_P2_MFP[19]), ALT[3] (SYS_P2_MFP[11]), and MFP[3] (SYS_P2_MFP[3]) determine the P2.3 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_cs function is selected. (0, 1, 0) = pwm_ch1 function is selected. (0, 1, 1) = dpll_div8 function is selected. (1, 0, 0) = tadc_dath function is selected. (1, 1, 0) = i2c0_scl function is selected. (1, 1, 1) = Reserved.
[2]	MFP [2]	P2.2 Alternate Function Select Bit Bits EXT[2] (SYS_P2_MFP[18]), ALT[5] (SYS_P2_MFP[10]), and MFP[2] (SYS_P2_MFP[2]) determine the P2.2 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_clk function is selected. (0, 1, 0) = pwm_ch0 function is selected. (0, 1, 1) = Reserved. (1, 0, 0) = tadc_datl function is selected. (1, 1, 0) = spi0_mosi function is selected. (1, 1, 1) = Reserved.
[1]	MFP [1]	P2.1 Alternate Function Select Bit Bits EXT[1] (SYS_P2_MFP[17]), ALT[1] (SYS_P2_MFP[9]), and MFP[1] (SYS_P2_MFP[1]) determine the P2.1 function.

		(0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_mosi function is selected. (0, 1, 0) = pwm_ch7 function is selected. (0, 1, 1) = tadc_vld function is selected. (1, 0, 0) = ll_debug[7] function is selected. (1, 1, 0) = xtl_c2_clk(for quick start) function is selected. (1, 1, 1) = Reserved.
[0]	MFP [0]	P2.0 Alternate Function Select Bit Bits EXT[0] (SYS_P2_MFP[16]), ALT[0] (SYS_P2_MFP[8]), and MFP[0] (SYS_P2_MFP[0]) determine the P2.0 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_miso function is selected. (0, 1, 0) = pwm_ch6 function is selected. (0, 1, 1) = tadc_clk function is selected. (1, 1, 0) = xtl_c1_clk(for quick start) function is selected. (1, 1, 1) = Reserved.

3.1.4.4 SYS_REGLCTL

Register	Offset	R/W	Description	Reset Value
SYS_REGLCTL	SYS_BA+0x40	R/W	Register Write-Protection Control Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	REGLCTL	Register Write-protection Code (Write Only) Some registers have write-protection function. Writing these registers have to disable the protected function by writing the sequence value 0x59, 0x16, 0x88 to this field. After this sequence is completed, the REGLCTL bit will be set to 1 and write-protection registers can be normal write. Register Write-protection Disable Index (Read Only) 0 = Write-protection Enabled for writing protected registers. Any write to the protected register is ignored. 1 = Write-protection Disabled for writing protected registers. Protected registers are listed below:

Register	Address	Note
WDT_CTL	WDT_BA+0x00	
WDT_ALTCTL	WDT_BA+0x04	
IPRST0	RCC_BA+0x04	
EFUSE_CTL	EF_BA+0x00	
EFUSE_TRG	EF_BA+0x1C	
EFUSE_FLASH_PERMISSION	EF_BA+0x78	

Note: The bits which are write-protected will be noted as" (Write Protect)" beside the description.

3.1.4.5 SYS_STATUS

Register	Offset	R/W	Description	Reset Value
SYS_ STATUS	SYS_BA+0x44	R/W	ROM Status Register	0x0000_0000

Bits	Description	
[31:0]	Reserved	Reserved.

3.1.4.6 SYS_CTRL0

Register	Offset	R/W	Description	Reset Value
SYS_CTRL0	SYS_BA+0x48	R/W	System Control0 Register	0x3F3F_FFC0

Bits	Description	
[31:0]	Reserved	Reserved.

3.1.4.7 SYS_CTRL1

Register	Offset	R/W	Description	Reset Value
SYS_CTRL1	SYS_BA+0x4C	R/W	System Control1 Register	0x0004_0004

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	Valid_removal_cycles	For usb insert detection usage The conditions for detecting an insertion event by the USB deivce: first in a removal state, and then DIP&DIM=0 is detected On the removal state, DIP=DIM=1 must maintain the specified number of 48Mhz cycles before a valid removal state is recognized.

3.1.5 System Timer (SysTick)

The PAN101x series MCU includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used as a Real Time Operating System (RTOS) tick timer or as a simple counter.

When system timer is enabled, it will count down from the value in the SysTick Current Value Register (SYST_VAL) to zero, and reload (wrap) to the value in the SysTick Reload Value Register (SYST_LOAD) on the next clock edge, and then decrement on subsequent clocks. When the counter transitions to zero, the COUNTFLAG status bit is set. The COUNTFLAG bit clears on reads.

The SYST_VAL value is UNKNOWN on reset. Software should write to the register to clear it to zero before enabling the feature. This ensures the timer to count from the SYST_LOAD value rather than an arbitrary value when it is enabled.

If the SYST_LOAD is zero, the timer will be maintained with a current value of zero after it is reloaded with this value. This mechanism can be used to disable the feature independently from the timer enable bit.

3.1.5.1 System Timer Control Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SCS Base Address: SCS_BA = 0xE000_E000				
SYST_CTRL	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000
SYST_LOAD	SCS_BA+0x14	R/W	SysTick Reload Value Register	0x00XX_XXXX
SYST_VAL	SCS_BA+0x18	R/W	SysTick Current Value Register	0x00XX_XXXX

3.1.5.2 System Timer Control Register Description

SysTick Control and Status Register (SYST_CTRL)

Register	Offset	R/W	Description	Reset Value
SYST_CTRL	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000

Bits	Description	
[31:17]	Reserved	Reserved.
[16]	COUNTFLAG	System Tick Counter Flag Returns 1 if timer counted to 0 since last time this register was read. COUNTFLAG is set by a count transition from 1 to 0. COUNTFLAG is cleared on read or by a write to the Current Value register.
[15:3]	Reserved	Reserved.
[2]	CLKSRC	System Tick Clock Source Select Bit 0 = Clock source is optional, refer to STCLKSEL.(This function is not implemented. This bit must be set to 1). 1 = Core clock used for SysTick timer.
[1]	TICKINT	System Tick Interrupt Enable Bit 0 = Counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to 0 has occurred. 1 = Counting down to 0 will cause the SysTick exception to be pended. Clearing the SysTick Current Value register by a register write in software will not cause SysTick to be pended.
[0]	ENABLE	System Tick Counter Enable Bit 0 = Counter Disabled. 1 = Counter Enabled and will operate in a multi-shot manner.

SysTick Reload Value Register (SYST_LOAD)

Register	Offset	R/W	Description	Reset Value
SYST_LOAD	SCS_BA+0x14	R/W	SysTick Reload Value Register	0x00XX_XXXX

Bits	Description	
[31:24]	Reserved	Reserved.
[23:0]	RELOAD	System Tick Reload Value Value to load into the Current Value register when the counter reaches 0.

SysTick Current Value Register (SYST_VAL)

Register	Offset	R/W	Description	Reset Value
SYST_VAL	SCS_BA+0x18	R/W	SysTick Current Value Register	0x00XX_XXXX

Bits	Description	
[31:24]	Reserved	Reserved.
[23:0]	CURRENT	System Tick Current Value Current counter value. This is the value of the counter at the time it is sampled. The counter does not provide read-modify-write protection. The register is write-clear. A software write of any value will clear the register to 0. Unsupported bits RAZ (see SysTick Reload Value Register).

3.1.6 Nested Vectored Interrupt Controller (NVIC)

3.1.6.1 Overview

The PAN101x series MCU provides an interrupt controller as an integral part of the exception mode, named as “Nested Vectored Interrupt Controller (NVIC)”, which is closely coupled to the processor core and provides following features.

3.1.6.2 Features

- Nested and Vectored interrupt support
- Automatic processor state saving and restoration
- Dynamic priority change
- Reduced and deterministic interrupt latency

The NVIC prioritizes and handles all supported exceptions. All exceptions are handled in “Handler Mode”. This NVIC architecture supports 32 (IRQ[31:0]) discrete interrupts with 4 levels of priority. All of the interrupts and most of the system exceptions can be configured to different priority levels. When an interrupt occurs, the NVIC will compare the priority of the new interrupt to the current running one’s priority. If the priority of the new interrupt is higher than the current one, the new interrupt handler will override the current handler.

When an interrupt is accepted, the starting address of the Interrupt Service Routine (ISR) is fetched from a vector table in memory. There is no need to determine which interrupt is accepted and branch to the starting address of the correlated ISR by software. While the starting address is fetched, NVIC will also automatically save processor state including the registers “PC, PSR, LR, R0~R3, R12” to the stack. At the end of the ISR, the NVIC will restore the mentioned registers from stack and resume the normal execution. Thus it will take less and deterministic time to process the interrupt request.

The NVIC supports “Tail Chaining” which handles back-to-back interrupts efficiently without the overhead of states saving and restoration and therefore reduces delay time in switching to pending ISR at the end of current ISR. The NVIC also supports “Late Arrival” which improves the efficiency of concurrent ISRs. When a higher priority interrupt request occurs before the current ISR starts to execute (at the stage of state saving and starting address fetching), the NVIC will give priority to the higher one without delay penalty. Thus it advances the real-time capability.

3.1.6.3 Exception Model and System Interrupt Map

The following table lists the exception model supported by NuMicro® Mini58 series. Software can set four levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as 0 and the lowest priority is denoted as 3. The default priority of all the user-configurable interrupts is 0. Note that the priority 0 is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

Table 3-2 Exception Model

Exception Name	Vector Number	Priority
Reset	1	-3
NMI	2	-2
Hard Fault	3	-1
Reserved	4 ~ 10	Reserved
SVCall	11	Configurable
Reserved	12 ~ 13	Reserved
PendSV	14	Configurable
SysTick	15	Configurable
Interrupt (IRQ0 ~ IRQ31)	16 ~ 47	Configurable

Table 3-3 System Interrupt Map Vector Table

Exception Number	Interrupt Number (Bit In Interrupt Registers)	Interrupt Name	Source Module	Interrupt Description
1 ~ 15	31	Reserved	-	-
16	30	LP	Low Power	LP interrupt
17	29	SLPTMR	SLPTMR	SLPTMR interrupt
18	28	BOD	BOD	BOD interrupt
19	27	DMA	DMA	DMA interrupt
20	26	Reserved	-	-
21	25	Reserved	-	-
22	24	Reserved	-	-
23	23	CLKTRIM	CLKTRIM	CLKTRIM interrupt
24	22	USB	USB	USB interrupt
25	21	USB_DMA	USB_DMA	USB_DMA interrupt
26	20	Reserved	-	-
27	19	Reserved	-	-
28	18	GPIO_P3	GPIO	External signal interrupt from GPIO group P30，P31
29	17	GPIO_P2	GPIO	External signal interrupt from GPIO group P2
30	16	GPIO_P1	GPIO	External signal interrupt from GPIO group P1
31	15	GPIO_P0	GPIO	External signal interrupt from GPIO group P0
32	14	TIMER2	TIMER2	Timer 2 interrupt

33	13	TIMER1	TIMER1	Timer 1 interrupt
34	12	UART1	UART1	UART1 interrupt
35	11	LL	LL	Link Layer interrupt
36	10	SPI1	SPI1	SPI1 interrupt
37	9	Reserved	-	-
38	8	TIMER0	TIMER0	Timer0 interrupt
39	7	WWDT	WWDT	WWDT interrupt
40	6	WDT	WDT	WDT interrupt
41	5	ADC	ADC	ADC interrupt
42	4	PWM0	PWM0	PWM0 interrupt
43	3	UART0	UART0	UART0 interrupt
44	2	Reserved	-	-
45	1	SPI0	SPI0	SPI0 interrupt
46	0	I2C0	I2C0	I2C0 interrupt

3.1.6.4 Vector Table

When an interrupt is accepted, the processor will automatically fetch the starting address of the interrupt service routine (ISR) from a vector table in memory. For ARMv6-M, the vector table based address is fixed at 0x00000000. The vector table contains the initialization value for the stack pointer on reset, and the entry point addresses for all exception handlers. The vector number on previous page defines the order of entries in the vector table associated with the exception handler entry as illustrated in previous section.

Table 3-4 Vector Table Format

Vector Table Word Offset (Bytes)	Description
0x00	Initial Stack Pointer Value
Exception Number * 0x04	Exception Entry Pointer using that Exception Number

3.1.6.5 Operation Description

NVIC interrupts can be enabled and disabled by writing to their corresponding Interrupt Set-Enable or Interrupt Clear-Enable register bit-field. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current enabled state of the corresponding interrupts. When an interrupt is disabled, interrupt assertion will cause the interrupt to become Pending; however, the interrupt will not be activated. If an interrupt is Active when it is disabled, it remains in its Active state until cleared by reset or an exception return. Clearing the enable bit prevents new activations of the associated interrupt.

NVIC interrupts can be pended/un-pended using a complementary pair of registers to those used to enable/disable the interrupts, named the Set-Pending Register and Clear-Pending Register respectively. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current pended state of the corresponding interrupts. The Clear-Pending Register has no effect on the execution status of an Active interrupt.

NVIC interrupts are prioritized by updating an 8-bit field within a 32-bit register (each register supporting four interrupts).

The general registers associated with the NVIC are all accessible from a block of memory in the System Control Space and will be described in next section.

3.1.6.6 NVIC Control Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SCS Base Address: SCS_BA = 0xE000_E000				
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31 Set-Enable Control Register	0x0000_0000
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31 Clear-Enable Control Register	0x0000_0000
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31 Set-Pending Control Register	0x0000_0000
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31 Clear-Pending Control Register	0x0000_0000
NVIC_IPR0	SCS_BA+0x400	R/W	IRQ0 ~ IRQ3 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR1	SCS_BA+0x404	R/W	IRQ4 ~ IRQ7 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR2	SCS_BA+0x408	R/W	IRQ8 ~ IRQ11 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR3	SCS_BA+0x40C	R/W	IRQ12 ~ IRQ15 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR4	SCS_BA+0x410	R/W	IRQ16 ~ IRQ19 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR5	SCS_BA+0x414	R/W	IRQ20 ~ IRQ23 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR6	SCS_BA+0x418	R/W	IRQ24 ~ IRQ27 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR7	SCS_BA+0x41C	R/W	IRQ28 ~ IRQ31 Interrupt Priority Control Register	0x0000_0000

3.1.6.7 NVIC Control Register Description

IRQ0 ~ IRQ31 Set-Enable Control Register (NVIC_ISER)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31 Set-Enable Control Register	0x0000_0000

Bits	Description	
[31:0]	SETENA	Interrupt Enable Bits Enable one or more interrupts. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Write Operation: 0 = No effect. 1 = Write 1 to enable associated interrupt. Read Operation: 0 = Associated interrupt status Disabled. 1 = Associated interrupt status Enabled. **Note:** Read value indicates the current enable status.

IRQ0 ~ IRQ31 Clear-Enable Control Register (NVIC_ICER)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31 Clear-Enable Control Register	0x0000_0000

Bits	Description	
[31:0]	CLRENA	Interrupt Disable Bits Disable one or more interrupts. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Write Operation: 0 = No effect. 1 = Write 1 to disable associated interrupt. Read Operation: 0 = Associated interrupt status is Disabled. 1 = Associated interrupt status is Enabled. **Note:** Read value indicates the current enable status.

IRQ0 ~ IRQ31 Set-Pending Control Register (NVIC_ISPR)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31 Set-Pending Control Register	0x0000_0000

Bits	Description	
[31:0]	SETPEND	Set Interrupt Pending Bits Write Operation: 0 = No effect. 1 = Write 1 to set pending state. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Read Operation:

		0 = Associated interrupt in not in pending status. 1 = Associated interrupt is in pending status. **Note:** Read value indicates the current pending status.

IRQ0 ~ IRQ31 Clear-Pending Control Register (NVIC_ICPR)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31 Clear-Pending Control Register	0x0000_0000

Bits	Description	
[31:0]	CLRPEND	Clear Interrupt Pending Bits Write Operation: 0 = No effect. 1 = Write 1 to clear pending state. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Read Operation: 0 = Associated interrupt is not in pending status. 1 = Associated interrupt is in pending status. **Note:** Read value indicates the current pending status.

IRQ0 ~ IRQ3 Interrupt Priority Register (NVIC_IPR0)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR0	SCS_BA+0x400	R/W	IRQ0 ~ IRQ3 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_3	Priority of IRQ3 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_2	Priority of IRQ2 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_1	Priority of IRQ1 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_0	Priority of IRQ0 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ4 ~ IRQ7 Interrupt Priority Register (NVIC_IPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR1	SCS_BA+0x404	R/W	IRQ4 ~ IRQ7 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_7	Priority of IRQ7 0 denotes the highest priority and 3 denotes the lowest priority.

[29:24]	Reserved	Reserved.
[23:22]	PRI_6	Priority of IRQ6 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_5	Priority of IRQ5 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_4	Priority of IRQ4 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ8 ~ IRQ11 Interrupt Priority Register (NVIC_IPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR2	SCS_BA+0x408	R/W	IRQ8 ~ IRQ11 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_11	Priority of IRQ11 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_10	Priority of IRQ10 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_9	Priority of IRQ9 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_8	Priority of IRQ8 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ12 ~ IRQ15 Interrupt Priority Register (NVIC_IPR3)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR3	SCS_BA+0x40C	R/W	IRQ12 ~ IRQ15 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_15	Priority of IRQ15 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_14	Priority of IRQ14 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_13	Priority of IRQ13 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_12	Priority of IRQ12

		0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ16 ~ IRQ19 Interrupt Priority Register (NVIC_IPR4)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR4	SCS_BA+0x410	R/W	IRQ16 ~ IRQ19 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_19	Priority of IRQ19 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_18	Priority of IRQ18 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_17	Priority of IRQ17 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_16	Priority of IRQ16 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ20 ~ IRQ23 Interrupt Priority Register (NVIC_IPR5)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR5	SCS_BA+0x414	R/W	IRQ20 ~ IRQ23 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_23	Priority of IRQ23 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_22	Priority of IRQ22 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_21	Priority of IRQ21 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_20	Priority of IRQ20 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ24 ~ IRQ27 Interrupt Priority Register (NVIC_IPR6)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR6	SCS_BA+0x418	R/W	IRQ24 ~ IRQ27 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_27	Priority of IRQ27 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_26	Priority of IRQ26 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_25	Priority of IRQ25 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_24	Priority of IRQ24 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

IRQ28 ~ IRQ31 Interrupt Priority Register (NVIC_IPR7)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR7	SCS_BA+0x41C	R/W	IRQ28 ~ IRQ31 Interrupt Priority Control Register	0x0000_0000

Bits	Description	
[31:30]	PRI_31	Priority of IRQ31 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRI_30	Priority of IRQ30 0 denotes the highest priority and 3 denotes the lowest priority.
[21:16]	Reserved	Reserved.
[15:14]	PRI_29	Priority of IRQ29 0 denotes the highest priority and 3 denotes the lowest priority.
[13:8]	Reserved	Reserved.
[7:6]	PRI_28	Priority of IRQ28 0 denotes the highest priority and 3 denotes the lowest priority.
[5:0]	Reserved	Reserved.

3.1.6.8 System Control Block Registers (SCB)

The PAN101x series MCU status and operating mode control are managed System Control Block Registers. Including CPUID, PAN101x series MCU interrupt priority and PAN101x series MCU power management can be controlled through these system control registers.

3.1.6.9 System Control Block Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SCS Base Address: SCS_BA = 0xE000_E000				
SCS_CPUID	SCS_BA+0xD00	R	CPUID Base Register	0x410C_C200
SCS_ICSR	SCS_BA+0xD04	R/W	Interrupt Control State Register	0x0000_0000
SCS_AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000
SCS_SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000
SCS_SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000
SCS_SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

3.1.6.10 System Control Block Register Description

CPUID Base Register (SCS_CPUID)

Register	Offset	R/W	Description	Reset Value
SCS_CPUID	SCS_BA+0xD00	R	CPUID Base Register	0x410C_C200

Bits	Description	
[31:24]	IMPLEMENTER	Implementer Code Implementer code assigned by M6_core.
[23:20]	Reserved	Reserved.
[19:16]	PART	Architecture of the Processor Read as 0xC for M6_core parts.
[15:4]	PARTNO	Part Number of the Processor Read as 0xC20.
[3:0]	REVISION	Revision Number Read as 0x0.

Interrupt Control State Register (SCS_ICSR)

Register	Offset	R/W	Description	Reset Value
SCS_ICSR	SCS_BA+0xD04	R/W	Interrupt Control State Register	0x0000_0000

Bits	Description	
[31]	NMIPENDSET	NMI Set-pending Bit Write Operation: 0 = No effect. 1 = Changes NMI exception state to pending. Read Operation: 0 = NMI exception is not pending. 1 = NMI exception is pending. **Note**: Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it detects a write of 1 to this bit. Entering the handler then clears this bit to 0. This means a read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
[30:29]	Reserved	Reserved.
[28]	PENDSVSET	PendSV Set-pending Bit Write Operation: 0 = No effect. 1 = Changes PendSV exception state to pending. Read Operation: 0 = PendSV exception is not pending. 1 = PendSV exception is pending. **Note**: Writing 1 to this bit is the only way to set the PendSV exception state to pending.
[27]	PENDSVCLR	PendSV Clear-pending Bit Write Operation: 0 = No effect. 1 = Removes the pending state from the PendSV exception. **Note**: This bit is write-only. To clear the PENDSV bit, you must “write 0 to PENDSVSET and write 1 to PENDSVCLR” at the same time.
[26]	PENDSTSET	SysTick Exception Set-pending Bit Write Operation: 0 = No effect. 1 = Changes SysTick exception state to pending. Read Operation: 0 = SysTick exception is not pending. 1 = SysTick exception is pending.
[25]	PENDSTCLR	SysTick Exception Clear-pending Bit Write Operation: 0 = No effect. 1 = Removes the pending state from the SysTick exception. **Note**: This bit is write-only. When you want to clear PENDST bit, you must “write 0 to PENDSTSET and write 1 to PENDSTCLR” at the same time.
[24]	Reserved	Reserved.
[23]	ISRPREEMPT	Interrupt Preemption Bit If set, a pending exception will be serviced on exit from the debug halt state.

		Note: This bit is read-only.
[22]	ISRPENDING	Interrupt Pending Flag, Excluding NMI and Faults 0 = Interrupt not pending. 1 = Interrupt pending. **Note**: This bit is read-only.
[21]	Reserved	Reserved.
[20:12]	VECTPENDING	Exception Number of the Highest Priority Pending Enabled Exception 0 = No pending exceptions. Non-zero = Exception number of the highest priority pending enabled exception. **Note**: These bits are read-only.
[11:9]	Reserved	Reserved.
[8:0]	VECTACTIVE	Contains the Active Exception Number 0 = Thread mode. Non-zero = Exception number of the currently active exception. **Note**: These bits are read-only.

Application Interrupt and Reset Control Register (SCS_AIRCR)

Register	Offset	R/W	Description	Reset Value
SCS_AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000

Bits	Description	
[31:16]	VECTORKEY	Register Access Key Write Operation: When writing to this register, the VECTORKEY field need to be set to 0x05FA, otherwise the write operation would be ignored. The VECTORKEY filed is used to prevent accidental write to this register from resetting the system or clearing of the exception status. Read Operation: Read as 0xFA05.
[15:3]	Reserved	Reserved.
[2]	SYSRESETREQ	System Reset Request Writing this bit 1 will cause a reset signal to be asserted to the chip to indicate a reset is requested. The bit is a write only bit and self-clears as part of the reset sequence.
[1]	VECTCLRACTIVE	Exception Active Status Clear Bit Reserved for debug use. When writing to the register, user must write 0 to this bit, otherwise behavior is unpredictable.
[0]	Reserved	Reserved.

System Control Register (SCS_SCR)

Register	Offset	R/W	Description	Reset Value
SCS_SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000

Bits	Description	
[31:5]	Reserved	Reserved.

[4]	SEVONPEND	Send Event On Pending Bit 0 = Only enabled interrupts or events can wake-up the processor, disabled interrupts are excluded. 1 = Enabled events and all interrupts, including disabled interrupts, can wake-up the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects next WFE. The processor also wakes up on execution of an SEV instruction or an external event.
[3]	Reserved	Reserved.
[2]	SLEEPDEEP	Processor Deep Sleep and Sleep Mode Selection Controls whether the processor uses sleep or deep sleep as its low power mode: 0 = Sleep mode. 1 = Deep Sleep mode.
[1]	SLEEPONEXIT	Sleep-on-exit Enable This bit indicates sleep-on-exit when returning from Handler mode to Thread mode: 0 = Do not sleep when returning to Thread mode. 1 = Enter Sleep, or Deep Sleep, on return from ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.
[0]	Reserved	Reserved.

System Handler Priority Register 2 (SCS_SHPR2)

Register	Offset	R/W	Description	Reset Value
SCS_SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000

Bits	Description	
[31:30]	PRSH_11	Priority Of System Handler 11 – SVCall 0 denotes the highest priority and 3 denotes the lowest priority.
[29:0]	Reserved	Reserved.

System Handler Priority Register 3 (SCS_SHPR3)

Register	Offset	R/W	Description	Reset Value
SCS_SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

Bits	Description	
[31:30]	PRSH_15	Priority of System Handler 15 – SysTick 0 denotes the highest priority and 3 denotes the lowest priority.
[29:24]	Reserved	Reserved.
[23:22]	PRSH_14	Priority of System Handler 14 – PendSV 0 denotes the highest priority and 3 denotes the lowest priority.
[21:0]	Reserved	Reserved.

3.2 Power Management

3.2.1 Overview

The PAN101x series' low power design chapter contains the following function:

- System power design
- Low power system
- Low power mode

3.2.2 System Power

3.2.2.1 Power Supply

Table 3-5 Power Supply Illustration

Name	Direction	Voltage(V)	Note
VBAT	I	1.8-3.6	System 3V power supply input
VOUT_BK	O	1.5(typ) 1.4-3.6	BUCK power supply output1(VCC_xxx)
VCC_DIG	I	1.4-3.6	Digital (HLDO) VCC input
VCC_ANA	I	1.4-3.6	Analog VCC input
VCC_RF	I	1.4-3.6	RF VCC input
DVDD_PAD	O	1.2(typ)	HLDO output, connect with a capacitor

3.2.3 Low Power System

3.2.3.1 Low power system introduction

For the entire system, please pay attention to the following power consumption :

1) System operating power consumption which includes processor, memory (Flash & RAM), peripherals, clock, analog circuit, etc.
2) Standby power consumption which includes leakage of the circuit, clock circuit (such as 32K), running peripherals (wake-up modules such as WDT), analog circuits (IO, POR, small LDO, etc.) and RAM data storage requirements.
3) System wake-up delay which means the time required to wake up the CPU from sleep mode.

PAN101x series is mainly battery-powered and interrupt-driven in many applications. When there is no interruption or data which needs to be processed, the entire system enters sleep mode. When there is an interrupt request, the CPU will be waken up, processes the interrupt, and then goes to sleep after completion. The whole schedule can refer to Figure 3-1.

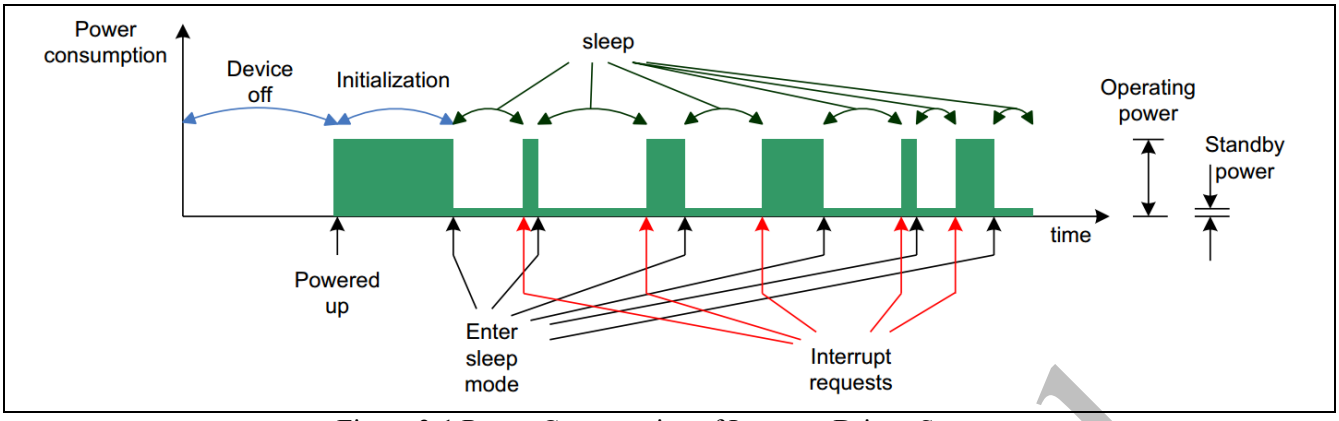


Figure 3-1 Power Consumption of Interrupt Driven System

3.2.3.2 Low Power Mode of MCU

Sleep Mode

There are two kinds of sleep mode: normal sleep mode and deep sleep mode, which can be configured by setting the CPU's internal registers.

According to the configurations, PAN101x series can choose different low power mode.

1. Turn off part or all of the clock.
2. Reduce the clock frequency.
3. Reduce voltage.
4. Turn off part of the power supply; (Standby mode can be set by cutting off the DVDD power supply).

There are two ways to enter sleep mode by configure the SCR register.

1. Wait for WFI instruction.
2. Use the Sleep-On-Exit function.

Table 3-6 System Control Register SCR (0xE000ED10)

Bits	Field	R/W	Description	Reset Value
[31:5]	Reserved	-	Reserved	-
[4]	SEVONPEND	R/W	When set to 1, an event is generated for each new pending of an interrupt. This can be used to wake up the processor if Wait-for-Event (WFE) sleep is used.	0
[3]	Reserved	-	Reserved	-
[2]	SLEEPDEEP	R/W	When set to 1, deep sleep mode is selected when sleep mode is entered. When this bit is 0, normal sleep mode is selected when sleep mode is entered.	0
[1]	SLEEPONEXIT	R/W	When set to 1, enter sleep mode (Wait-for-Interrupt) automatically when exiting an exception handler and returning to thread level. When set to 0, this feature is disabled.	0
[0]	Reserved	-	Reserved	-

If SLEEPDEEP bit is set to 0, the system will enter Normal Sleep mode. If set 1, it will enter

Deep Sleep.

WFI

On the one hand, WFI instruction can be waken up by the higher-level interrupts. On the other hand, it can be waken up by debug requests.

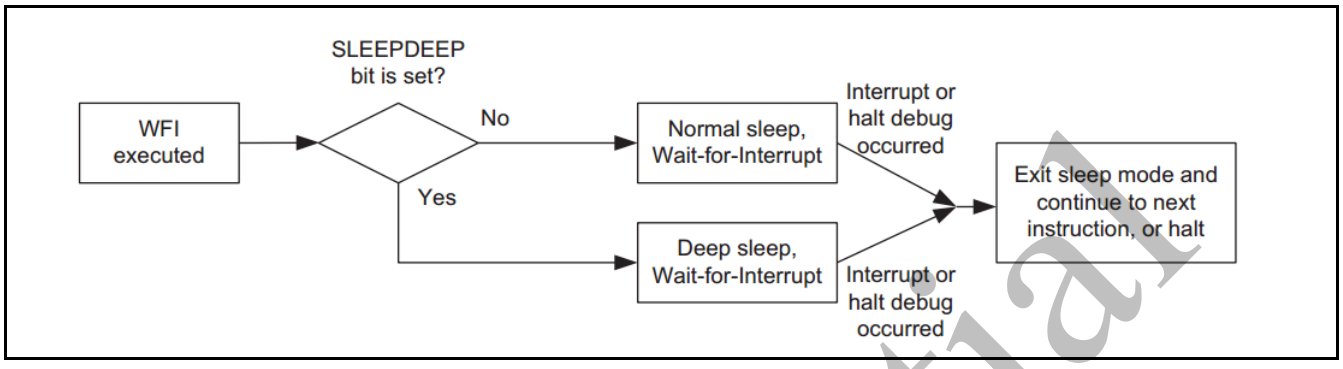


Figure 3-2 WFI Schedule

When the WFI instruction is executed, or the Sleep-on-Exit function is used to enter the sleep mode, the CPU stops executing the instruction and waits for the interrupt request. Only if the priority of the interrupt request is higher than the current priority can the CPU be awakened. WFI wake-up behavior is shown in Table 3-7.

Table 3-7 WFI Wake-up Behavior

WFI Behavior	Wakeup	ISR Execution
PRIMASK cleared		
IRQ priority > current level	Y	Y
IRQ priority ≤ current level	N	N
PRIMASK set (interrupt disabled)		
IRQ priority > current level	Y	N
IRQ priority ≤ current level	N	N

The role of PRIMASK is to shield all interrupt sources. When the WFI enters sleep mode and PRIMASK is set to 1, if a new interrupt request with higher priority enters, the new interrupt can wake up the system, but the interrupt program will not be executed. The interrupt program can be executed until PRIMASK is cleared. The application scenario of this function is shown in Figure 3-3.

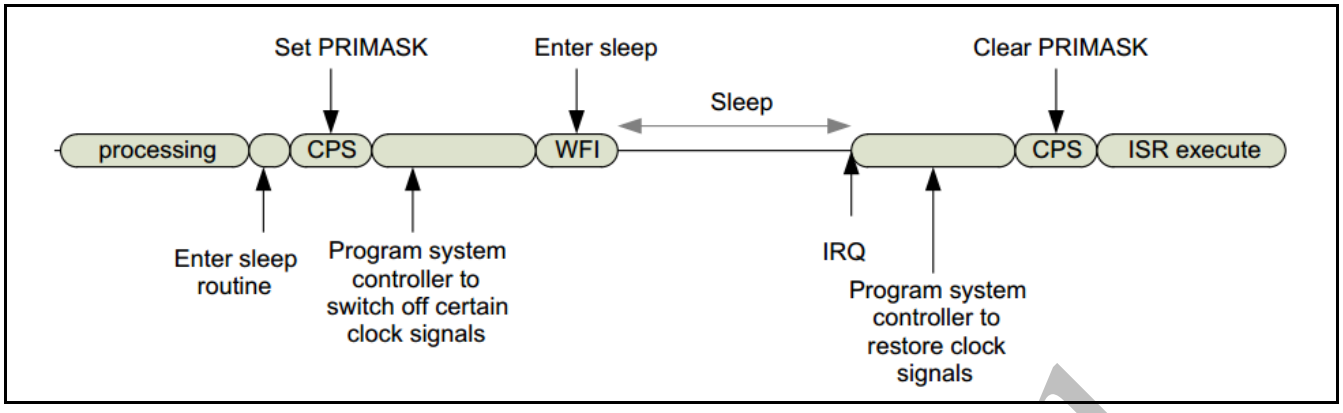


Figure 3-3 Execute RRIMASK during Sleep Mode

The function can be used to turn off all clocks before entering sleep mode, turn on the clocks after the system waken up, and then process the interrupt program. This function can save more power during sleep mode.

Sleep-On-Exit Function

Sleep-On-Exit function: if the system enters sleep mode, the interrupt wakes up system will sleep once finishing processing the interrupt program without sending WFI commands. This function is mainly used for interrupt-oriented applications (IoT applications are mainly interrupt-oriented). The specific process is shown in Figure 3-4. The running effect is shown in Figure 3-5.

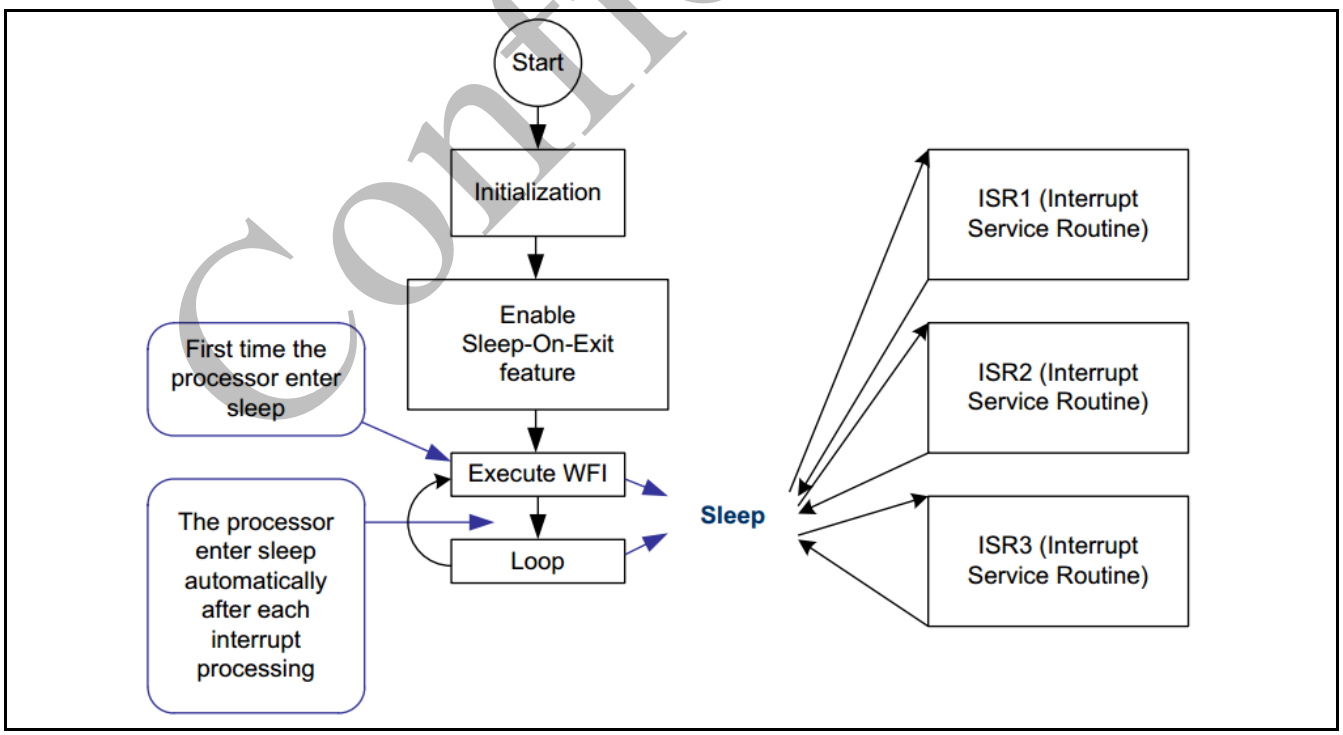


Figure 3-4 Sleep-On-Exit Process

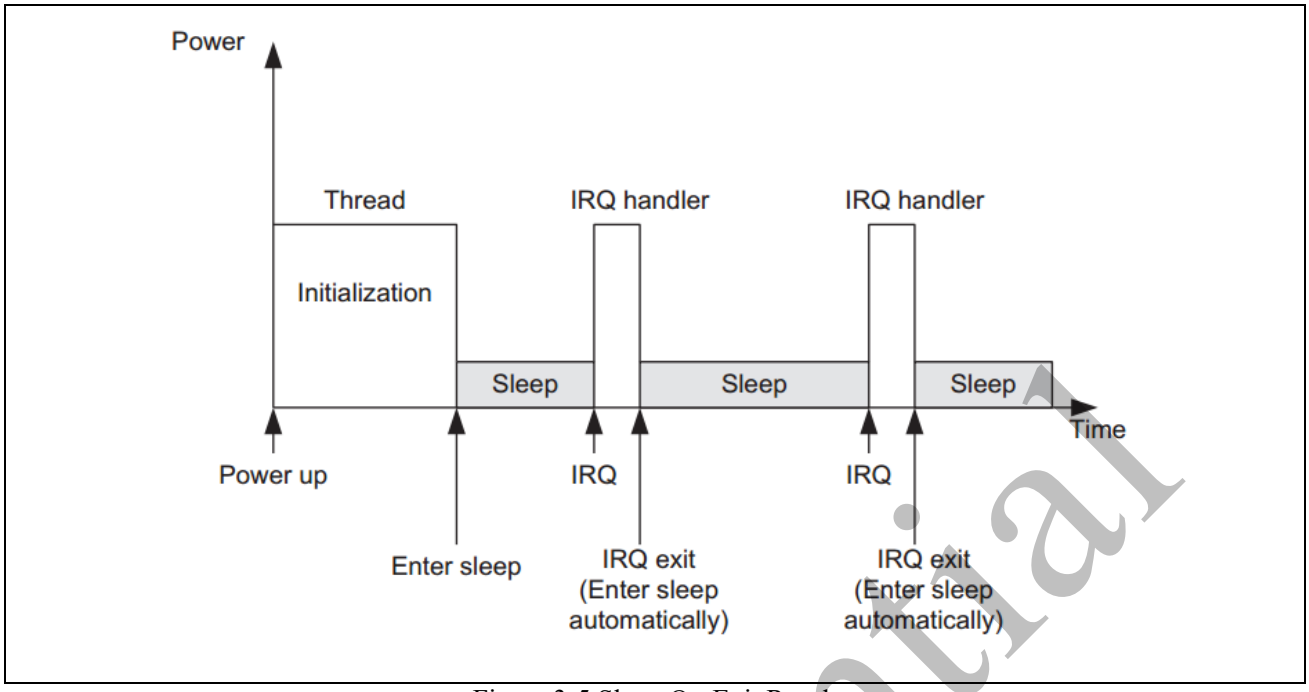


Figure 3-5 Sleep-On-Exit Result

The Sleep-On-Exit function reduces the working time of the processor and reduces the power consumption of stacking and popping between interrupts. The Sleep-On-Exit function is controlled by the SLEEPONEXIT bit in the SCR register. In interrupt-driven applications, please set this register at the end of initialization.

3.2.4 Low Power Mode

Mode	Enter Low Power Config	Wakeup Source	Clock	Retention Region
STANDBY _M0	Sleep_mode = 3 Buck_en_ctrl=0 Buck_bp_ctrl=0 Flashldo_bp_en_ctrl =0 Flashldo_lp_en_en_ctrl =0 WFI	P00，P01，P02 BOD/LVR PIN RESET	None	None
STANDBY _M1	Sleep_mode = 2 ldo_pwr_ctrl = 0 ldol_ pwr _ctrl = 0/1 cpu_ pwr _ctrl = 0/1 sram0/1_ pwr _ctrl = 0/1 ll_ram_ pwr _ctrl=0/1 phy_ram_ pwr _ctrl=0/1 Buck_en_ctrl=0 Buck_bp_ctrl=0 Flashldo_bp_en_ctrl =0 Flashldo_lp_en_en_ctrl =0 WFI	All GPIO SLPTMR WDT BOD/LVR PIN RESET	CLK32K	LPLDOL/H: LL_RAM(selectable) PHY_RAM/PHY_REGS(selectable) SRAM0/1(selectable) decrypt_ram(selectable) LPLDOL/H: asnactrl_l, rcc regs GPIO WDT BOD，LVR

DEEPSLEEP	Sleep_mode = 1 ldol_power_ctrl = 0/1 Buck_en_ctrl=0 Buck_bp_ctrl=0 Flashldo_bp_en_ctrl =0/1 Flashldo_lp_en_en_ctrl =1/0 WFI	All GPIO SLPTMR WDT TIMER0/1/2 BOD/LVR PIN RESET	CLK32K	LPLDOL/H：all logic modules LPLDOL/H：Wakeup GPIO WDT Timer0/1/2(be carefull when use)
SLEEP	Sleep_mode = 0 WFI	All Peri Interrupt BOD/LVR PIN RESET	CLK32K High Frequence Clock	HP_LDO: all logic modules

3.2.5 Cpu Vector Table Remap

Support cpu vector table remap function.Cpu_addr_remap_en (Cpu_addr_remap_ctrl[31]) is the enable control of this function, Cpu_remap_addr(Cpu_addr_remap_ctrl[24:0]) is the remap address of the cpu vector table.

3.2.6 BUCK Low Power Config

The output voltage with different mode of buck:

Bucken	Bypass	Vout(v)
0	X	Floating
1	0	1.5
1	1	3

Buck(bypass) low power config:

State	Bucken	Bypass
Work	1	1
Low power (deepsleep/standby_m0/m1)	00(suggested)	
Wakeup	1	1
Work	1	1

Buck(normal) low power config:

State	Bucken	Bypass
Work	1	0
Low power (deepsleep/standby_m0/m1)	00(suggested)	
Wakeup	1	0
Work	1	0

3.2.7 FLASH Low Power Config

The output voltage with different mode of flash ldo:

En	Bypass	Vout
x	1	The supply voltage(3.3V)
1	0	Ldo(1.5-2.0V)(suggested 1.6V)
0	0	Floating

Flash ldo (bypass) low power config:

State	Lp_en	En	Bypass
Work	0	X	1
Low power（deepsleep）	0	0	1
Low power（standby_m1/m0）	0	0	0
Wakeup	0	X	1
Work	0	X	1

Flash ldo (normal) low power config:

State	Lp_en	En	Bypass
Work	0	1	0
Low power（deepsleep）	1	0	0
Low power（standby_m1/m0）	0	0	0
Wakeup	0	1	0
Work	0	1	0

3.2.8 Register Map

Register	Offset	R/W	Description	Reset Value
ANA Base Address: ANA_BA = 0x4007_0000				
LP_REG_SYNC	ANA_BA+0x00	R/W	Slow clock region regs sync control	0x0000_0000
LP_FL_CTRL	ANA_BA+0x04	R/W	Low power control	0x0000_0048
LP_SPACING_TIME0	ANA_BA+0x08	R/W	The spcing time0 of slptmr	0x0000_0000
LP_SPACING_TIME1	ANA_BA+0x0C	R/W	The spcing time1 of slptmr	0x0000_0000
LP_SPACING_TIME2	ANA_BA+0x10	R/W	The spcing time2 of slptmr	0x0000_0000
LP_SLPTMR	ANA_BA+0x14	R	SLPTMR value	0xxxxx_xxxx
LP_INT_CTRL	ANA_BA+0x18	R/W	Low power interrupt control	0x0001_0001
LP_DLY_CTRL	ANA_BA+0x1C	R/W	Startup delay control	0x003F_0408
LP_PATA_POLY	ANA_BA+0x20	R/W	PTAT & POLY control	0x0003_01a1
LP_HPLDO	ANA_BA+0x24	R/W	HPLDO control	0x007B_0241
LP_LPLDO	ANA_BA+0x28	R/W	LPLDO control	0x001D_0010
LP_ANALDO	ANA_BA+0x2C	R/W	ANALDO control	0x0000_0041
LP_FSYNLDO	ANA_BA+0x30	R/W	FSYNLDO control	0x0000_0040
LP_SW	ANA_BA+0x34	R/W	Power switch control	0x0000_01FF
LP_BUCK	ANA_BA+0x38	R/W	BUCK control	0x0002_4923
ANA_ADCLDO	ANA_BA+0x3C	R/W	ADCLDO control	0x0000_0040
ANA_RFFELDO	ANA_BA+0x40	R/W	RFFELDO control	0x0000_0040
ANA_VCOLDO	ANA_BA+0x44	R/W	VCOLDO control	0x0000_0040
ANA_DFT	ANA_BA+0x48	R/W	Analog DFT control	0x0F00_0000
ANA_MISC	ANA_BA+0x4C	R/W	Analog MISC control	0x0000_3AD3
ANA_RESERVED	ANA_BA+0x50	R/W	Analog reserved control	0xFF00_FF00
ACT_32K_CTRL	RCC_BA+0x54	R/W	active 32K clock control	0x0000_0000
ACT_32K_BASECORR	RCC_BA+0x58	R/W	active 32K clock base correction	0x0000_0000
CPU_ADDR_REMAP_CTRL	RCC_BA+0x5C	R/W	Cpu address remap control	0x0000_0000
RCL_HW_CAL_CTRL	RCC_BA+0x60	R/W	RCL clock hardware calibration control	0x0000_0000

3.2.9 Register Description

3.2.9.1 LP_REG_SYNC

Register	Offset	R/W	Description	Reset Value
LP_REG_SYNC	ANA_BA+0x00	R/W	Slow clock region regs sync control	0x0000_0000

Bits	Description		
[31:2]	Reserved		Reserved.
[1]	REG_SYNC_TRG	R/W	The trigger of fast clock and slow clock. Set it to 1 and wait it clear to 0 by hardware, when reg_sync is enable
[0]	REG_SYNC	R/W	regs sync control between fast clock and slow clock 0：the sync disable 1：the sync enable

Note: all need reg_sync regs is written whit suffix ***_AON

3.2.9.2 LP_FL_CTRL

Register	Offset	R/W	Description	Reset Value
LP_FL_CTRL	ANA_BA+0x04	R/W	Low power control	0x0000_0048

Bits	Description		
[31]	LDOL_PWR_CTRL	R/W	The power switch between LPLDOH and LPLDOL, at low power mode 0: disable 1: enable
[30]	LDO_PWR_CTRL	R/W	The power switch between HPLDO and LPLDOL, at low power mode 0: disable 1: enable
[29]	Reserved		Reserved
[28]	LL_SRAM_PWR_CTRL	R/W	The power switch of LL DATA SRAM, at low power mode 0: disable 1: enable
[27]	PHY_RAM_PWR_CTRL	R/W	The power switch of PHY SEQ SRAM and PHY REGS, at low power mode 0: disable 1: enable
[26]	CPU_PWR_CTRL	R/W	The power switch of decrypt ram, at low power mode 0: disable 1: enable
[25]	SRAM1_PWR_CTRL	R/W	The power switch of SRAM1, at low power mode 0: disable 1: enable
[24]	SRAM0_PWR_CTRL	R/W	The power switch of SRAM0, at low power mode 0: disable 1: enable
[23]	PMU_EN_CTRL	R/W	The low power mode control of PMU modules(POLY, PTAT, ANALDO, FSYNLDO) 0: disable 1: enable
[22]	HPLDO_EN_CTRL	R/W	The low power mode control of HPLDO 0: disable 1: enable
[21]	Reserved	R/W	Reserved
[20]	RCL_EN_CTRL_AON	R/W	The low power mode control of RC 0: disable 1: enable (should enable high voltage sync)
[19]	XTL_EN_CTRL_AON	R/W	The low power mode control of XTL 0: disable 1: enable (should enable high voltage sync)

[18]	XTH_EN_CTRL	R/W	The low power mode control of XTH 0: disable 1: enable
[17]	RCH_EN_CTRL	R/W	The low power mode control of RCH 0: disable 1: enable
[16]	DPLL_EN_CTRL	R/W	The low power mode control of DPLL 0: disable 1: enable
[15]	BKEN_EN_CTRL	R/W	The low power mode control of BKEN 0: disable 1: enable
[14]	BKBP_EN_CTRL	R/W	The low power mode control of BKBP 0: disable 1: enable
[13]	FLASHLDO_LP_EN_EN_CTRL	R/W	The low power mode control of FLASHLDO_LP_EN 0: disable 1: enable
[12]	FLASHLDO_BP_EN_CTRL	R/W	The low power mode control of FLASHLDO_BP 0: disable 1: enable
[11:8]	Reserved		Reserved
[7]	PWR_CTRL_EN	R/W	Low power control mode 0: automatic, bit[30],[23:21],[18:14] is controled by hardware; others controlled by software 1: manumatic, bit[31:12] is controlled by software
[6]	SLOW_CLK_EN	R/W	Reserved
[5]	LPLDOLH_ISO_EN	R/W	The isolate control bwtween LPLDOL and LPLDOH, active at Standby_m1，deepsleep mode 0: isolate disable 1: isolate enable（timer，pwm can not work）
[4]	CPU_RET_EN		Cpu retention enable 0: disable 1: enable
[3]	SLPTMR_EN	R/W	SLPTMR enable control 0: disable 1: enable
[2]	EXT_IO_WK_LEVEL_SEL_AON	R/W	Extent IO wake up level select 0: low level wake up 1: high level wake up Active @standby_m0 mode for P00，P01，P02 (should enable high voltage sync)
[1:0]	SLEEP_MODE	R/W	00: sleep 01: deepsleep 10: standby_m1 11: standby_m0 Active after WFI

3.2.9.3 LP_SPACING_TIME0

Register	Offset	R/W	Description	Reset Value
LP_SPACING_TIME0	ANA_BA+0x08	R/W	The spcing time0 of slptmr	0x0000_0000

Bits	Description		
[31：0]	SPACING_TIME	R/W	The spacing time of slptmr. Unit: 31.2us

3.2.9.4 LP_SPACING_TIME1

Register	Offset	R/W	Description	Reset Value
LP_SPACING_TIME1	ANA_BA+0x0C	R/W	The spcing time1 of slptmr	0x0000_0000

Bits	Description		
[31：0]	SPACING_TIME	R/W	The spacing time of slptmr. Unit: 31.2us

3.2.9.5 LP_SPACING_TIME2

Register	Offset	R/W	Description	Reset Value
LP_SPACING_TIME2	ANA_BA+0x10	R/W	The spcing time2 of slptmr	0x0000_0000

Bits	Description		
[31：0]	SPACING_TIME	R/W	The spacing time of slptmr. Unit: 31.2us

3.2.9.6 LP_SLPTMR

Register	Offset	R/W	Description	Reset Value
LP_SLPTMR	ANA_BA+0x14	R	SLPTMR value	0xxxxx_xxxx

Bits	Description		
[31:0]	SLPTMR_VALUE	R	slptmr value

3.2.9.7 LP_INT_CTRL

Register	Offset	R/W	Description	Reset Value
LP_INT_CTRL	ANA_BA+0x18	R/W	Low power interrupt control	0x0001_0001

Bits	Description		
[31:21]	Reserved		Reserved.
[20]	SLPTMR_WK_EN	R/W	slptmr wakeup enable control 0: disable 1: enable
[19]	SLPTMR2_FLG	R	slptmr flag, write 1 to clear this flag to 0
[18]	SLPTMR1_FLG	R	slptmr flag, write 1 to clear this flag to 0
[17]	SLPTMR0_FLG	R	slptmr flag, write 1 to clear this flag to 0
[16]	SLPTMR_INT_EN	R/W	slptmr interrupt enable control
[15:5]	Reserved		Reserved
[4]	SRAM_RET_FLG	R	sram retention flag, write 1 to clear this flag to 0
[3]	STANDBY_M0_FLG	R	Standby_m0 mode flag, write 1 to clear this flag to 0
[2]	STANDBY_M1_FLG	R	Standby_m1 mode flag, write 1 to clear this flag to 0
[1]	DEEPSLEEP_FLG	R	Deepsleep mode flag, write 1 to clear this flag to 0
[0]	LP_INT_EN	R/W	Low power interrupt enable control Must be 1 @standby_m0 mdoe

3.2.9.8 LP_DLY_CTRL

Register	Offset	R/W	Description	Reset Value
LP_DLY_CTRL	ANA_BA+0x1C	R/W	Startup delay control	0x003F_0408

Bits	Description		
[31:26]	Reserved		Reserved.
[25:16]	DLY1_TIME_AON	R/W	Used for standby_m0 mode for wait LPLDOH ready. Clock source is Slow clk (should enable high voltage sync)
[15:12]	Reserved		Reserved.
[11]	HPLDO_RDY_BYP	R/W	0: should wait HPLDO ready 1: do not care HPLDO ready
[10]	DLY2_TIME_EN	R/W	Dly_time enable control
[9:0]	DLY2_TIME	R/W	Used for wait fast clock ready. Clock source is Slow clk

3.2.9.9 LP_PTAT_POLY

Register	Offset	R/W	Description	Reset Value
LP_PTAT_POLY	ANA_BA+0x20	R/W	PTAT & POLY control	0x0003_01a1

Bits	Description		
[31:19]	Reserved		Reserved.
[18:16]	IPOLY_TRIM	R/W	The current tune of poly
[15:9]	Reserved		Reserved.
[8]	POLYEN	R/W	Enable POLY 1: enable 0: disable
[7]	PTATCOREFLTRACT	R/W	The filter mode of POLY top bias voltage 1: small filter resister mode 0: large filter resister mode
[6]	PTATCOREFLTRSHRT	R/W	The filter enable of POLY top bias voltage 1: disable 0: enable
[5:3]	PTATVTRIM	R/W	The trim of PTAT voltage
[2:1]	PTAT_TEMPTRIM	R/W	PTAT vbg temp trim
[0]	PTATCOREEN	R/W	PMU-PTAT enable 0: disable 1: enable

3.2.9.10 LP_HPLDO

Register	Offset	R/W	Description	Reset Value
LP_HPLDO	ANA_BA+0x24	R/W	HPLDO control	0x007B_0241

Bits	Description		
[31:24]	Reserved		Reserved.
[22:20]	FLASHLDO_VTRIM	R/W	The voltage trim of Flash LDO 000:1.5V 111:2.2V 100mv each step
[19]	FLASHLDO_CAP_SEL	R/W	The external cap control of Flash LDO 1: ext-cap active 0: inside cap
[18]	FLASHLDO_LP_EN	R/W	The enable control of Flash LDO LP mode 1: enable 0: disable
[17]	FLASHLDO_EN_AON	R/W	The enable control of Flash LDO (should enable high voltage sync) 1: enable 0: disable
[16]	FLASHLDO_BYPASS_AON	R/W	The control of Flash LDO bypass(should enable high voltage sync) 1: bypass 0: normal, the output voltage from LDO
[15:10]	Reserved		Reserved.
[9]	HPLDO_SEL	R/W	The select of external cap 1: ext-cap active 0: inside cap
[8:7]	Reserved		Reserved.
[6:3]	HPLDO_VTRIM	R/W	The voltage trim of HPLDO
[2]	HPLDO_BYPASS	R/W	The control of HPLDO bypas 1: bypass 0: normal, the output voltage from LDO
[1]	Reserved		Reserved.
[0]	HPLDO_EN	R/W	The enable control of HPLDO 1: enable 0: disable

3.2.9.11 LP_LPLDO

Register	Offset	R/W	Description	Reset Value
LP_LPLDO	ANA_BA+0x28	R/W	LPLDO control	0x001D_0010

Bits	Description		
[31:25]	Reserved		Reserved.
[24]	LPLDOH_MODE_AON	R/W	The mode of LPLDOH 1: for low power. can reduce 0.2uA of LPLDOH 0: normal mode (should enable high voltage sync)
[23:21]	LPLDOH_VREF_TRIM_AON	R/W	The voltage trim of LPLDOH mode2(Lpldoh_mode_AON is1): 000: 300mV 001: 250mV ... 111: 650mV (should enable high voltage sync)
[20:17]	LPLDOH_DVDDSEL_AON	R/W	The voltage trim of LPLDOH (should enable high voltage sync)
[16]	LPLDOH_EN_AON	R/W	The enable control of LPLDOH (should enable high voltage sync) 1: enable 0: disable
[15:5]	Reserved		Reserved.
[4:1]	LPLDOL_DVDDSEL	R/W	The voltage trim of LPLDOL
[0]	LPLDOL_EN	R/W	The enable control of LPLDOL 1: enable 0: disable

3.2.9.12 LP_ANALDO

Register	Offset	R/W	Description	Reset Value
LP_ANALDO	ANA_BA+0x2C	R/W	ANALDO control	0x0000_0041

Bits	Description		
[31:7]	Reserved		Reserved.
[6:3]	ANALDOVTRIM	R/W	The voltage trim of ANA LDO
[2:1]	Reserved		Reserved.
[0]	ANALDOEN	R/W	Enable ANA LDO 1: enable 0: disable

3.2.9.13 LP_FSYNLDO

Register	Offset	R/W	Description	Reset Value
LP_FSYNLDO	ANA_BA+0x30	R/W	FSYNLDO control	0x0000_0040

Bits	Description		
[31:7]	Reserved		Reserved.
[6:3]	fsynldovtrim	R/W	The voltage trim of FSYN LDO 4‘b0000 0.986V 4’b1000 1.2V 4‘b1111 1.38V
[2:1]	Reserved		Reserved.
[0]	fsynldoen	R/W	Enable FSYN LDO 1: enable 0: disable

3.2.9.14 LP_SW

Register	Offset	R/W	Description	Reset Value
LP_SW	ANA_BA+0x34	R/W	Power switch control	0x0000_01FF

Bits	Description		
[31:9]	Reserved		Reserved.
[8]	DVDDCPU_EN	R/W	The power switch control of dvddcpu 1: enable 0: disable
[7]	DVDDPHY_EN	R/W	The power switch control of dvddphy 1: enable 0: disable
[6]	DVDDLL_EN	R/W	The power switch control of dvddll 1: enable 0: disable
[5]	VDDFLASH_EN	R/W	The power switch control of Flash 1: enable 0: disable
[4]	DVDDLPLDO_EN	R/W	The power switch control of dvddlpldo 1: enable 0: disable
[3]	DVDDSRAM1_EN	R/W	The power switch control of dvddsram1 1: enable 0: disable
[2]	DVDDSRAM0_EN	R/W	The power switch control of dvddsram0 1: enable 0: disable
[1]	HPDVDD_EN	R/W	The power switch control of dvdd
[0]	HPDVDD_PD	R/W	11: enable 10: enable 01: disable and pulldown hpldo 00: disable and hpldo floating

3.2.9.15 LP_BUCK

Register	Offset	R/W	Description	Reset Value
LP_BUCK	ANA_BA+0x38	R/W	BUCK control	0x00042123

Bits	Description		
[31:26]	Reserved		Reserved.
[25]	BK_VLMT_EN_AON	R/W	The control of BUCK voltage limit 0: disable 1: enable (should enable high voltage sync)
[24]	BK_CAL_OUT	R	The output of BUCK calibration
[23:22]	BK_CAL_EN_AON	R/W	The automatic calibration control of BUCK DCDC 自校准使能 2b'00: disable 2b'01: nouse 2b'10: ZERO CMP calibration enable 2b'11: IMAX CMP calibration enable (should enable high voltage sync)
[21]	Reserved		Reserved.
[20]	BK_TEST_EN	R/W	The control of BUCK test 0: disable 1: enable
[19]	BK_MODE_SEL_AON	R/W	BUCK mode control 0: PFM 1: PWM (should enable high voltage sync)
[18:14]	BK_ZERCAL_AON	R/W	The mamutical calibration option of zero detection (should enable high voltage sync)
[13:9]	BK_IMXCAL_AON	R/W	The manutical calibration option of overcurrent detection (should enable high voltage sync)
[8:6]	BK_IMAX_AON	R/W	The current limit option (should enable high voltage sync)
[5:2]	BK_VOUT_AON	R/W	The trim of BUCK Vout (should enable high voltage sync)
[1]	BK_EN_AON	R/W	Buck enable control 0: disable 1: enable (should enable high voltage sync) Note: the signal is 1.2V for analog, to 3V just for retention @lp mode
[0]	BK_BP_AON	R/W	BUCK Bypass control 0: disable 1: enable (should enable high voltage sync) Note: the signal is 1.2V for analog, to 3V just for retention @lp mode

3.2.9.16 ANA_ADCLDO

Register	Offset	R/W	Description	Reset Value
ANA_ADCLDO	ANA_BA+0x3C	R/W	ADCLDO control	0x0000_0040

Bits	Description		
[31:7]	Reserved		Reserved.
[6:3]	RXADCLDOVTRIM	R/W	The output voltage trim of RXADC LDO 4‘b0000: 0.986V 4’b1000: 1.2V 4‘b1111: 1.38V
[2:1]	Reserved		Reserved.
[0]	RXADCLDOEN	R/W	The enable control RXADC LDO 1: enable 0: disable

3.2.9.17 ANA_RFFELDO

Register	Offset	R/W	Description	Reset Value
ANA_RFFELDO	ANA_BA+0x40	R/W	RFFELDO control	0x0000_0040

Bits	Description		
[31:7]	Reserved		Reserved.
[6:3]	RFFELDOVTRIM	R/W	The output voltage trim of RFFE LDO 4‘b0000: 0.986V 4’b1000: 1.2V 4‘b1111: 1.38V
[2:1]	Reserved		Reserved.
[0]	RFFELDOEN	R/W	The enable control RFFE LDO 1: enable 0: disable

3.2.9.18 ANA_VCOLDO

Register	Offset	R/W	Description	Reset Value
ANA_VCOLDO	ANA_BA+0x44	R/W	VCOLDO control	0x0000_0040

Bits	Description		
[31:7]	Reserved		Reserved.
[6:3]	FSYNVCOLDOVTRIM	R/W	The output voltage trim of FSYN LDO 4‘b0000: 0.986V 4’b1000: 1.2V 4‘b1111: 1.38V
[2:1]	Reserved		Reserved.
[0]	FSYNVCOLDOEN	R/W	The enable control FSYN LDO 1: enable 0: disable

3.2.9.19 ANA_DFT

Register	Offset	R/W	Description	Reset Value
ANA_DFT	ANA_BA+0x48	R/W	Analog DFT control	0x0F00_0000

Bits	Description		
[31:24]	BATTERY_TEST_TRIM	R/W	load trim
[23]	BATTERY_TEST_SEL	R/W	battery test select,1:constant current,0:constant resistance
[22]	BATTERY_TEST_EN	R/W	battery test enable,1:enable,0:disable
[21:17]	Reserved		Reserved.
[16:15]	PAD_TST_SEL	R/W	register for DFT pad select,00 :test IQ 01:test DFT iby DFT PAD,:10:test DFT in GPIO PAD,11:not uesd
[14]	DFT_EN_V	R/W	Enable DFT voltage output mode
[13]	DFT_EN_I	R/W	Enable DFT current output mode
[12]	DFT_EN_CLK	R/W	Enable DFT clock output mode
[11]	DFT_FLAG	R/W	Enable DFT flag output mode
[10:7]	DFT_V_SEL	R/W	Registers for selecting proper output voltage
[6]	DFT_I_SEL	R/W	Registers for selecting proper output current
[5:3]	DFT_CLK_SEL	R/W	Registers for selecting proper output clock
[2:1]	DFT_FLAG_SEL	R/W	Registers for selecting proper output flag
[0]	REG_V_DFT_BYPASS	R/W	register for DFT voltage output mode buffer bypass function

3.2.9.20 ANA_MISC

Register	Offset	R/W	Description	Reset Value
ANA_MISC	ANA_BA+0x4C	R/W	Analog MISC control	0x0000_3AD3

Bits	Description		
[31:29]	Reserved		Reserved.
[28]	USB_PU2	R/W	USB PAD pullup2 enable
[27]	USB_PU	R/W	USB PAD pullup enable
[26]	USB_EN	R/W	USB PAD enable control
[25:20]	FLASH_PAD_PDEN	R/W	Flash PAD pulldown enable control(wp,so,si,sclk,hold,cs)
[19:14]	FLASH_PAD_PUEN	R/W	Flash PAD pullup enable control(wp,so,si,sclk,hold,cs)
[13:8]	FLASH_PAD_DIEN	R/W	Flash PAD input enable control(wp,so,si,sclk,hold,cs)
[7]	LS_EN_GLOBAL_HPLDO	R/W	The enable control of level shift between HPLDO region and analog supply voltage region 1: enable 0: disable
[6]	LS_EN_GLOBAL_AON	R/W	The enable control of level shift between LPLDOH region and analog supply voltage region 1: enable 0: disable
[5]	Reserved		Reserved.
[4]	PMUVREFBUFEN	R/W	The enable control of VBG_BUFFER 1: enable 0: PowerDown
[3]	PMULDO_TM	R/W	ldo test mode(all ldo reuse)
[2:0]	RST_VREF1P2V_TRIM_AON	R/W	Trim of rst vref trim (should enable high voltage sync)

3.2.9.21 ANA_RESERVED

Register	Offset	R/W	Description	Reset Value
ANA_RESERVED	ANA_BA+0x50	R/W	Analog reserved control	0xFF00_FF00

Bits	Description		
[31:24]	Reserved_hv_h_aon	R/W	Reserved (should enable high voltage sync)
[23:16]	Reserved_hv_l_aon	R/W	Reserved (should enable high voltage sync)
[15:8]	Reserved_lv_h	R/W	Reserved
[7:0]	Reserved_lv_l	R/W	Reserved

3.2.9.22 ACT_32K_CTRL

Register	Offset	R/W	Description	Reset Value
ACT_32K_CTRL	RCC_BA+0x54	R/W	active 32K clock control	0x0000_0000

Bits	Description		
[31:22]	ACT_32K_FINECORR	R/W	The fine correct value of ACT CNT, units is 32MHz
[21:12]	ACT_32K_FINEOFFSET	R	The offset between SLP_32K and ACT_32K, units is 32MHz
[11:4]	Reserved		Reserved
[3]	ACT_32K_UP_DONE	R	Active clock update done flag 0: slow clk is being used 1: done, active clock is being used
[2]	Reserved		Reserved
[1]	ACT_32K_SEL	R/W	Link layer 32k clock source select 0: SLP_CLK (32K RC or 32K XTL) 1: ACT_32K
[0]	ACK_32K_EN	R/W	Active 32k clock enable 0: clock disable 1: clock enable

3.2.9.23 ACT_32K_BASECORR

Register	Offset	R/W	Description	Reset Value
ACT_32K_BASECORR	RCC_BA+0x58	R/W	active 32K clock base correction	0x0000_0000

Bits	Description		
[31:0]	ACT_32K_BASECORR	R/W	The base correct value of ACT CNT, units is 32KHz

3.2.9.24 CPU_ADDR_REMAP_CTRL

Register	Offset	R/W	Description	Reset Value
CPU_ADDR_REMAP_CTRL	RCC_BA+0x5C	R/W	Cpu address remap control	0x0000_0000

Bits	Description		
[31]	CPU_ADDR_REMAP_EN	R/W	Cpu address remap enable control 0: disable 1: enable, cpu address 0-255Byte will be remap to the releated region define by cpu_remap_addr
[30:24]	Reserved		Reserved
[23:0]	CPU_REMAP_ADDR	R/W	The start address of remap to cpu address 0.(unit is 256B)

3.2.9.25 RCL_HW_CAL_CTRL

Register	Offset	R/W	Description	Reset Value
RCL_HW_CAL_CTRL	RCC_BA+0x60	R/W	RCL clock hardware calibration control	0x0000_0000

Bits	Description		
[31:10]	Reserved		Reserved
[9]	HW_CAL_BUSY	R	Rcl hardware calibration is going on.
[8]	HW_CAL_EN	R/W	Rcl hardware calibration enable
[7:0]	HW_CAL_SPACING_TIME	R/W	The spacing time of enable rcl hardware calibrarion. Unit: 32ms

3.3 Reset and Clock Controller (RCC)

3.3.1 Overview

The RCC module can implement reset function and generate clocks for the whole chip, including system clocks and all peripheral clocks.

3.3.2 Reset

Reset Source	Trigger Condition	Reset Region	Flag
POR	Power on reset and power down reset (threshold voltage: 1.65V)	Whole chip	PORRF
PAD_RESET	Pull down Reset PIN	Whole chip without ROMFLAG	PINRF
LVR	Low voltage reset (threshold voltage: 1.8V)	Whole chip without ROMFLAG and itself control logic	LVRRF
BOD	Brown-out Detector Reset (threshold voltage: 1.75V-2.95V)	Whole chip without ROMFLAG and itself control logic	BODRF
CHIP	Software config releated reg	Whole chip without ROMFLAG	CHIPRF
WDT	Count reachs the target time	Reset region is selecable: 1. low voltage of digital region without rcc and anactrl modules 2. Whole chip without ROMFLAG	WDTRF
WWDT	Count reachs the target time	low voltage of digital region without rcc and anactrl modules	WDTRF
SYS	Software config releated reg	low voltage of digital region without rcc, cpu debug and anactrl modules	SYSRF
CPU	Software config releated reg	Just reset cpu, icache, fmc and decryption modules (do not sugget use)	CPURF
MODULE RESET	Software config releated reg	Reset of each function module	-

3.3.2.1 Reset Block Diagram

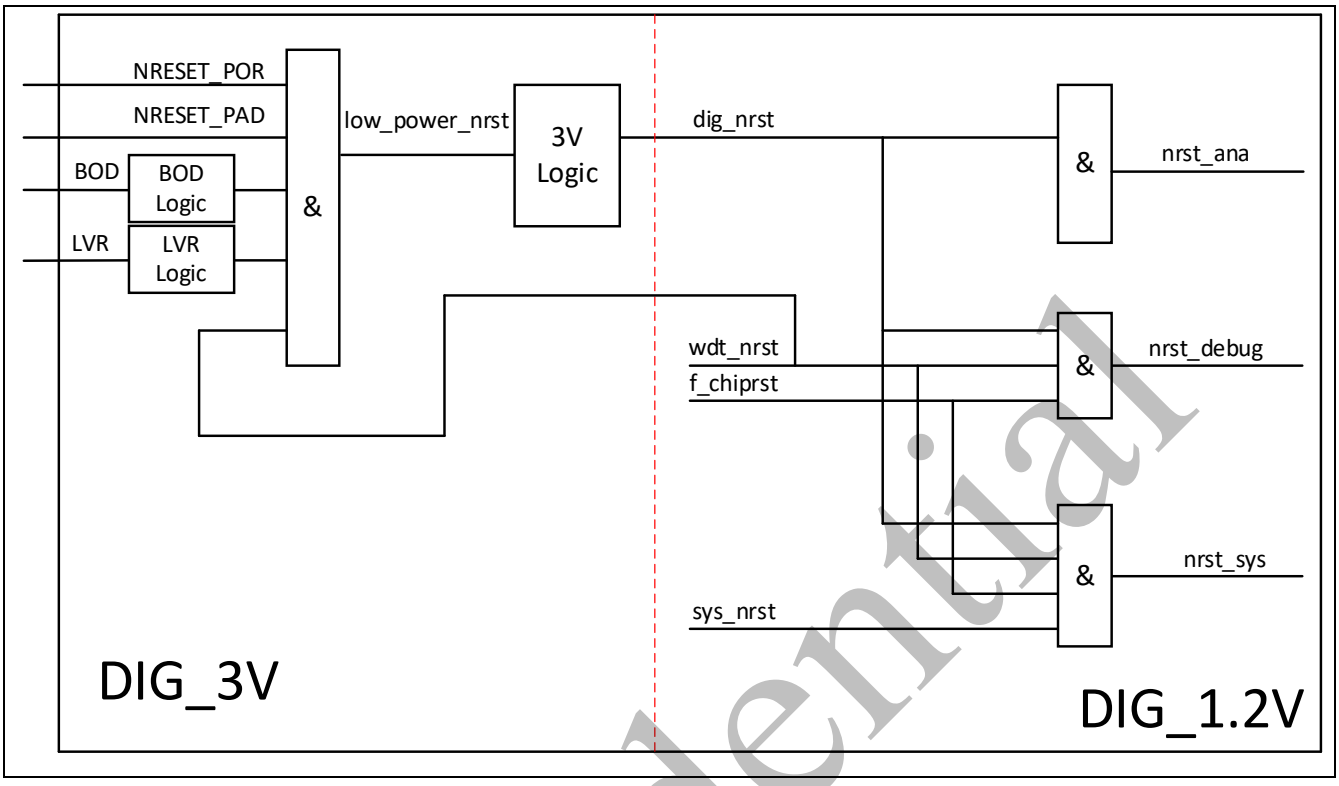


Figure 3-6 Reset Block Diagram

Nrst_ana: Used to reset rcc and anactrl

Nrst_debug: Used to reset CPU debug

Nrst_sys: Used to reset all 1.2V logic except CPU debug, rcc, anactrl

3.3.2.2 nRESET Reset

The nRESET reset means to generate a reset signal by pulling low nRESET pin, which is an asynchronous reset input pin and can be used to reset system at any time. When the nRESET voltage is lower than 0.2 VDD and the state keeps longer than 36 us (glitch filter), chip will be reset. The nRESET reset will control the chip in reset state until the nRESET voltage rises above 0.7 VDD and the state keeps longer than 36 us (glitch filter). The PINRF (SYS_RSTSTS[1]) will be set to 1 if the previous reset source is nRESET reset. Figure 3-7 shows the nRESET reset waveform.

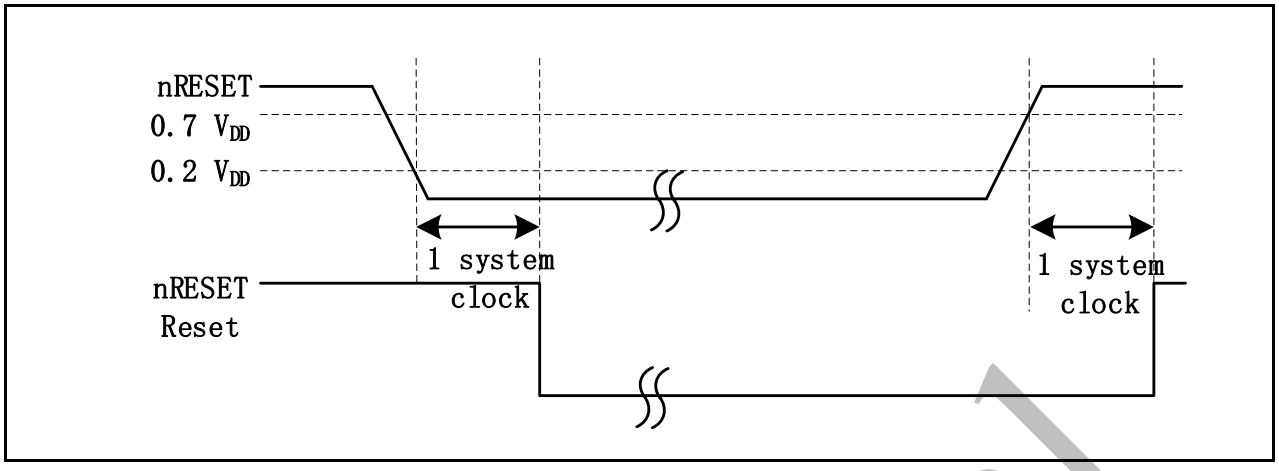


Figure 3-7 nRESET Reset Waveform

3.3.2.3 Power-On Reset (POR)

The Power-on reset (POR) is used to generate a stable system reset signal and forces the system to be reset when power-on to avoid unexpected behavior of MCU. When applying the power to MCU, the POR module will detect the rising voltage and generate reset signal to system until the voltage is ready for MCU operation. At POR reset, the PINRF (SYS_RSTSTS[1]) will be set to 1 to indicate there is a POR reset event. The PINRF (SYS_RSTSTS[1]) bit can be cleared by writing 1 to it. Figure 3-8 shows the waveform of Power-On reset.

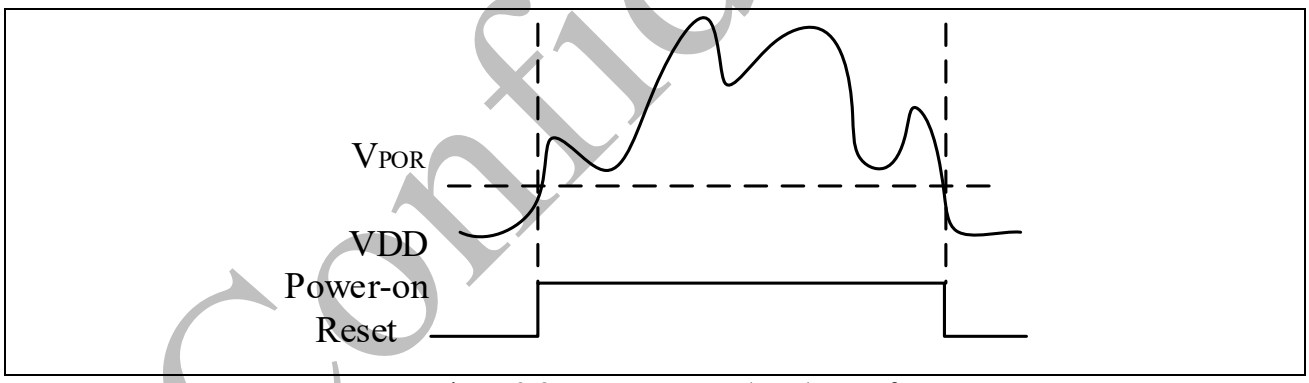


Figure 3-8 Power-on Reset (POR) Waveform

3.3.2.4 Low Voltage Reset (LVR)

Low Voltage Reset detects AVDD during system operation. When the AVDD voltage is lower than VLVR and the state keeps longer than De-glitch time (see *BLDBCTL* register), chip will be reset. The LVR reset will control the chip in reset state until the AVDD voltage rises above VLVR and the state keeps longer than De-glitch time. The PINRF (SYS_RSTSTS[1]) will be set to 1 if the previous reset source is nRESET reset. Figure 3-9 shows the Low Voltage Reset waveform.

T1: < debounce time, invalid

T2: low voltage > debounce time, lvr pulled low

T3: high voltage > debounce time, lvr pulled high

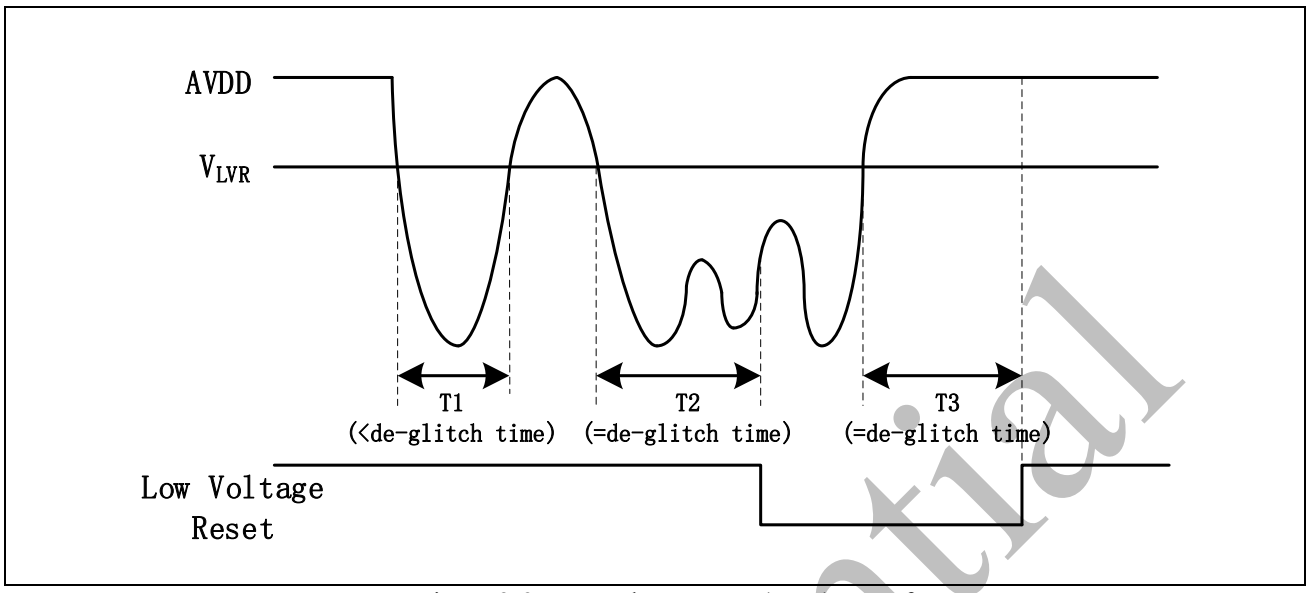


Figure 3-9 Low Voltage Reset (LVR) Waveform

3.3.2.5 Brown-out Detector Reset (BOD Reset)

If the Brown-out Detector (BOD) function is enabled by setting the Brown-out Detector Enable Bit BODEN (ANAC_RCCCTL [24]), Brown-Out Detector function will detect AVDD during system operation. When the AVDD voltage is lower than VBOD which is decided by BODEN and BODVL and the state keeps longer than De-glitch time (see *BODCTL* register), chip will be reset. The BOD reset will control the chip in reset state until the AVDD voltage rises above VBOD and the state keeps longer than De-glitch time. Figure 3-10 shows the Brown-Out Detector waveform.

T1: < debounce time, invalid

T2: low voltage > debounce time, lvr pulled low

T3: high voltage > debounce time, lvr pulled high

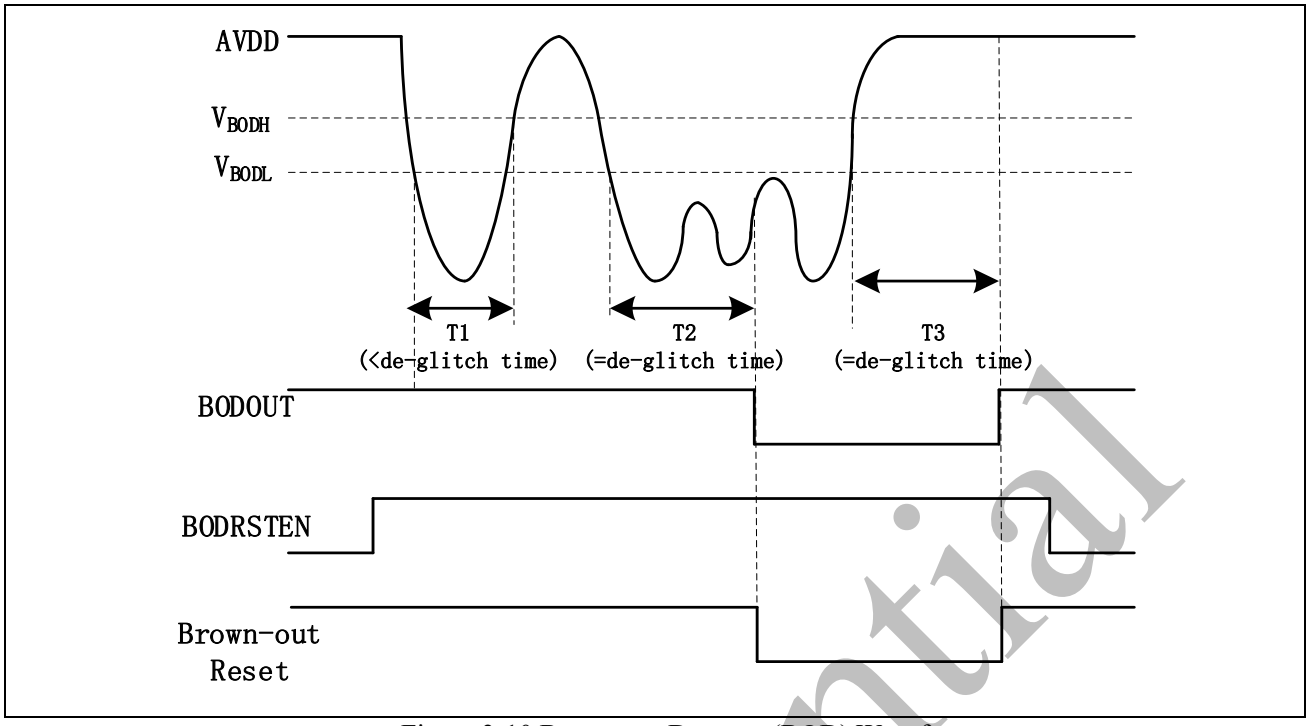


Figure 3-10 Brown-out Detector (BOD) Waveform

3.3.2.6 Watchdog Timer Reset

In most industrial applications, system reliability is very important. To automatically recover the MCU from failure status is one way to improve system reliability. The watch-dog timer (WDT) is widely used to check if the system works fine. If the MCU is crashed or out of control, it may cause the watch-dog time-out. User may decide to enable system reset during watch-dog time-out to recover the system and take action for the system crash/out-of-control after reset.

Software can check if the reset is caused by watch-dog time-out to indicate the previous reset is a watch-dog reset and handle the failure of MCU after watch-dog time-out reset by checking WDTRF (SYS_RSTSTS[2]).

3.3.3 Clock Controller

3.3.3.1 Analog Clock

Analog Clock Source

As shown in Figure 3-11, the analog clock source includes RCH, HXT, RF_DPLL, MCU_DPLL and RCL. The digital clock source is composed of RCH, HXT and MCU_DPLL. RCH is configured as the default clock once powered on.

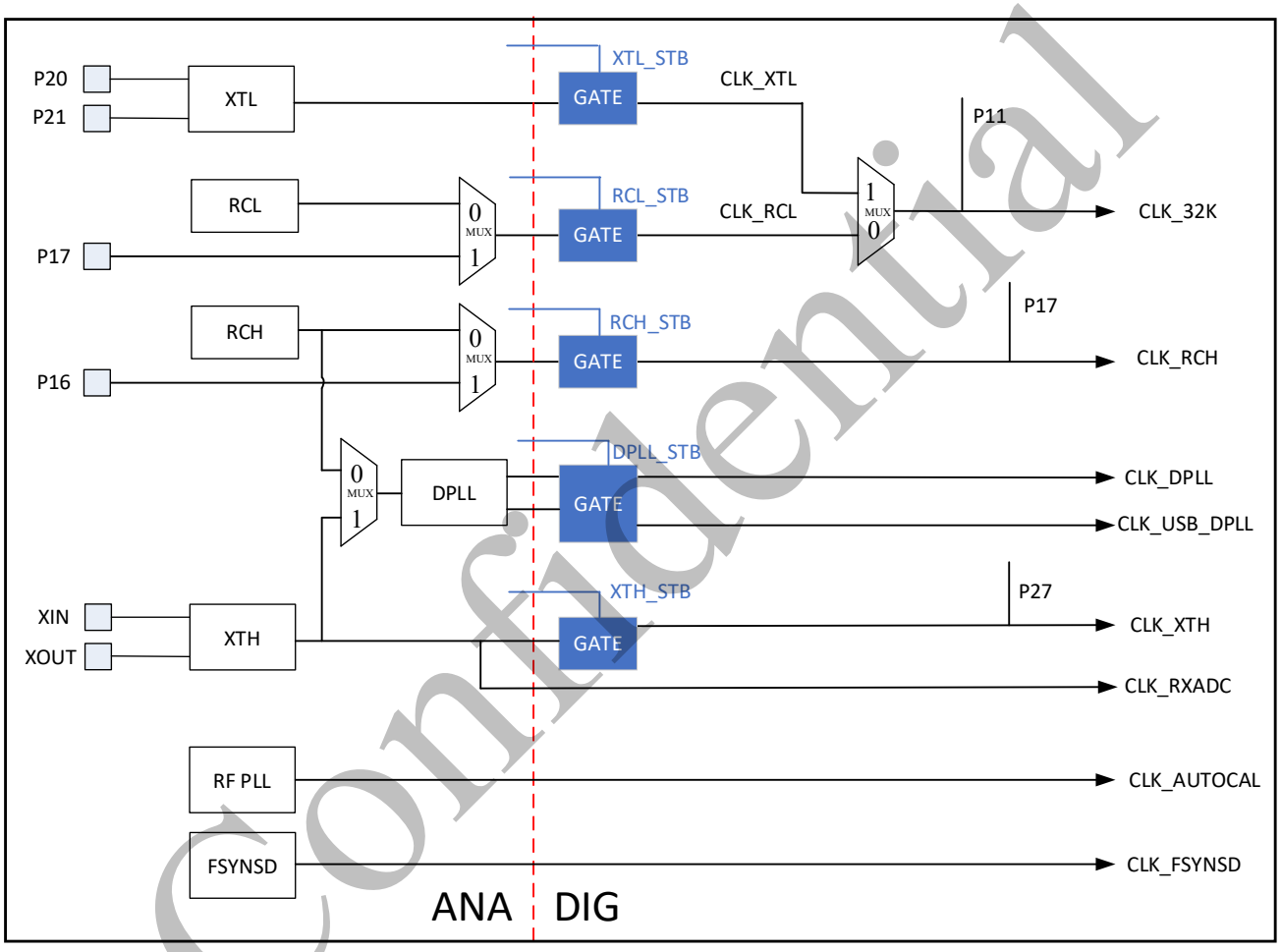


Figure 3-11 System Clock Generator Block Diagram

The analog clock are listed below:

Table 3-8 Analog Clock Source

Name	Analog&digital interface	Source	Frequency	Function
CLK_XTH	phy_clk_xo_fsyn_dig_1p2v	XTH	32MHz	System clock/BLE module clock
CLK_XTL	phy_clk_xo_losc_3p0v	XTL	32.768KHz	System low speed clock
CLK_RCH	phy_clk_xo_losc_3p0v	RCH	32MHz	System clock/BLE module clock
CLK_RCL	phy_clk_rco_losc_3p0v	RCL	32KHz	System low speed clock
CLK_DPLL	phy_clk_pll_1p2v	DPLL	64/48MHz	System clock
CLK_USB_DPLL	phy_clk_pll_usb_1p2v	DPLL	48MHz	Usb clock

Analog Clock Block Diagram

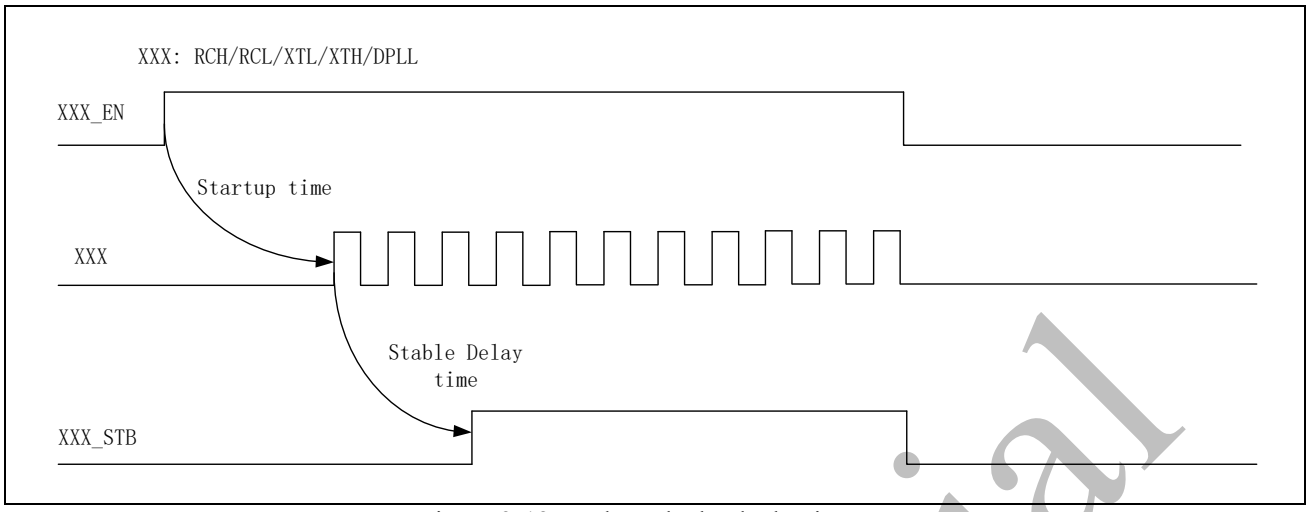


Figure 3-12 Analog Clock Block Diagram

EN_XXX represents the enable signal of the corresponding clock. If the enable signal is pulled high, the clock is output to the digital circuit after a clock setup time. The setup time is as follows: XTL 200ms, RCH 5us, XTH 200us.

The STB Delay time is controlled by two bits to control the time of XXX_STB is pulled high. More information please refer to the register definition.

After each clock reaches the digital part, it is used inside the digital after passing through the XXX_STB Gate. the Gate is added on the digital side.

3.3.3.2 System Clock Block Diagram

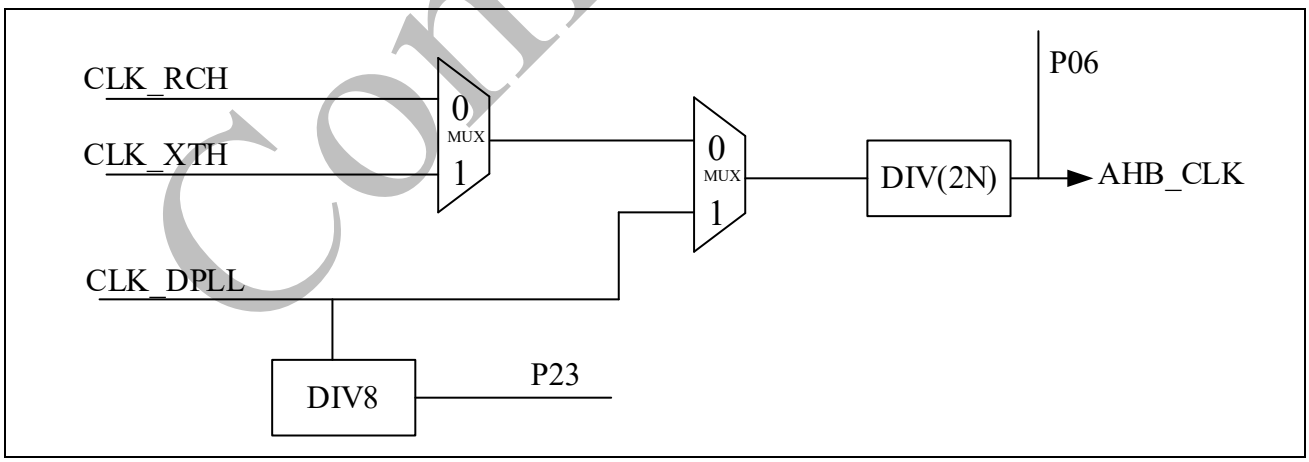


Figure 3-13 Digital Clock Block Diagram

Compared with Figure 3-11, the difference are shown as follow:

- XTL and RCL are deleted as the system clock
- Frequency division after DPLL is deleted
- System frequency division is added after selection

- Output of clock for test: 32K: P11; RCH: P17; XTH: P27; AHB_CLK: P06; DPLL(div8): P23
- Clock pin: XTL: P20,P21; RCL external input: P17; RCH external input: P16

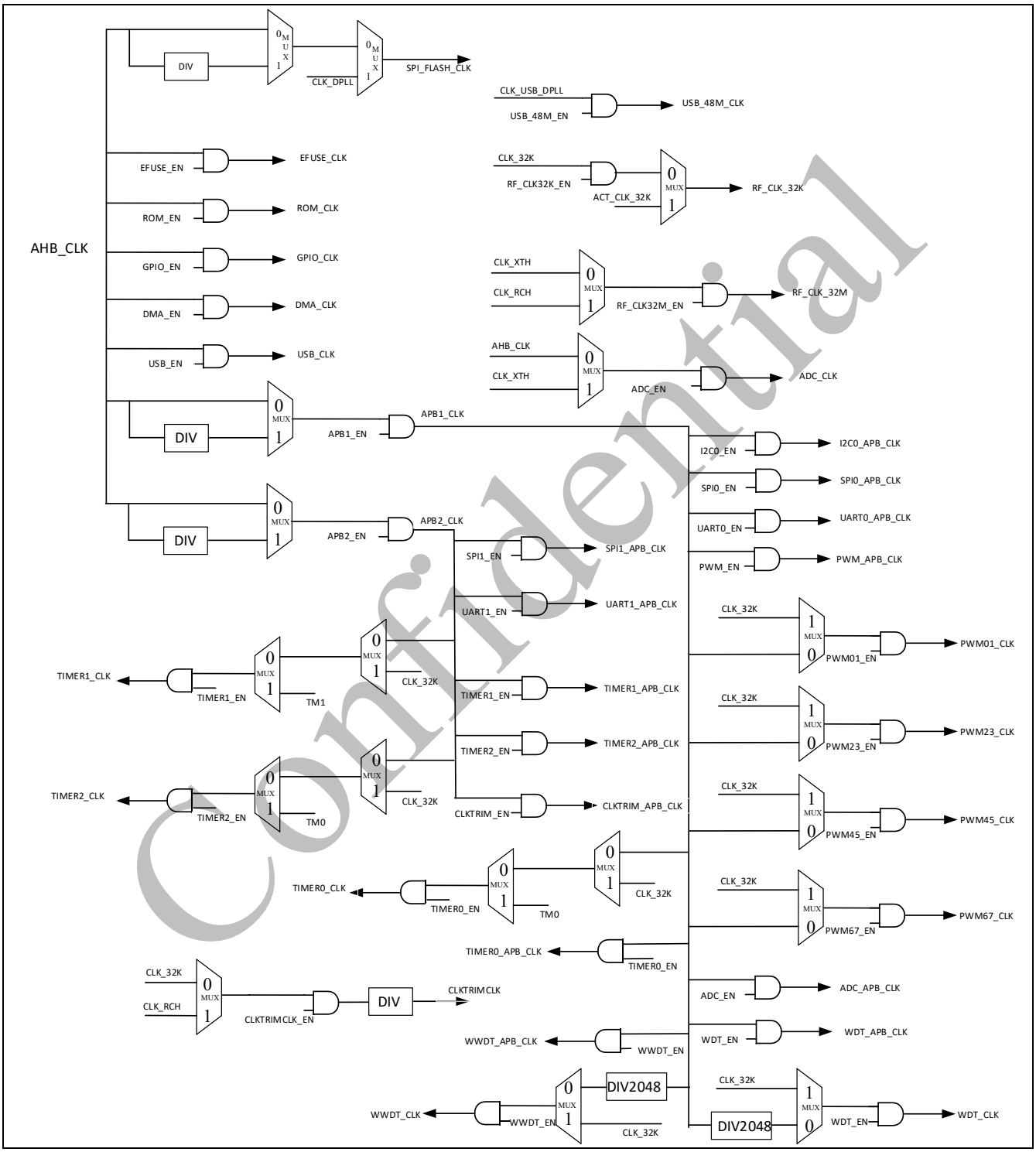


Figure 3-14 System Periphral Clock

3.3.4 RCC Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
RCC Base Address: RCC_BA = 0x4004_0000				
RSTSTS	RCC_BA+0x00	R/W	System Reset Status Register	0x0000_0040
IPRST0	RCC_BA+0x04	R/W	Peripheral Reset Control Register 0	0x0000_0000
IPRST1	RCC_BA+0x08	R/W	Peripheral Reset Control Register 1	0x0000_0000
BODCTL	RCC_BA+0x0C	R/W	Brown-out Detector Control Register	0x000C_0040
BLDBCTL	RCC_BA+0x10	R/W	BOD LVR De-Bounce Control Register	0x0000_0101
CLK_TOP_CTRL	RCC_BA+0x14	R/W	System Clock Global Control Register	0x0000_0000
RCL_CTRL	RCC_BA+0x18	R/W	RCL Control Register	0x0104_8081
RCH_CTRL	RCC_BA+0x1C	R/W	RCH Control Register	0x0103_5021
XTL_CTRL	RCC_BA+0x20	R/W	XTL Control Register	0x0003_1020
XTH_CTRL	RCC_BA+0x24	R/W	XTH Control Register	0x0000_0020
DPLL_CTRL	RCC_BA+0x28	R/W	DPLL Control Register	0x0000_1480
AHB_CLK_CTRL	RCC_BA+0x2C	R/W	AHB Periphral Clock Control Register	0x0001_0C264
APB1_CLK_CTRL0	RCC_BA+0x30	R/W	APB1 Periphral Clock Control Register	0x0000_0000
APB2_CLK_CTRL0	RCC_BA+0x34	R/W	APB2 Periphral Clock Control Register	0x0000_0000
CLKTRIM_CLK_CTRL	RCC_BA+0x38	R/W	CLKTRIM Peripherals Clock Control	0x00F4_0000

3.3.5 RCC Register Description

3.3.5.1 RSTSTS

Register	Offset	R/W	Description	Reset Value
RSTSTS	RCC_BA+0x00	R/W	System Reset Status Register	0x0000_0040

Bits	Description	
[31:8]	Reserved	Reserved.
[7]	CPURF	CPU Reset Flag The CPU reset flag is set by hardware if software writes CPURST (IPRST0[1]) 1 to reset MCU and Flash Memory Controller (FMC). 0 = No reset from CPU. 1 = The MCU and FMC are reset by software setting CPURST to 1. Note: Software can write 1 to clear this bit to zero.
[6]	PORRF	POR Reset Flag The POR reset flag is set by the “Reset Signal” from the Power-on Reset (POR) Controller to indicate the previous reset source. 0 = No reset from POR. 1 = POR had issued the reset signal to reset the system. Note: Software can write 1 to clear this bit to zero.
[5]	SYSRF	System Reset Flag The system reset flag is set by the “Reset Signal” from the MCU to indicate the previous reset source. 0 = No reset from MCU 1 = The MCU had issued the reset signal to reset the system by writing 1 to the bit RCCRESETREQ (SCS_AIRCR[2]), Application Interrupt and Reset Control Register, address = 0xE000ED0C) in system control registers of MCU. Note: Software can write 1 to clear this bit to zero.
[4]	BODRF	BOD Reset Flag The BOD reset flag is set by the “Reset Signal” from the Brown-out Detector to indicate the previous reset source. 0 = No reset from BOD. 1 = The BOD had issued the reset signal to reset the system. Note: Software can write 1 to clear this bit to zero.
[3]	LVRRF	LVR Reset Flag The LVR reset flag is set by the “Reset Signal” from the Low-Voltage-Reset (LVR) to indicate the previous reset source. 0 = No reset from LVR. 1 = LVR had issued the reset signal to reset the system. Note: Software can write 1 to clear this bit to zero.
[2]	WDTRF	WDT Reset Flag The WDT reset flag is set by the “Reset Signal” from the Watchdog Timer or Window Watchdog Timer to indicate the previous reset source. 0 = No reset from watchdog timer or window watchdog timer. 1 = The watchdog timer or window watchdog timer had issued the reset signal to reset the system.

		Note: Software can write 1 to clear this bit to zero.
[1]	PINRF	NRESET Pin Reset Flag The nRESET pin reset flag is set by the “Reset Signal” from the nRESET pin Controller to indicate the previous reset source. 0 = No reset from nRESET pin. 1 = Pin nRESET had issued the reset signal to reset the system. Note: Software can write 1 to clear this bit to zero.
[0]	CHIPRF	CHIP Reset Flag The CHIP reset flag is set by the “Reset Signal” from the CHIPRST (IPRST0[0]) to indicate the previous reset source. 0 = CHIPRST. 1 = CHIPRST had issued the reset signal to reset the system. Note: Software can write 1 to clear this bit to zero.

3.3.5.2 IPRST0

Register	Offset	R/W	Description	Reset Value
IPRST0	RCC_BA+0x04	R/W	Peripheral Reset Control Register 0	0x0000_0000

Bits	Description	
[31:11]	Reserved	Reserved.
[10]	FTORRST	FTOR Reset (Write Protect) FTOR reset is used to reset system and switch from flash mode to rom mode. if setting this bit, reset range: completely consistent with RCCRESETREQ (sys reset), and this bit will automatically return to 0. 0 = FTOR normal operation. 1 = FTOR reset.
[9]	CHIPSCOPEEN	CHIP Reset scope selection The CHIPSCOPEEN is the enable signal for expaned the reset scope of CHIP Reset. 0 = CHIP reset can't clear ROMFLAG to 0 1 = CHIP reset can clear ROMFLAG to 0
[8]	MDMRST	MDM Reset 0 = MDM normal operation. 1 = MDM reset.
[7]	MDMSTDBYRST	MDM STDBY Reset 0 = MDM STDBY normal operation. 1 = MDM STDBY reset.
[6]	USBRST	USB Reset 0 = USB normal operation. 1 = USB reset.
[5]	Reserved	Reserved.
[4]	EFUSERST	EFUSE Reset 0 = EFUSE normal operation. 1 = EFUSE reset.
[3]	LLRST	BLE LL Reset 0 = BLE LL normal operation. 1 = BLE LL reset.
[2]	DMARST	DMA Reset 0 = DMA normal operation. 1 = DMA reset.
[1]	CPURST	Processor Core One-shot Reset (Write Protect) Setting this bit will only reset the processor core and Flash Memory Controller (FMC), and this bit will automatically return to 0. 0 = Processor core normal operation. 1 = Processor core one-shot reset. Note: This bit is write protected. Refer to the RCC_REGLCTL register. Note: Must set AHB_CLK_CTRL[20](spi_flash_clk_sel) reg to 0 before set cpurst to 1

[0]	CHIPRST	CHIP One-shot Reset (Write Protect) Setting this bit will reset the whole chip, including Processor core and all peripherals, and this bit will automatically return to 0. The CHIPRST is the same as the POR reset, all the chip controllers is reset and the chip settings from flash are also reload. 0 = Chip normal operation. 1 = CHIP one-shot reset. Note: This bit is write protected. Refer to the RCC_REGLCTL register.

3.3.5.3 IPRST1

Register	Offset	R/W	Description	Reset Value
IPRST1	RCC_BA+0x08	R/W	Peripheral Reset Control Register 1	0x0000_0000

Bits	Description	
[31:17]	Reserved	Reserved.
[16]	CLKTRIMRST	CLKTRIM Controller Reset 0 = CLKTRIM controller normal operation. 1 = CLKTRIM controller reset.
[15]	GPIORST	GPIO Controller Reset 0 = GPIO controller normal operation. 1 = GPIO controller reset.
[14]	TMR2RST	Timer2 Controller Reset 0 = Timer2 controller normal operation. 1 = Timer2 controller reset.
[13]	TMR1RST	Timer1 Controller Reset 0 = Timer1 controller normal operation. 1 = Timer1 controller reset.
[12]	TMR0RST	Timer0 Controller Reset 0 = Timer0 controller normal operation. 1 = Timer0 controller reset.
[11]	WWDTRST	WWDT Controller Reset 0 = WWDT controller normal operation. 1 = WWDT controller reset.
[10]	WDTRST	WDT Controller Reset 0 = WDT controller normal operation. 1 = WDT controller reset.
[9]	ADCRST	ADC Controller Reset 0 = ADC controller normal operation. 1 = ADC controller reset.
[8]	PWM0RST	PWM Controller Reset 0 = PWM controller normal operation. 1 = PWM controller reset.
[7]	UART1RST	UART1 Controller Reset 0 = UART1 controller normal operation. 1 = UART1 controller reset.
[6]	UART0RST	UART0 Controller Reset 0 = UART0 controller normal operation. 1 = UART0 controller reset.
[5:4]	Reserved	Reserved
[3]	SPI1RST	SPI1 Controller Reset 0 = SPI1 controller normal operation. 1 = SPI1 controller reset.
[2]	SPI0RST	SPI0 Controller Reset 0 = SPI0 controller normal operation. 1 = SPI0 controller reset.

[1]	Reserved	Reserved
[0]	I2C0RST	I2C0 Controller Reset 0 = I2C0 controller normal operation. 1 = I2C0 controller reset.

3.3.5.4 BODCTL

Register	Offset	R/W	Description	Reset Value
BODCTL	RCC_BA+0x0C	R/W	Brown-out Detector Control Register	0x000C_0040

Bits	Description	
[31:23]	Reserved	Reserved.
[22]	BOD_TST_AON	BOD analog test enable control (should enable high voltage sync) 0 = BOD test disable 1 = LVR test enable
[21]	LVR_TST_AON	LVR analog test enable control (should enable high voltage sync) 0 = LVR test disable 1 = LVR test enable
[20:18]	BOD_SEL_AON	BOD voltage threshold select control(should enable high voltage sync) 000: 1.75 001: 1.95 010: 2.15 011: 2.35 100: 2.55 101: 2.75 110: 2.95 111: NC
[17]	EN_BOD_AON	BOD enable control (should enable high voltage sync) 0 = BOD is disable 1 = BOD is enable
[16]	EN_LVR_AON	LVR enable control (should enable high voltage sync) 0 = LVR is disable 1 = LVR is enable. The typical reset voltage threshold is 1.7V
[15:7]	Reserved	Reserved.
[6]	BODOUT	Brown-out Detector Output Status 1 = Brown-out Detector status output is 0, the detected voltage is higher than BODVL setting. 0= Brown-out Detector status output is 1, the detected voltage is lower than BODVL setting.
[5]	Reserved	Reserved.
[4]	BODIF	Brown-out Detector Interrupt Flag 0 = Brown-out Detector does not detect any voltage draft at VDD down through or up through the voltage of BODVL setting. 1 = When Brown-out Detector detects the VDD is dropped through the voltage of BODVL setting or the VDD is raised up through the voltage of BODVL setting, this bit is set to 1 and the Brown-out interrupt is requested if Brown-out interrupt is enabled.
[3]	BODRSTEN_AON	Brown-out Reset Enable Bit (Write Protect) (should enable high voltage sync) 0 = Brown-out “INTERRUPT” function Enabled; when the Brown-out Detector function is enable and the detected voltage is lower than the threshold, then assert a signal to interrupt the MCU 1 = Brown-out “RESET” function Enabled; when the Brown-out Detector function is enable and the detected voltage is lower than the threshold then assert a signal to reset

		the chip. Note: When the BOD_EN is enabled and the interrupt is asserted, the interrupt will be kept till the BOD_EN is set to 0. The interrupt for CPU can be blocked by disabling the NVIC in CPU for BOD interrupt or disable the interrupt source by disabling the BOD_EN and then re-enabling the BOD_EN function if the BOD function is required.
[2:0]	Reserved	Reserved.

3.3.5.5 BLDBCTL

Register	Offset	R/W	Description	Reset Value
BLDBCTL	RCC_BA+0x10	R/W	BOD LVR De-Bounce Control Register	0x0000_0101

Bits	Description	
[31:14]	Reserved	Reserved.
[13:8]	LVRDBSEL_AON	LVR De-Bounce (glitch) time Control register(should enable high voltage sync) [0]=1: about 2^0 SLOW clock [1]=1: about 2^1 SLOW clock [2]=1: about 2^2 SLOW clock [3]=1: about 2^3 SLOW clock [4]=1: about 2^4 SLOW clock [5]=1: about 2^5 SLOW clock(default) Note: If software enables more than one bit, the bit with the smallest number will be selected and the other enabled channels will be ignored.
[7:6]	Reserved	Reserved.
[5:0]	BODDBSEL_AON	BOD De-Bounce (glitch) time Control register(should enable high voltage sync) [0]=1: about 2^0 SLOW clock [1]=1: about 2^1 SLOW clock [2]=1: about 2^2 SLOW clock [3]=1: about 2^3 SLOW clock [4]=1: about 2^4 SLOW clock [5]=1: about 2^5 SLOW clock(default) Note: If software enables more than one bit, the bit with the smallest number will be selected and the other enabled channels will be ignored.

3.3.5.6 CLK_TOP_CTRL

Register	Offset	R/W	Description	Reset Value
CLK_TOP_CTRL	RCC_BA+0x14	R/W	System Clock Global Control Register	0x0000_0000

Bits	Description		
[31:24]	Reserved		Reserved
[23:20]	APB2_DIV	RW	APB2 clock divider control [3:0]=0: clock does not divide [3:0]=N: clock is divided, clock = 1/(2*N), (0<N<16) Note: should first set these divider bits if need; then enable AHB_CLK_CTRL[4].
[19:16]	APB1_DIV	RW	APB1 clock divider control [3:0]=0: clock does not divide [3:0]=N: clock is divided, clock = 1/(2*N), (0<N<16) Note: should first set these divider bits if need; then enable AHB_CLK_CTRL[3].
[15:12]	AHB_DIV	RW	AHB clock divider control [3:0]=0: clock does not divide [3:0]=N: clock is divided, clock = 1/(N+1), (0<N<16)
[11]	Reserved		Reserved
[10]	CLK32K_SEL_AON	RW	0：RCL；1：XTL(should enable high voltage sync)
[9:8]	SYS_CLK_SEL	RW	System Clock source select 00：RCH (default) 01：XTH 10/11：DPLL
[7:1]	Reserved		Reserved
[1]	Rsicv_sys_cnt_en	RW	Rsicv system counter enable control 0: disable 1:enable

3.3.5.7 RCL_CTRL

Register	Offset	R/W	Description	Reset Value
RCL_CTRL	RCC_BA+0x18	R/W	RCL Control Register	0x0104_8081

Bits	Description		
[31:25]	Reserved		Reserved
[24]	RCL_Stable	RO	RCO_LOSC clock stable flag
[23:20]	Reserved	RW	Reserved
[19:18]	RCL_TEMPTRIM_AON	RW	The temperature tune of RCL (should enable high voltage sync)
[17:16]	RCL_DELAY_AON	RW	RCO_LOSC startup counter register (should enable high voltage sync)
[15:8]	RCL_FREQ_FINE_AON	RW	Fine calibration trim of RCL (should enable high voltage sync) Default: 8'b10000000
[7:4]	RCL_FREQ_COARSE_AON	RW	Coarse calibration trim of RCL (should enable high voltage sync) Default: 4'b1000
[3]	Reserved	RW	Reserved
[2]	RCL_CLK_SEL_AON	RW	Register for RCO_LOSC clock selection (should enable high voltage sync)
[1]	RCL_TM_AON	RW	RCO_LOSC test enable (should enable high voltage sync)
[0]	RCL_EN_AON	RW	RCL enable control 0: disable 1: enable (should enable high voltage sync)

3.3.5.8 RCH_CTRL

Register	Offset	R/W	Description	Reset Value
RCH_CTRL	RCC_BA+0x1C	R/W	RCH Control Register	0x0103_5021

Bits	Description		
[31:25]	Reserved		Reserved
[24]	RCH_Stable	RO	RCO_HOSC clock stable flag
[23:18]	Reserved		Reserved
[17:16]	RCH_DELAY	RW	RCO_HOSC startup counter register
[15:8]	RCH_FREQ	RW	RCH_HOSC frequency register, 0.4% per step
[7:6]	Reserved		Reserved
[5:4]	RCH_BIAS	RW	Coarse frequency register 5.2MHz perstep
[3]	Reserved		Reserved
[2]	RCH_CLK_SEL	RW	Register for RCO_HOSC clock selection
[1]	RCH_TM	RW	RCO_HOSC test enable
[0]	RCH_EN	RW	RCO_HOSC enable control 0: disable 1: enable (should enable high voltage sync)

3.3.5.9 XTL_CTRL

Register	Offset	R/W	Description	Reset Value
XTL_CTRL	RCC_BA+0x20	R/W	XTL Control Register	0x0003_1020

Bits	Description		
[31:25]	Reserved		Reserved
[24]	XTL_Stable	RO	XO_LOSC clock stable flag
[23:18]	Reserved		Reserved
[17:16]	XTL_DELAY_AON	RW	XO_LOSC startup counter register (should enable high voltage sync)
[15:14]	Reserved	RW	Reserved
[13]	XTL_CAP_EN_AON	RW	XTL inside cap enable 1: enable 0: disable (should enable high voltage sync)
[12:7]	XTL_TUNE_AON	RW	XO_LOSC freq trim (should enable high voltage sync)
[6:4]	XTL_CORE_AON	RW	XO_LOSC core bias register (should enable high voltage sync)
[3:2]	Reserved	RW	Reserved
[1]	XTL_TM_AON	RW	XO_LOSC test enable (should enable high voltage sync)
[0]	XTL_EN_AON	RW	XTL enable control 0: disable 1: enable (should enable high voltage sync)

3.3.5.10 XTH_CTRL

Register	Offset	R/W	Description	Reset Value
XTH_CTRL	RCC_BA+0x24	R/W	XTH Control Register	0x0000_0020

Bits	Description		
[31:25]	Reserved		Reserved
[24]	XTH_Stable	RO	XO_FSYN clock stable flag
[23:17]	Reserved		Reserved
[16]	XTH_DELAY	RW	XO_FSYN startup counter register
[15:11]	Reserved		Reserved
[10]	XTH_ICORE	RW	The current tune of XTH 0: default 1: large cap mode
[9:4]	XTH_XOCAP_SEL	RW	Xo cap select control
[3]	XTH_DEGLITCH_EN	RW	Xth32m deglitch enable
[2]	XTH_STARTUP_FAST	RW	XO_FSYN fast startup register
[1]	XTH_TEST_EN	RW	XO_FSYN enable
[0]	XTH_EN	RW	XO_FSYN enable control 0: disable 1: enable

3.3.5.11 DPLL_CTRL

Register	Offset	R/W	Description	Reset Value
DPLL_CTRL	RCC_BA+0x28	R/W	DPLL Control Register	0x0000_1480

Bits	Description		
[31:25]	Reserved		Reserved
[24]	DPLL_Stable	RO	DPLL stable flag
[17:16]	DPLL_DELAY	RW	PLL startup counter register
[15:14]	Reserved		Reserved
[13]	DPLL_DIV_SEL	RW	Reserved
[12:11]	DPLL_VCO_FREQ_TRIM	RW	DPLL VCO frequence trim control 00: 65M~193M 01: 99M~220M 10: 129M~246M 11: 158M~271M
[10:8]	DPLL_KVCO_CTRL	RW	DPLL KVCO control 000: 70MHz/V 111: 275MHz/V step: 30MHz/V
[7:6]	DPLL_ICP_CTRL	RW	DPLL charge pump current control 00: 0.5*I 01: I 10: 1.5*I 11: 2*I
[5:4]	DPLL_ICP_BIAS	RW	DPLL charge pump bias current control 00: 5uA 01: 7.5uA 10: 10uA 11: 12.5uA
[3]	DPLL_REF_CLK	RW	PLL reference clock register 0/1: RCO/XO
[2]	DPLL_FREQ	RW	PLL frequency register 0/1: 48MHz/32MHz
[1]	DPLL_TM	RW	Register for pll test mode
[0]	DPLL_EN	RW	DPLL enable control 0: disable 1: enable

3.3.5.12 AHB_CLK_CTRL

Register	Offset	R/W	Description	Reset Value
AHB_CLK_CTRL	RCC_BA+0x2C	R/W	AHB Periphral Clock Control Register	0x0001_0C26

Bits	Description		
[31:22]	Reserved		reserved
[21]	DPLL_CLK_TST_EN	RW	DPLL clk test enable 0: disable 1: enable (DPLL/8)
[20]	SPI_FLASH_CLK_SEL	RW	Spi flash clock source select control 0: ahb clk or ahb_clk div by Spi_flash_clk_div[3:0] 1: dpll clock
[19:16]	SPI_FLASH_CLK_DIV	RW	Spi flash clock source divide [3:0] = 0: clock does not divide [3:0] = N: clock is divided, clock = 1/(2N), (0<N<16)
[15]	Reserved	RW	reserved
[14]	USB_48M_CLK_EN	RW	USB 48Mhz clock enable control 0: disable 1: enable
[13]	USB_AHB_CLK_EN	RW	USB AHB clock enable control 0: disable 1: enable
[12]	Reserved		Reserved
[11]	EFUSE_CLK_EN	RW	EFUSE clock enable control. 0: disable 1: enable Default 1
[10]	ROM_CLK_EN	RW	ROM clock enable control. 0: disable 1: enable Defaule 1
[9]	Reserved	RW	reserved
[8]	BLE_CLK32M_SEL	RW	BlE 32MHz clock select 0 = 32M XTAL 1 = 32M RC Note: should first select the clk source if need; then enable BLE_CLK32K_EN.
[7]	BLE_CLK32K_EN	RW	BLE 32KHz clock enable control 0: disable 1: enable
[6]	BLE_CLK32M_EN	RW	BLE 32MHz clock enable control 0: disable 1: enable
[5]	RCC_AHB_CLK_EN	RW	RCC ahb_clk enable control. Default 1 0: disable 1: enable

[4]	APB2_CLK_EN	RW	APB2 clock enable control 0: disable 1: enable
[3]	APB1_CLK_EN	RW	APB1 clock enable control 0: disable 1: enable
[2]	ST_CLK_EN（M0）	RW	System Tick clock enable 0: disable 1: enable
[1]	GPIO_CLK_EN	RW	GPIO clock enable control 0: disable 1: enable
[0]	DMA_CLK_EN	RW	DAM clock enable control 0: disable 1: enable

3.3.5.13 APB1_CLK_CTRL0

Register	Offset	R/W	Description	Reset Value
APB1_CLK_CTRL0	RCC_BA+0x30	R/W	APB1 Periphral Clock Control Register	0x0000_0000

Bits	Description		
[31:25]	Reserved		Reserved
[24]	Adc_clk_sel	RW	Adc sample clock select 0: ahb clk 1: XTH Note: should first select the clk source if need; then enable Adccken.
[23]	Pwm0_67cksel	RW	Pwm67 clock select 0: apb 1: RCL/XTL Note: should first select the clk source if need; then enable Pwm0_67cken.
[22]	Pwm0_45cksel	RW	Pwm45 clock select 0: apb 1: RCL/XTL Note: should first select the clk source if need; then enable Pwm0_45cken.
[21]	Pwm0_23cksel	RW	Pwm23 clock select 0: apb 1: RCL/XTL Note: should first select the clk source if need; then enable Pwm0_23cken.
[20]	Pwm0_01cksel	RW	Pwm01 clock select 0: apb 1: RCL/XTL Note: should first select the clk source if need; then enable Pwm0_01cken.
[19:18]	tmr0sel	RW	00: APB1_CLK 01: RCL/XTL 10, 11: TM0_I, Note: should first select the clk source if need; then enable Tmr0cken.
[17]	wwdtsel	RW	0: milli_clk 1: RCL/XTL Note: should first select the clk source if need; then enable Wwdtcken.
[16]	wdtsel	RW	0: RCL/XTL 1: milli_clk Note: should first select the clk source if need; then enable Wdtcken.
[15:14]	Reserved		Reserved
[13]	milli_clk_en	RW	1: enable 0: disable
[12]	Tmr0cken	RW	Timer0 clock enable 0: clock disable 1: clock enable
[11]	Wwdtcken	RW	Wwdt clock enable 0: clock disable 1: clock enable
[10]	Wdtcken	RW	Wdt clock enable 0: clock disable 1: clock enable
[9]	Adccken	RW	Adc clock enable 0: clock disable 1: clock enable
[8]	Pwm0cken	RW	Pwm apb clock enable 0: clock disable 1: clock enable
[7]	Pwm0_67cken	RW	Pwm67 clock enable

			0: clock disable 1: clock enable
[6]	Pwm0_45cken	RW	Pwm45 clock enable 0: clock disable 1: clock enable
[5]	Pwm0_23cken	RW	Pwm23 clock enable 0: clock disable 1: clock enable
[4]	Pwm0_01cken	RW	Pwm01 clock enable 0: clock disable 1: clock enable
[3]	Uart0cken	RW	Uart0 clock enable 0: clock disable 1: clock enable
[2]	Reserved		Reserved
[1]	Spi0cken	RW	Spi0 clock enable 0: clock disable 1: clock enable
[0]	I2c0cken	RW	I2c0 clock enable 0: clock disable 1: clock enable

3.3.5.14 APB2_CLK_CTRL0

Register	Offset	R/W	Description	Reset Value
APB2_CLK_CTRL0	RCC_BA+0x34	R/W	APB2 Periphral Clock Control Register	0x0000_0000

Bits	Description		
[31:12]	Reserved		Reserved
[11:10]	tmr2sel	RW	00: APB2_CLK 01: RCL/XTL 10, 11: TM2_I Note: should first select the clk source if need; then enable Tmr2cken.
[9:8]	tmr1sel	RW	00: APB2_CLK 01: RCL/XTL 10, 11: TM1_I Note: should first select the clk source if need; then enable Tmr1cken.
[7:6]	Reserved		Reserved
[5]	tmr2cken	RW	Timer2 clock enable 0: clock disable 1: clock enable
[4]	Tmr1cken	RW	Timer1 clock enable 0：clock disable 1：clock enable
[3]	Uart1cken	RW	Uart1 clock enable 0: clock disable 1: clock enable
[2]	Reserved		Reserved
[1]	Spi1cken	RW	Spi1 clock enable 0: clock disable 1: clock enable
[0]	Reserved		Reserved

3.3.5.15 CLKTRIM_CLK_CTRL

Register	Offset	R/W	Description	Reset Value
CLKTRIM_CLK_CTRL	RCC_BA+0x38	R/W	CLKTRIM Peripherals Clock Control	0x00F4_0000

Bits	Description		
[31:24]	Reserved		Reserved
[23:15]	xth_div	RW	Xth divider for xtl quick setup，default value：0x1e8
[14]	Xtl_quick_en	RW	Xtl quick setup enable control 0: disable 1: enable
[13]	Reserved	RW	Reserved
[12:4]	clktrimclk_div	RW	clktrim clock divide [8:0]=0: clock does not divide [8:0]=N: clock is divided, clock = 1/(2*N), (0<N<512)
[3]	Reserved	RW	Reserved
[2]	clktrimclk_sel	RW	clktrim clock select 0: 32K clk 1: 32M RC clk
[1]	clktrimclk_en	RW	clktrim clock enable 0: clock disable 1: clock enable
[0]	Reserved	RW	Reserved

3.4 Flash Memory Controller (FMC)

3.4.1 Overview

In PAN101x series, the flash is used to store the application code. The SPI flash generally use NOR FLASH in consideration of the speed, whose interface supports SPI communication with 1/2/4 line mode. The speed of the SPI interface can be influenced by SOC, SPI NOR FLASH, package and PCB alignment. Apparently, the higher the speed is, the higher the efficiency of program executes.

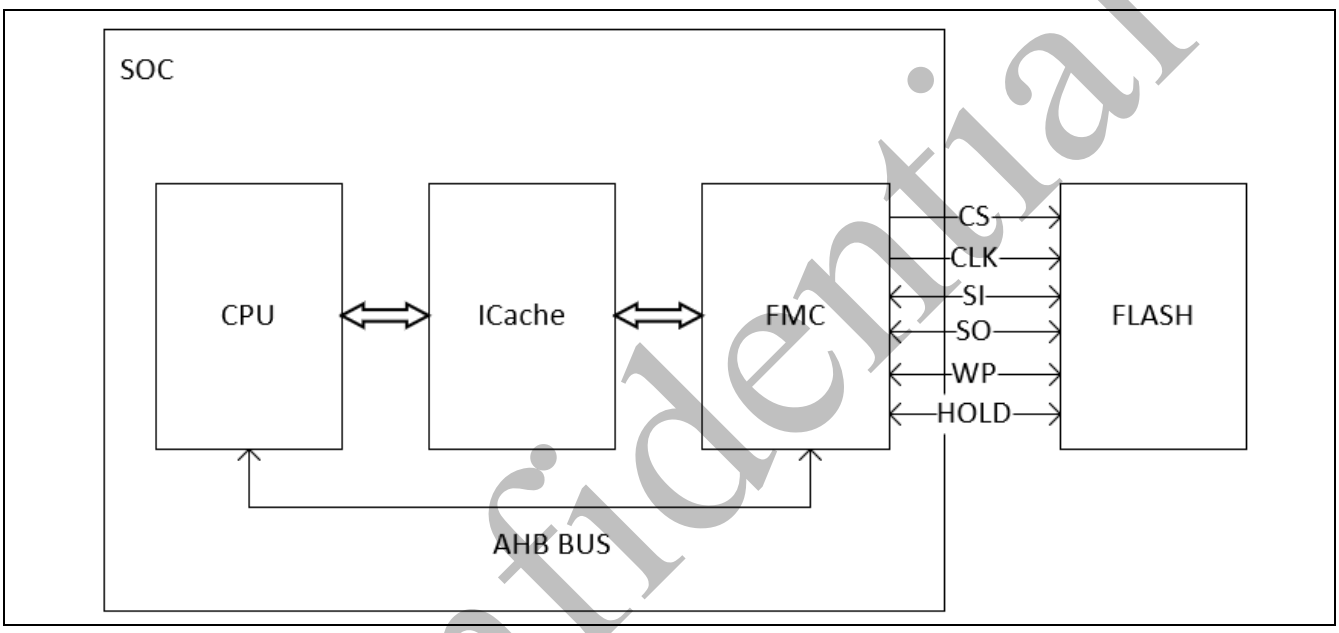


Figure 3-15 SPI Flash Controller Diagram

CPU access to flash through ICache is defined as cpu_access; CPU direct access to flash is defined as fw_access. The cpu_access path only has read requests. The fw_access path has a read/write/erase/ request.

3.4.2 Features

- SPI interface supporting 1/2/4 line mode
- Support IAP function
- The fw_access channel supports all read commands. The cpu_access channel supports all read commands except the (03H) command
- Support all erase commands: PE, SE, BE, CE
- The write command only supports PP
- Support flash low power consumption function, hardware can be configured whether to send DP and RDP commands automatically

- Support the sampling edge configuration of the read channel;
- Support remap function;
- Support CRC32 check function.
- Support suspend and resume operations during program and erase.
- Support erase protection for the flash info area.

3.4.3 I-cache

The I-cache applied in the PAN101x series includes the features as follow:

- Cache capacity 4K, 2-way group associative organization structure, block size 32 bytes
- Support wrap round reading and incr burst reading
- Support bypass instruction fetch, cold start
- Support gating
- Support clearing the cache
- Does not support write-back function
- Cache line uses FIFO replacement algorithm

3.4.3.1 I-cache Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
I_cache Base Address: ICA_BA = 0x0880_0000				
x_cache_en	ICA_BA+0x00	R/W	I_cache Control Register	0x0000_0000
x_cache_ini	ICA_BA+0x08	R/W	I_cache Initialize Control Register	0x0000_0004

3.4.3.2 I-cache Register description

I_cache Control Register (x_cache_en)

Register	Offset	R/W	Description	Reset Value
x_cache_en	ICA_BA +0x00	R/W	I_cache Control Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	I_CACHE_EN	I_cache enable bit. Set this bit to enable cache, and clear this bit to disable cache. 0 = Disable I_cache. 1 = Enable I_cache.

I_cache Initialize Control Register (x_cache_ini)

Register	Offset	R/W	Description	Reset Value
x_cache_ini	ICA_BA +0x08	R/W	I_cache Initialize Control Register	0x0000_0004

Bits	Description	
[31:3]	Reserved	Reserved.
[2]	CLK_GATED_EN	The default value is 1, which means the clock of i-cache is always on. The software can configure this bit to to reduce i_cache power consumption. The clock is gated when i-cache is not busy and will be on when needed.
[1]	FLASH_HAS_NO_WRAP	If the flash does not support wrap-round read and write, write 1 to this bit, and I_cache reads 32 bytes from the 32-byte boundary address. For example, the flash returns data in the following address sequence: 0x00-0x04-0x08-0x0c-0x10-0x14-0x18- 0x1c. If the flash supportswrap-round read and write, this bit is 0, and I_cache reads 32 bytes from the flash loop each time. For example,the flash can return data in the following address sequence: 0x10-0x14-0x18-0x1c-0x00-0x04-0x08-0x0c. **Note:** When modifying this register, clk_gated_en must be 1.
[0]	INI_TRG	Write 1 to clear cache and this bit will be cleared to 0 by hardware automatically when I_cache operation has finished. 0 = I_cache operation has finished. 1 = I_cache is progressed.

3.4.4 IAP

PAN101x series supports vector table mapping.

The PAN101x series provide In-Application-Programming (IAP) function for user to switch the code execution. User enable the IAP function by booting chip and setting the X_FL_REMAP_ADDR[31:0].

Once chip boots with IAP function is enabled, the vector table will remap to a predetermined address.

3.4.4.1 Remap Function

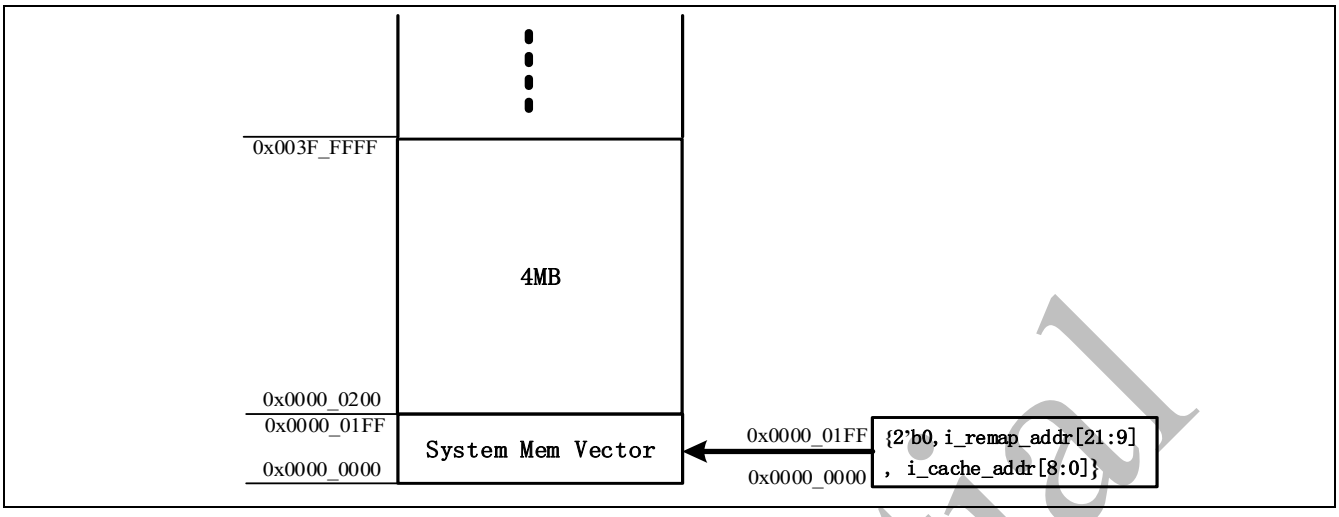


Figure 3-16 Remap Diagram

Remap process:

1. Set remap_adr[31:0] to store the remap address.
2. In the cpu_ctl sub-module, according to the cache request, the remap address is sent under the condition of i_remap_addr[21:9]=13'd0 and related fw. (Need to send the remap_addr of the fw_ctl module to cpu_ctl)

After the system is interrupted, the program is mapped to 0x0000-0x01FF. At this time, the absolute start address of FLASH is mapped to {2'b0, i_remap_addr[21:9], i_cache_addr[8:0]}, where i_remap_addr[21: 9] is the Page of the mapped address, i_cache_addr[8:0] is the Byte of the mapped address, so as to jump to the desired program to continue compiling.

3.4.5 Suspend erase/program to read flash

When flash is in a program/erase process, if an IRQ suspend interrupt is detected, it supports pausing the erase/write process, performing a flash read, and then resuming the erase/write process.In addition to this, 32-bit interrupt enable is configurable.

The Suspend instruction can pause a Page Program, Sector Erase, or Block Erase operation to allow access to the memory array. After the programming or erasing operation has entered the suspended state, the memory array can be read except for the page being programmed or the sector or block being erased.

Suspended Operation	Readable Region of Memory Array
Page Program	All but the Page being programmed
Page Erase	All but the Page being erased
Sector Erase(4KB)	All but the 4KB Sector being erased

Block Erase(32KB)	All but the 32KB Block being erased
Block Erase(64KB)	All but the 64KB Block being erased

When the Serial NOR Flash receives the Suspend instruction, there is a latency of tPSL or tESL before the Write Enable Latch (WEL) bit clears to “0” and the SUS2 or SUS1 sets to “1”, after which the device is ready to accept one of the commands listed in "Table Acceptable Commands During Program/Erase Suspend after tPSL/tESL" (e.g. FAST READ). Refer to " AC Characteristics" for tPSL and tESL timings.

Status Register bit 15 (SUS2) and bit 10 (SUS1) can be read to check the suspend status. The SUS2 (Program Suspend Bit) sets to “1” when a program operation is suspended. The SUS1 (Erase Suspend Bit) sets to “1” when an erase operation is suspended. The SUS2 or SUS1 clears to “0” when the program or erase operation is resumed

3.4.6 SPI Flash Register Map

Register	Offset	R/W	Description	Reset Value
FMC BASE address: 0x4005_0000 **BASE=0x4005_0000**				
X_FL_RDATA	BASE+0x00	RW	Read and write byte count configuration.	32’h0000_0000
X_FL_CTL	BASE+0x04	RW	FMC control information, startup, and other configurations	32’h0000_0400
X_FL_WDATA3	BASE+0x08	RW	FMC write data configuration.	32’h0000_0000
X_FL_X_MODE	BASE+0x0c	RW	Control information configuration (suspend, resume, x_mode, etc.)	32’h5010_1001
X_FL_X2_CMD	BASE+0x10	RW	Working mode CMD configuration (2-wire, four-wire, DP, RDP, etc.	32’habb9_eb3b
X_FL_REMAP_ADDR	BASE+0x14	RW	Flash address remapping	32’h0000_0000
X_FL_DP_CTL	BASE+0x18	RW	Deepsleep enable control, configuration of waiting dp_cnt, rdp_cnt.	32’h0120_0701
X_FL_IRQ_CTL	BASE+0x1c	RW	Optional 32-bit interrupt enable configuration for FMC during suspend	32’h0000_0000
X_FL_SUS_RESU_CMD	BASE+0x20	RW	FMC command configuration for different suspend and resume operations	32’h757a_b030
FMC BASE1 address: 0x4005_0400 **BASE1=0x4005_0400** RAM cache for reading and writing				
Buffer data1	BASE1+0x00	RW	Data to be programmed to one page	32’h0000_0000
Buffer data2	BASE1+0x04	RW	Data to be programmed to one page	32’h0000_0000
……	……	……		
Buffer data n	BASE1+0xfc	RW	Data to be programmed to one page	32’h0000_0000

FMC BASE2 address: 0x4005_0500
BASE2=0x4005_0500 used fo Deccrp ram bist address

Deccrp ram bist address	BASE2+0x00	RW	Deccrp ram bist address	
Deccrp ram bist address	BASE2+0x04	RW	Deccrp ram bist address	
……	……	……		
Deccrp ram bist address	BASE2+0xff	RW	Deccrp ram bist address	

3.4.7 SPI Flash Register Description

3.4.7.1 Read Data Register (X_FL_RDATA)

Register	Offset	R/W	Description	Reset Value
X_FL_RDATA	BASE+0x00	R	Read Data Register	0x0000_0000

Bits	Descriptions	
[31:8]	BYTES_NUM_R	Read byte number (>=0) The read bytes are placed in the buffer, and the maximum number is up to 256 bytes. If the CRC32 check function is used, the maximum byte is 16M.
[7:3]	Reserved	Reserved
[2:0]	BYTES_NUM_W	Write byte number (>=1), the maximum value is 7. The number of write bytes is determined by the command sequence. The contents of the bytes are respectively placed in the wdata1/2/3/4/5/6 registers. If it is configured as 7, the seventh byte is FF by default.

3.4.7.2 Flash Control (X_FL_CTL)

Register	offset	R/W	Description	Reset value
X_FL_CTL	BASE+0x04	R/W	FMC control information, startup, and other configurations	32'h0000_0400

Bits	Description		
[31:24]	W_DATA2_RAW	R/W	Write Data Num=2
[23:16]	W_DATA1_RAW	R/W	Write Data Num=1 （must be CMD）
[15:14]	Reserved		
[13:11]	FLASH_CAPACITY	R/W	In order to improve Flash compatibility, the hardware flash memory size (byte) is notified by software to avoid overflow access to the info area by means of wraparound to protect the Flash info area. 3'b000: The maximum capacity of flash is unknown, and the hardware only implements info protection with 4kbyte address offset. In this case, the hardware itself cannot restrict the wraparound access to the info area. The customers need to use software to bypass the wraparound access to the info area. This situation is used for customers to choose a large flash. 3'b001: The flash capacity is 2M bytes. 3'b010: The flash capacity is 1M byte. 3'b011: The flash capacity is 512K byte. 3'b100: The flash capacity is 256K byte. 3'b101: The flash capacity is 128K byte. By accessing the absolute physical address of the flash, perform remainder analysis on the capacity of the flash. If the remainder belongs to the 0 ~ 4k byte range, it is considered an error access.
[10]	TX_ADDRESS_ TRANSACTION	R/W	Is W_data2/3/4 transmitting an address? 1'b1: If it is an address, then the actual address is {w_data2, w_data3,w_data4}+0x1000 1'b0: If it is not an address, no changes will be made. Only when info_en=0 & tx_address_transaction=1, the FMC hardware adds 1 to w_data3 [4]. If {w_data2[7:0], w_data3[7:0], w_data4[7:0]} represent addresses, then the hardware automatically adds 4k bytes of address offset. For Write status register (WRSR), if you need to send w_data3, modify flash status register[15:8] based on the content of w_data3, then you need to reset tx_address_transaction or set info_en to 1 in time to prevent the hardware from adding 1 to w_data3[4], ensuring that flash status register[12] is the same as w_data3[4]. Note: It is best not to make any changes after configuring it at the beginning.
[9]	INFO_EN	R/W	The first 4kB of the Flash physical address is used as the info area. 1'b1: FMC can read, write, and erase this area. 1'b0: If tx_address_transaction=1, FMC can not erase the area. If forced to erase, the actual area being erased is 4k-8k. If tx_address_transaction=0, then FMC can read, write, and erase the area.

			When info_en==1 || (info_en==0 && tx_address_transaction==0), FMC can read, write, and erase the info 4k area of flash 0x0~0xfff. A flash info protection mechanism is added based on the above to prevent users from accessing the overflow 4K info area using the Wrap Around method of flash. Perform remainder judgment on the new address after correction adding 4k, when info_en protection is enabled and w_data/2/3/4 is the address. When the new 24 bit address is less than 24'h1000, the hardware error flag is valid, the state machine for flash read/write/erase will be in IDLE state, and the software operation trg will be cleared. Then, it should be noted that a normally executed read/write/erase program will automatically reset long_time_op and pp_active to zero. However, due to the overflow protection mechanism, it will prevent the execution of read/write/erase programs, that is, it will not automatically reset long_time_op and pp_active. So users must reset long_time_op to 0, otherwise FMC will crash.
[8]	PP_ACTIVE	R/W	Set to 1 for "Write Page" operation, and the hardware is automatically reset to 0. It should be noted that a normally executed read/write/erase program will automatically reset long_time_op and pp_active to zero. However, due to the overflow protection mechanism, it will prevent the execution of read/write/erase programs, that is, it will not automatically reset long_time_op and pp_active. So users must reset long_time_op to 0, otherwise FMC will crash.
[7:1]	Reserved		
[0]	TRG	R/W	The software writes 1 to start the command and the hardware completes the automatic clearing. Software needs to determine if the bit is 0 before sending the next command. Any operation of the software, such as read, write, erase, suspend, and resume, is dependent on trg=0.

3.4.7.3 Flash write data (X_FL_WDATA3)

Register	offset	R/W	Description	Reset value
X_FL_WDATA3	BASE+0x08	R/W	FMC write data configuration.	32'h0000_0000

Bits	Description		
[31:24]	W_DATA6	R/W	write data num=6
[23:16]	W_DATA5	R/W	write data num=5
[15:8]	W_DATA4_RAW	R/W	write data num=4
[7:0]	W_DATA3_RAW	R/W	write data num=3

3.4.7.4 SPI Flash Mode Select Register (X_FL_X_MODE)

Register	offset	R/W	Description	Reset value
X_FL_X_MODE	BASE+0x0C	R/W	Control information configuration (suspend, resume, x_mode, etc.)	32'h5010_1001

Bits	Description		
[31:26]	TIMEOUT_CNT	R/W	Suspend timeout completion counter. After sending the suspend command, the maximum delay time for completing the suspend is typically 30us (for GD flash, 40us). For safety, it's best to set the suspend timeout to 40us (or GD 50us). The timeout counter does not affect normal suspensions, and it only comes into play when the suspend command is sent at the boundary. In other words, the timeout completion has a lower priority than regular suspend detection For a 40us timeout, 32MHz configuration is set as 010100 (default): 48MHz configuration is set as 011110. 64MHz configuration is set as 101000 For a 50us timeout, 32MHz configuration is set as 011001: 48MHz configuration is set as 100110. 64MHz configuration is set as 110010 due to insufficient bits in a register, the counter will be amplified 2^6 times within the FMC (Flash Memory Controller), which is equivalent to shifting it left by 6 bits TIMEOUT_cnt is used for addressing the issue at the boundary of the suspend command, which arises due to a delay in the FMC's query of flash status information. Therefore, when a flash erase/write operation is at its completion boundary and the FMC sends a suspend command, there is a scenario where the flash operation coincidentally finishes. In this case, the flash will not execute the suspend command as intended, but the FMC itself will continue to execute the suspend command. This situation can lead to erratic behavior in the FMC program. It's important to note that if the suspend operation completes due to a timeout, the FMC internally handles it by directly ending the suspend cycle, eliminating the need to send a resume command to terminate the suspend cycle."
[25]	RESUME_ENABLE	R/W	1'b0: After the suspend instruction takes effect, it does not send a resume instruction (default). 1'b1: After the suspend instruction takes effect, the sending of the resume instruction is enabled, and afterward, hardware automatically clears it. It's noted that the "resume" instruction is considered equivalent to a regular "fw_cmd" and has a similar effect to "trg" but does not depend

			on "bytes_num_r" and "bytes_num_w."
[24]	SUSPEND_SUCCESS	R	1'b0: During an erase/write operation, after sending the suspend instruction, the suspend has not yet taken effect (default). 1'b1: During an erase/write operation, after sending the suspend instruction, the suspend instruction has already taken effect, and you can proceed to send new commands (e.g., read or resume)
[23:21]	Reserved		
[20]	RX_NEG(reserved)	R/W	Control the FMC sampling clock to latch flash return data on which edge: 1'b0: Latch data on the rising edge. 1'b1: Latch data on the falling edge (default).
[19]	CRC32_EN	R/W	CRC32 enable. Hardware will not automatically clear it.
[18]	ENHANCE_MODE	R/W	1'b0: normal mode(default) 1'b1: enhance read mode only 2READ and 4READ command support enhance mode。
[17]	LONG_TIME_OP	R/W	1'b0:(default) 1'b1: Enable this signal before sending WRSR/PP/PE/CE command. Once set to 1, hardware will automatically send the RDSR command to read the WIP status. After detecting WIP as 0, this bit will be automatically cleared. When WIP is 1, the 'trg' signal is also 1
[16]	EN_BURST_WRAP	R/W	1'b0:(default) 1'b1:Enable this signal before sending the SBL (77h) command. Hardware will automatically clear it.
[15:8]	CS_HIGH_CYLE	R/W	The time CS signal needs to be pulled high between commands. The value of cs_high_cycle needs to be greater than the value of clk_dly (default 8'h10)
[7:2]	Reserved		
[1:0]	X_MODE	R/W	2'b00: x1 mode 2'b01: x2 mode(default) 2'b10: x4 mode 2'b11: forbiden Determine the configuration of x_mode based on the number of wires used for data retrieval

3.4.7.5 X_FL_X2_CMD

Register	offset	R/W	Description	Reset value
X_FL_X2_CMD	BASE+0x10	R/W	Working mode CMD configuration (2-wire, four-wire, DP, RDP, etc.	32'hABB9_EB3B

Bits	Description		
[31:24]	RDP_CMD	R/W	Release from deep sleep mode(default 8'hab)
[23:16]	DP_CMD	R/W	Deep sleep mode(default 8'hb9)
[15:8]	X4_CMD	R/W	4-wire mode command(default 8' heb)
[7:0]	X2_CMD	R/W	2-wire mode command(default 8' h3b)

3.4.7.6 X_FL_REMAP_ADDR

Register	Offset	R/W	Description	Reset Value
X_FL_REMAP_ADDR	BASE+0x14	R/W	Flash address remapping	0x0000_0000

Bits	Descriptions	
[31:0]	REMAP_ADDR	Flash address remapping

3.4.7.7 X_FL_DP_CTL

Register	offset	R/W	Description	Reset value
X_FL_DP_CTL	BASE+0x18	R/W	Deepsleep enable control, configuration of waiting dp_cnt, rdp_cnt.	32'h0120_0701

Bits	Description		
[31:16]	RDP_WT_CNT	R/W	The time to wait after sending the release deepsleep(RDP) command, in units of flash clk. It generally needs to be greater than or equal to 8us. For example, at a 32MHz clock, the value of this register needs to be greater than or equal to 0x100; at a 64MHz clock, the value needs to be greater than or equal to 0x200. Considering a margin for practical use, it's recommended to set it as follows: 32MHz clock, set it to 0x120 (default). 48MHz clock, set it to 0x180. 64MHz clock, set it to 0x240.
[15:4]	DP_WT_CNT	R/W	The time to wait after sending the deepsleep(DP) command, in units of flash clk. It generally needs to be greater than or equal to 3us. For example, at a 32MHz clock, the value of this register needs to be greater than or equal to 0x60; at a 64MHz clock, the value needs to be greater than or equal to 0xc0. Considering a margin for practical use, it's recommended to set it as follows: 32MHz clock, set it to 0x070 (default). 48MHz clock, set it to 0x0a8. 64MHz clock, set it to 0x0e0.
[3:1]	Reserved		
[0]	DP_EN	R/W	Enable signal for sending DP/RDP commands (default 1'b1).

3.4.7.8 32-bit interrupt suspend enable(X_FL_IRQ_CTL)

Register	offset	R/W	Description	Reset value
X_FL_IRQ_CTL	BASE+0x1c	R/W	Optional 32-bit interrupt enable configuration for FMC during suspend	32'h0000_0000

Bits	Description		
[31:0]	IER	R/W	32-bit interrupt suspend enable, for software definition (default 32'h0). 1'b0: Interrupt pause disabled. 1'b1: Interrupt pause enabled

3.4.7.9 Suspend/Resume Cmd(X_FL_SUS_RESU_CMD)

Register	offset	R/W	Description	Reset value
X_FL_SUS_RESU_CMD	BASE+0x20	R/W	FMC command configuration for different suspend and resume operations	32'h757a_b030

Bits	Description		
[31:24]	PROG_SUSPEND_CMD	R/W	The suspend command to be sent for pausing flash program (default 8'h75). If it's a Puya flash, configure it as 8'h75. If it's a GD flash, configure it as 8'h75. For specific details, please consult the required IP documentation
[23:16]	PROG_RESUME_CMD	R/W	The resume command to be sent to resume a flash program that has been paused (default 8'h7a). If it's a Puya flash, configure it as 8'h7a. If it's a GD flash, configure it as 8'h7a. For specific details, please consult the required IP documentation
[15:8]	ERASE_SUSPEND_CMD	R/W	The suspend command to be sent for pausing flash erase (default 8'hb0). If it's a Puya flash, configure it as 8'hb0. If it's a GD flash, configure it as 8'h75. For specific details, please consult the required IP documentation
[7:0]	ERASE_RESUME_CMD	R/W	The resume command to be sent to resume a paused flash erase operation (default 8'h75). If it's a Puya flash, configure it as 8'h30. If it's a GD flash, configure it as 8'h7a. For specific details, please consult the required IP documentation.

3.4.7.10 RAM cache for reading and writing

Register	offset	R/W	Description
FMC Base Address FMC_BASE1 = 0x4005_0400			
Buffer data1	BASE1+0x00	R/W	Data to be programmed to one page
Buffer data2	BASE1+0x04	R/W	Data to be programmed to one page
Buffer data3	BASE1+0x08	R/W	Data to be programmed to one page
..	..	..	…
Buffer data64	BASE1+0xfc	R/W	Data to be programmed to one page

3.4.7.11 Deccrp ram bist address

RegisterFile	offset	R/W	Description
FMC Base Address FMC_BASE2 = 0x4005_0500 FMC End Address FMC_BASE2 = 0x4005_05ff Total of 256 bytes, can be accessed arbitrarily, (byte/half-word/word)			

3.5 Firmware Encryption

3.5.1 Overview

In order to prevent a third party from maliciously copying the software code of a mature product on the market, directly omitting the cost of software development and forming an improper advantage, the PAN101x chip has an encryption function.

3.5.2 Features

- Support external program software encryption, hardware internal decryption (encryption algorithm is AES ECB mode).
- The size of the encryption program is 256B, and the interval selection does not support the first 512B program area and the new remap vector table area.
- Support to shield the program injection from FLASH/SRAM program, that is, the program does not run to this block of encryption area, and there is no right to data access.
- Support shielding debug access (for example: take plaintext data operation on encrypted area through SWD method).
- Support system low power consumption process.

3.5.3 Block Diagram

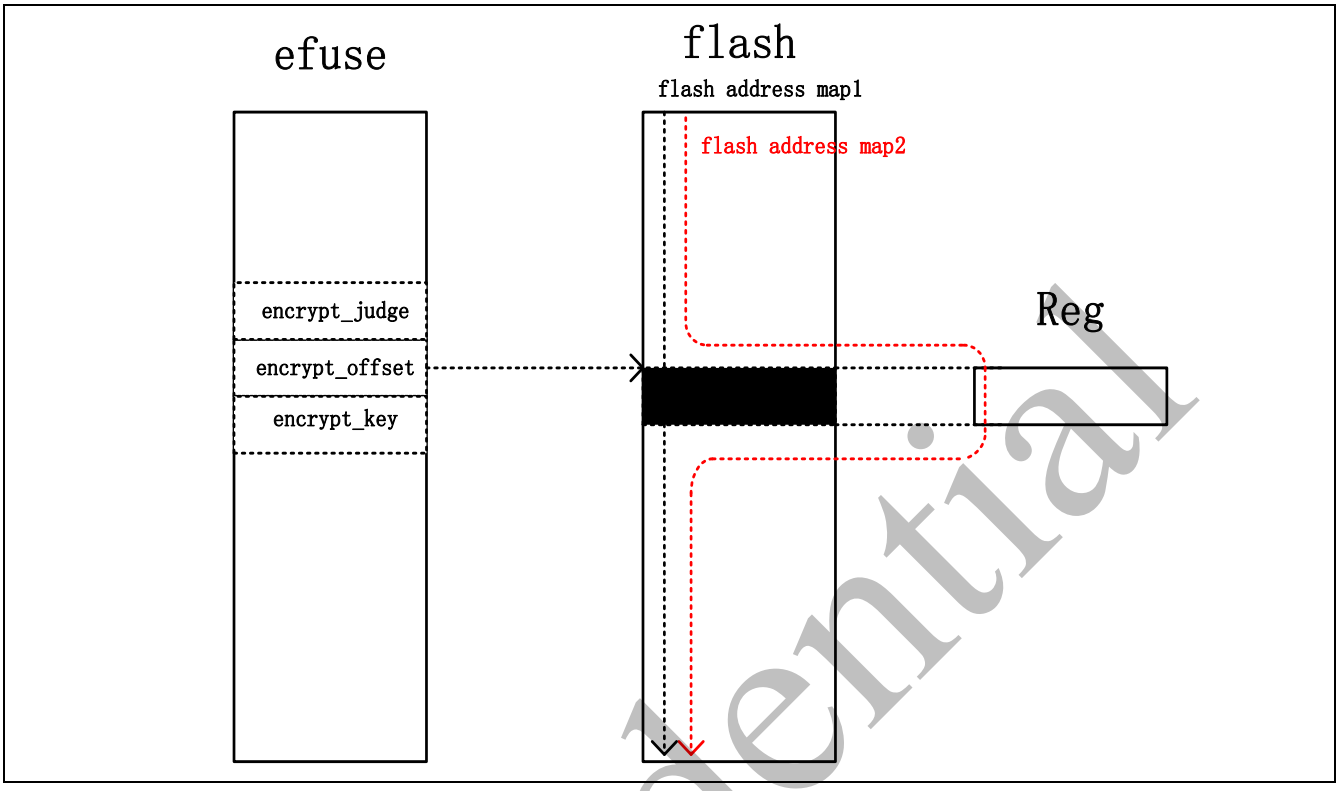


Figure 3-17 Encryption Block Diagram

In order to improve the encryption performance, efuse/AES and other IPs are introduced to cooperate with the encryption and decryption schemes. In Figure 3-17, the encrypted storage structure this time mainly involves three different storage media.

1) efuse can only be programmed once, and the data exists if power down.
2) The flash can be erased and written many times, and the data exists if power down.
3) Regfile supports reading and writing, but the data is lost after power down. This is only used to store the plaintext of the black encryption code in the flash.

There are three types of information related to encryption and decryption currently stored in efuse.

3.5.4 User Operation Process

The host computer should write the encrypt_key/encrypt_offset required for flash burning into the efuse. After writing the efuse, it needs to be read out to verify whether the programming is correct. If it is caused by an operation error or other hardware reasons, the encrypt_judge will not be burned at this time. Encryption and related hardware protection are not required. Flash re-burns the plaintext program to ensure the normal operation of the chip. If the key and offset

are burned correctly, it is required to continue to burn encrypt_judge. In order to prevent the misoperation of batch programming or malicious destruction of efuse, the hardware logic will be protected for these special areas can only be programmed once. The efuse programming process is shown in Figure 3-18.

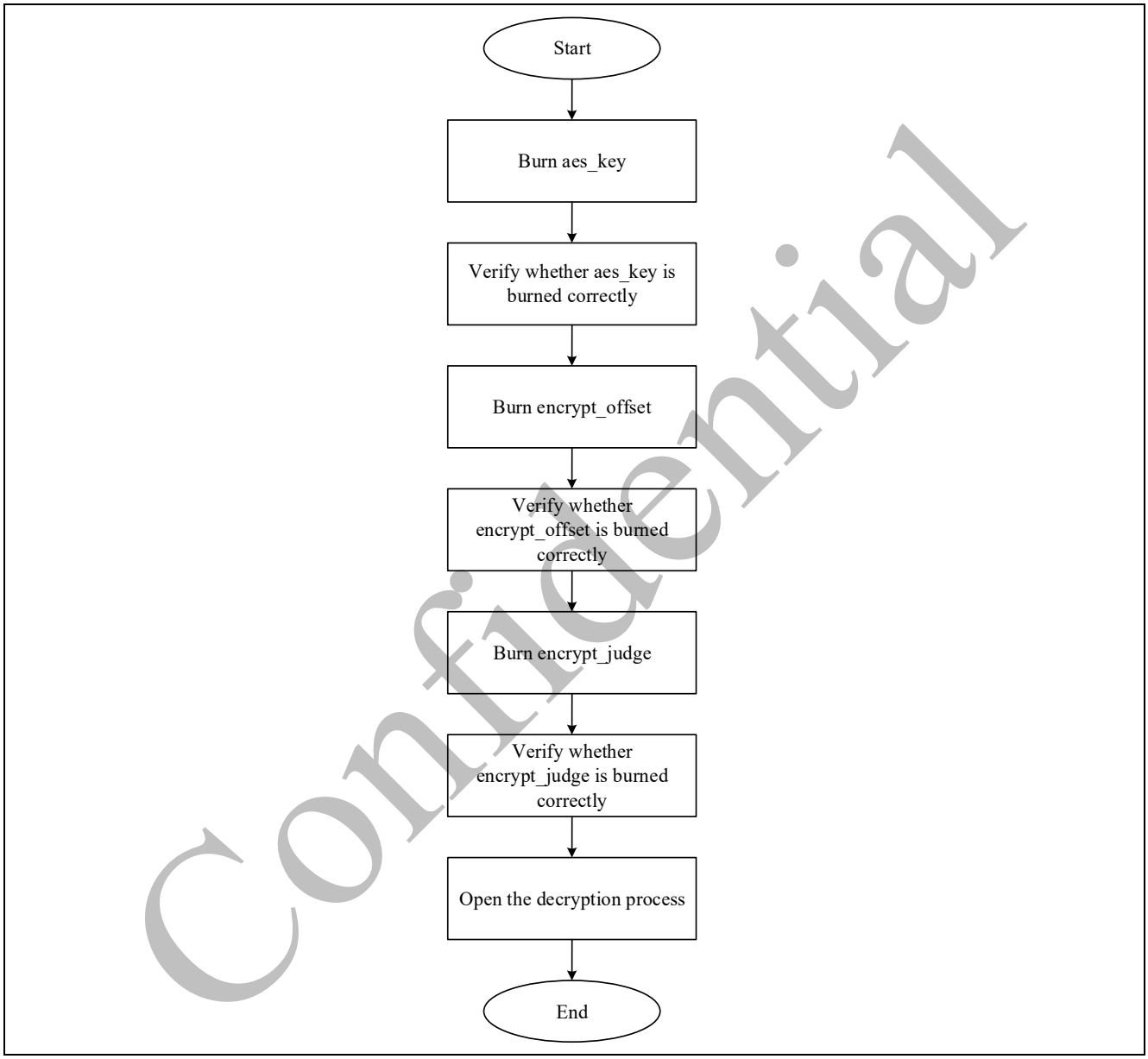


Figure 3-18 Efuse Encryption Area Burning Flowchart

Notes:

- *During the Burning process, the user should ensure that the encrypt_key, encrypt_offset and encrypt_judge burned in efuse cannot be wrong, which are the expected target data customized by the user.*
- *The focus of the AES encryption algorithm is to prevent the encrypt_key from being stolen, so after writing the encrypt_key to efuse, it should be ensured that no operation except*

hardware decryption has permission to read it in the application stage. This can be realized by the efuse controller hardware logic.

- *The encrypted information such as the chip key of the same batch should be the same. Even if the subsequent app program is updated, you only need to re-burn the flash, which is also convenient for the host computer to establish a database to store the encrypted information such as the key.*

3.5.5 Specific Protection Precautions

Prevent program injection:

The main purpose is to check whether the program has legally run to this block area. If it is legally run to this block area, operations such as fetching instructions or fetching data can be performed on this block area. The way to judge the legitimacy is in the effective bus transmission, the first behavior to access a certain area should be a fetch instructions (divide the flash area with the size of 256 BYTE), and the legitimacy is monitored and updated in real time. Generally speaking, if you fetch instructions in area A, then go to encrypted area B to fetch data, this is an illegal program injection operation. You need to fetch the instructions in the B area before accessing the data in the area. Violating this principle, the program may run away.

Prevent debug access:

It mainly monitors whether data is fetched in the encrypted area in debug mode. If it is directly shielded, the data 32'h00000000 is returned.

3.6 eFuse Controller

3.6.1 Overview

The eFuse can only be programmed once from 0 to 1, but it can be read multiple times. The main function is to store user configuration information, which can be used as security protection.

3.6.2 Features

- Support Byte read and write operation. Read operation supports up to 64MHz, write operation supports 32M/48M/64M, 48M/64M requires additional configuration of related registers, please refer to EFUSE_CTL, EFUSE_PROG_TIMING1, EFUSE_PROG_TIMING2, EFUSE_PROG_TIMING3 register descriptions. Moreover, the time to program a Byte is 56.25 μs(32M), the time to read a byte is 4.21875μs (32M, different from the read mode under load).
- When the chip is powered on, DVDD defaults to 1, and AVDD defaults to 0. When reading and writing to efuse, the software has the authority to configure the DVDD/AVDD switch.
- Support misoperation protection, and there is a misoperation flag register which is convenient to check whether there is a problem with this operation.
- Support hardware read and write protection for special encrypted information area.

3.7 DMA Serial Interface Controller (DMA)

3.7.1 Overview

The DMA in PAN101x series is an AMBA AHB module, and connects to the Advanced High-performance Bus (AHB). It can realize direct transmission of data without the participation of CPU, which can reduce the latency on the bus, improves system performance, and reduce power consumption greatly.

3.7.2 Features

- Up to 2 channels, one per source and destination pair
- One AHB master interface
- DMA to or from APB peripherals through the APB bridge
- Handshaking interfaces for source and destination peripherals (up to 16)
- DMA flow control, source flow control, destination flow control
- Support for memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral
- DMA transfers
- Little Endian

3.7.3 Block Diagram

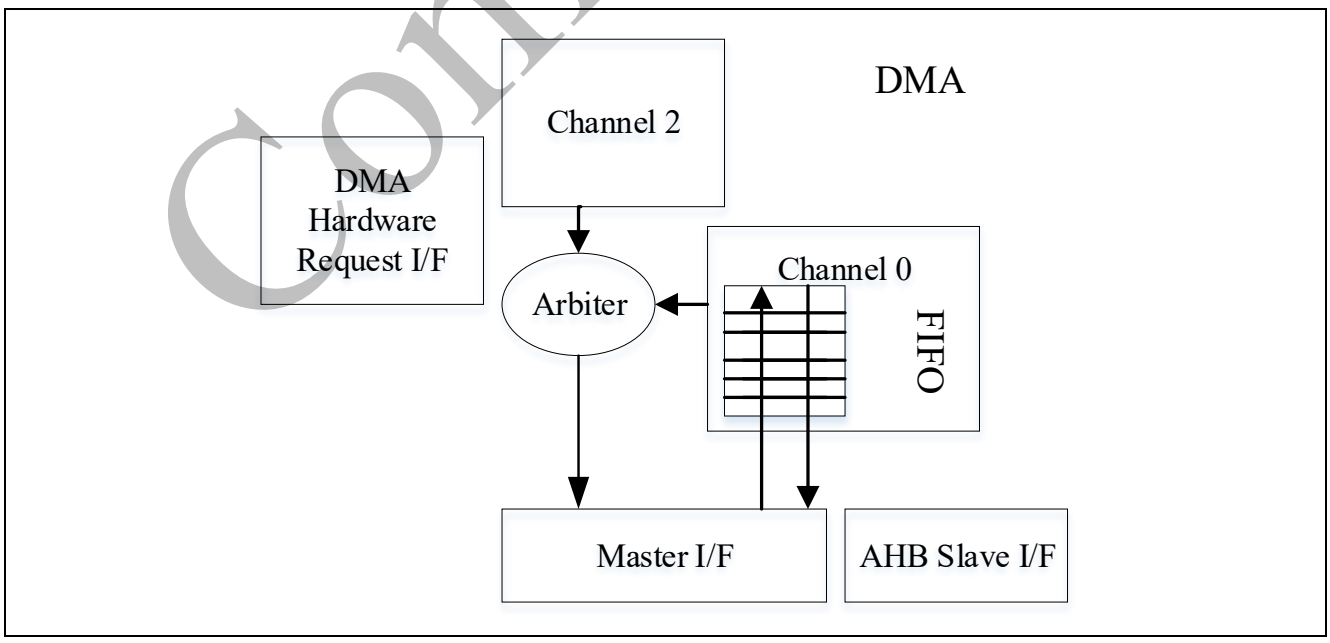


Figure 3-19 DMA Controller Block Diagram

3.7.4 Functional Description

3.7.4.1 AHB master interface

The DMA contains one full AHB masters. Figure 3-20 describes s a block diagram that shows the master connected into a system. This enables, for example, the DMAC to transfer data directly from the memory connected to AHB port 1 to any AHB peripheral connected to AHB port 2. It also enables transactions between the DMAC and any APB peripheral to occur independently of transactions on AHB bus 1. PAN101x series enables the peripherals, including UART, I2C and SPI, to interface to a DMA controller over the bus using a handshaking interface for transfer requests.

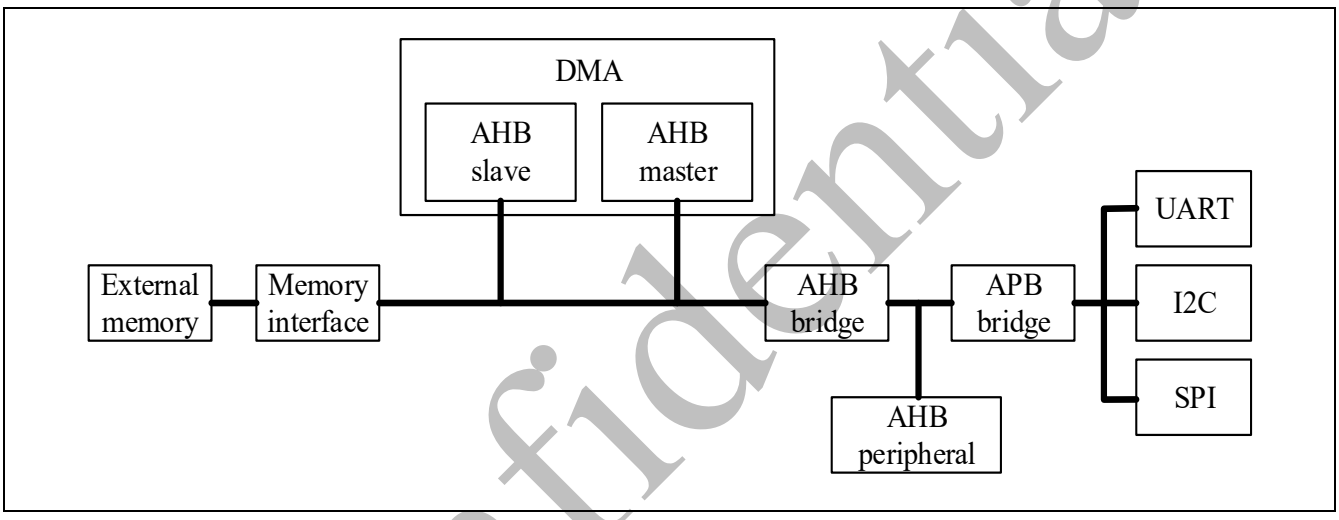


Figure 3-20 DMA Transfer Hierarchy

3.7.4.2 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or burst transactions. When the DMA is the flow controller, the DMA tries to efficiently transfer the data using as little of the bus bandwidth as possible.

A non-memory peripheral can request a DMA transfer through the DMA using one of two types of handshaking interfaces:

Hardware handshaking

- Figure 3-21 illustrates the hardware handshaking interface between a peripheral – whether a destination or source – and the DMA when the DMA is the flow controller.

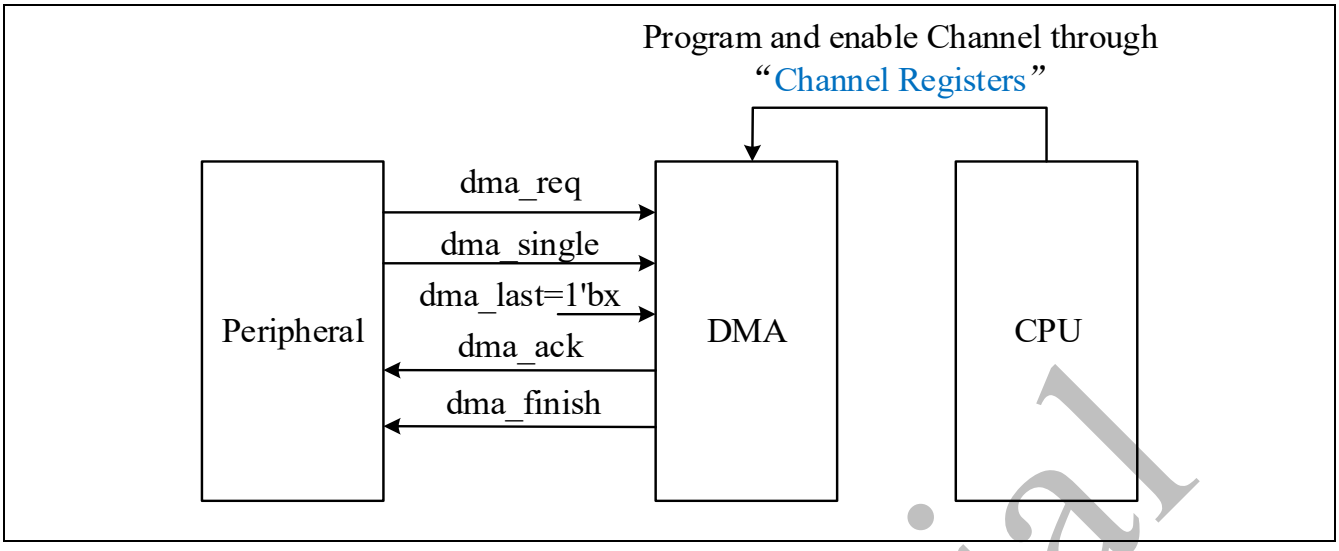


Figure 3-21 Hardware Handshaking Interface

Software handshaking

- When the slave peripheral requires the DMA to perform a DMA transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller. The interrupt service routine then uses the software register, which shows in the Figure 3-22.

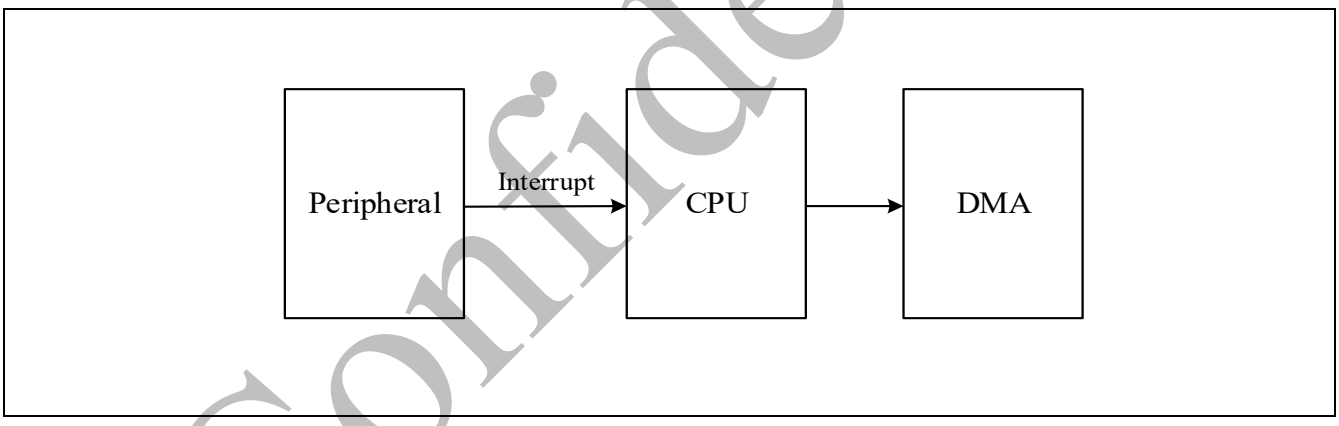


Figure 3-22 Software Controlled DMA Transfers

On completion of the transaction – single or burst – the relevant channel bit in the *SW_SOUR_TS_REQ/DEST_TS_REQ* register is cleared by hardware. Software can therefore poll this bit in order to determine when the requested transaction has completed. Alternatively, the IntSOUR_TC or IntDEST_TC interrupts can be enabled and unmasked in order to generate an interrupt when the requested transaction – single or burst – has completed.

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

3.7.4.3 Block Flow Controller and Transfer Type

The device that controls the length of a block is known as the flow controller. Either the DMA, the source peripheral, or the destination peripheral must be assigned as the flow controller. The PAN101x series surpports only the DMA to be the flow controller.

The following transfer types are supported:

- Peripheral to Peripheral
- Memory to Memory
- Memory to Peripheral
- Peripheral to Memory

The transfer types can be set through decoding the CTLx.ts_fw_ctl.

Table 3-9 lists the decoding for CTLx.ts_fw_ctl field.

Table 3-9 CTLx.ts_fw_ctl Field Decoding

CTLx.ts_fw_ctl Field	Transfer Type
000	Memory to Memory
001	Memory to Peripheral
010	Peripheral to Memory
011	Peripheral to Peripheral
others	-

Peripheral to Peripheral DMAC transfer

Figure 3-23 illustrates a peripheral-to-peripheral DMA transfer, where peripheral A (source) uses a hardware handshaking interface, and peripheral B (destination) uses a software handshaking interface. For example, the request to send data to peripheral B is originated by the CPU, while writing to peripheral B is handled by the DMA. The channel source and destination arbitrate independently for the AHB master, which are described in Arbitration for AHB Master Interface in detail.

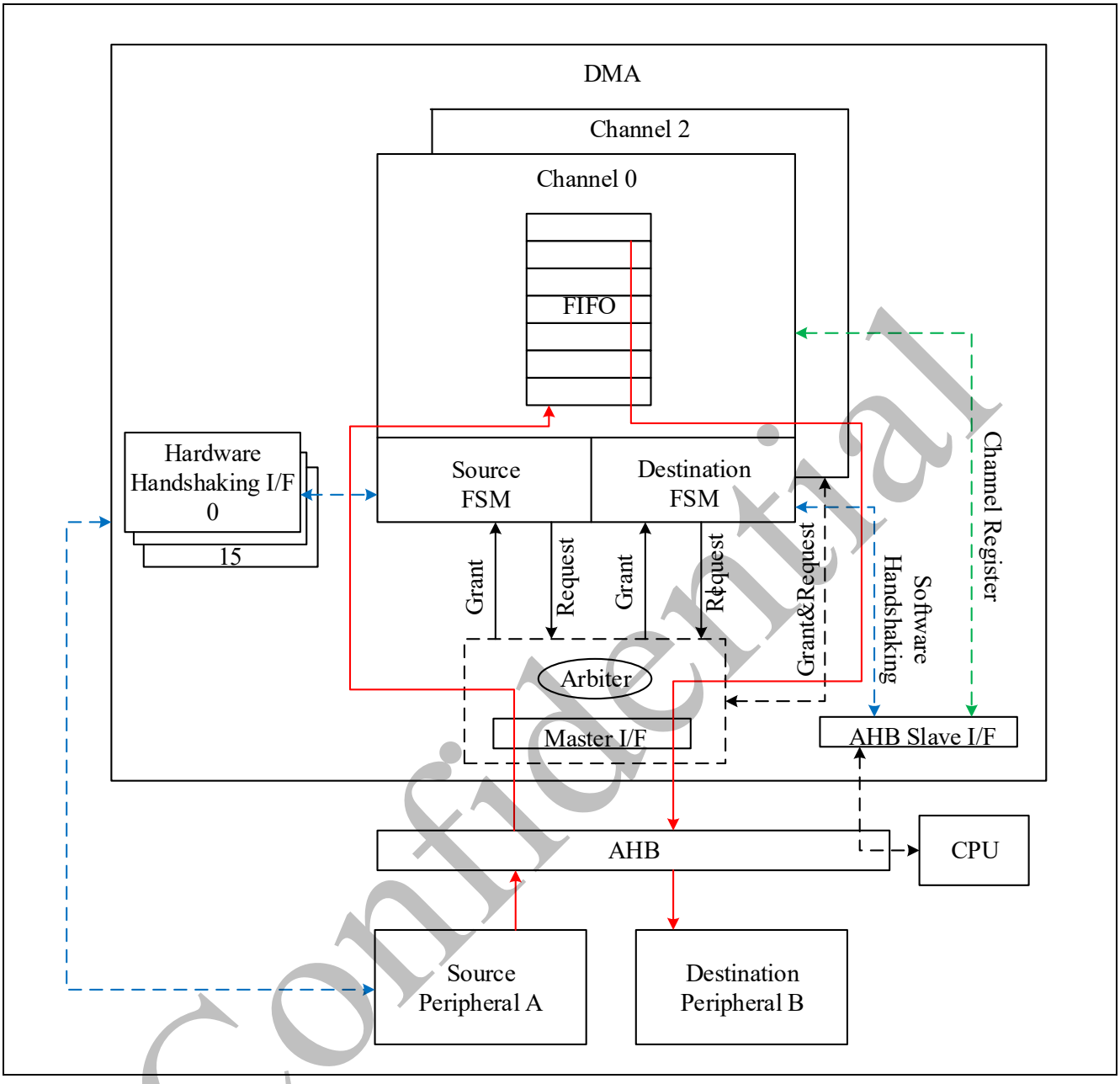


Figure 3-23 Peripheral-to-Peripheral DMA Transfer

3.7.4.4 Basic Interface Definitions

The following definitions are used in this chapter:

Source single transaction size in bytes

- src_single_size_bytes = CTLx.sour_ts_width/8 (1)

Source burst transaction size in bytes

- src_burst_size_bytes = CTLx.sour_bst_msize * src_single_size_bytes (2)

Destination single transaction size in bytes

- dst_single_size_bytes = CTLx.dest_ts_width/8 (3)

Destination burst transaction size in bytes

- dst_burst_size_bytes = CTLx.dest_bst_msize * dst_single_size_bytes (4)

Block size in bytes:

- DMA is flow controller – With the DMA as the flow controller, the processor programs the DMA with the number of data items (block size) of source transfer width (CTLx.sour_ts_width) to be transferred by the DMA in a block transfer; this is programmed into the CTLx.ts_num field. Therefore, the total number of bytes to be transferred in a block is:
- blk_size_bytes_DMA = CTLx.ts_num * src_single_size_bytes (5)

3.7.4.5 Single-block Transfer

The PAN101x series only supports single-block transfer. As shown in Figure 3-24, a single block is made up of numerous transactions – single and burst – which are in turn composed of AHB transfers. A peripheral requests a transaction through the handshaking interface to the DMA.

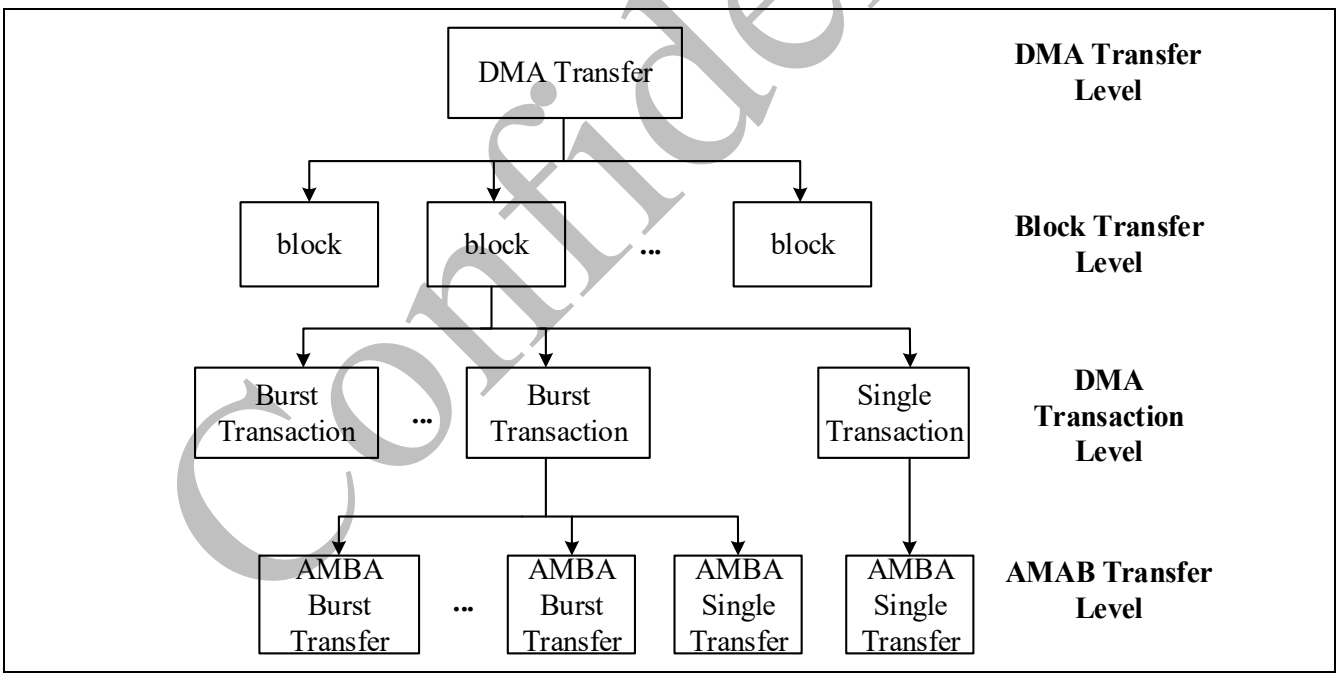


Figure 3-24 DMAC Transfer Hierarchy

The programing processes are shown as follow:

1 Read the Channel Enable register to choose a free (disabled) channel; refer to *CHANNEL_EN*.

2 Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: *INTSCL_DMA_TC, INTSCL_BLOK_TC,*

INTSCL_SOUR_TC, INTSCL_DEST_TC, and INTSCL_ERR. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.

3 Program the following channel registers:

1) Write the starting source address in the CH_SOUR_ADDRx register for channel x
2) Write the starting destination address in the CH_DEST_ADDRx register for channel x
3) Program CTLx and CH_CFGx.
4) Write the control information for the DMA transfer in the CTLx register for channel x. For example, in the register, you can program the following:
   a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the ts_fw_ctl of the CTLx register.
   b. Set up the transfer characteristics, such as:
      - Transfer width for the source in the sour_ts_width field.
      - Transfer width for the destination in the dest_ts_width field.
      - Incrementing/decrementing or fixed address for the source in the sour_addr_inc field.
      - Incrementing/decrementing or fixed address for the destination in the dest_addr_inc field.
5) Write the channel configuration information into the *CH_CFGx* register for channel x.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals; this is not required for memory.
   b. This step requires programming the swsour_hwhsh_sel/swdest_hwhsh_sel bits, respectively. Writing a 0 activates the hardware handshaking interface to handle source/destination requests.Writing a 1 activates the software handshaking interface to handle source and destination requests.
   c. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral; this requires programming the sour_interface and dest_interface bits, respectively.

4 Ensure that bit 0 of the *DMA_CFG* register is enabled before writing to *CHANNEL_EN*.

5 Source and destination request single and burst DMA transactions in order to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.

6 Once the transfer completes, hardware sets the interrupts and disables the channel. At this time, you can respond to either the Block Complete or Transfer Complete interrupts, or poll for the transfer complete raw interrupt status register (INTS_DMA_TC_RAW[n], n =

channel number) until it is set by hardware, in order to detect when the transfer is complete. Note that if this polling is used, the software must ensure that the transfer complete interrupt is cleared by writing to the Interrupt Clear register, INTSCL_DMA_TC[n], before the channel is enabled.

The flow diagram in Figure 3-25 shows an overview of programming the DMAC.

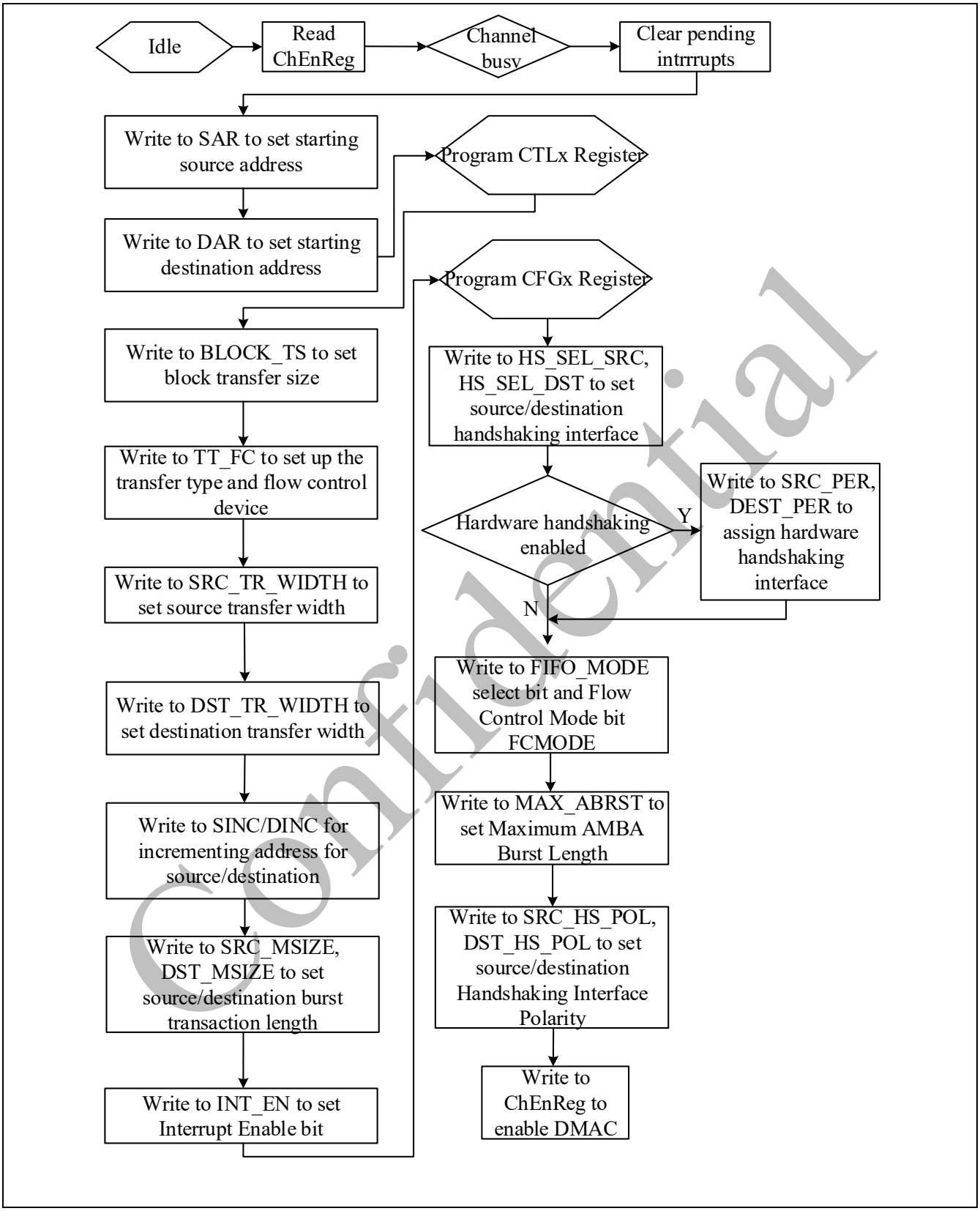


Figure 3-25 Flowchart for single-block DMAC Programming

3.7.4.6 Peripheral Burst Transaction Requests

For a source FIFO, an active edge is triggered on dma_req when the source FIFO exceeds some watermark level. For a destination FIFO, an active edge is triggered on dma_req when the destination FIFO drops below some watermark level.

This section investigates the optimal settings of these watermark levels on the source and destination peripherals and their relationship to, respectively:

- Source transaction length, CTLx.*sour_bst_msize*
- Destination transaction length, CTLx.*dest_bst_msize*

Transmit Watermark Level and Transmit FIFO Underflow

During SPI serial transfers, SPI transmit FIFO requests are made to the DMA whenever the number of entries in the SPI transmit FIFO is less than or equal to the SPI Transmit Data Level Register (SSI.DMATDLR) value. This is known as the watermark level. The DMA responds by writing a burst of data to the SPI transmit FIFO buffer, of length DMAC.CTLx.dest_bst_msize.

Data should be fetched from the DMA often enough for the SPI transmit FIFO to continuously perform serial transfers; that is, when the SPI transmit FIFO begins to empty, another burst transaction request should be triggered. Otherwise the SPI transmit FIFO runs out of data (underflow). To prevent this condition, the user must set the watermark level correctly.

Receive Watermark Level and Receive FIFO Overflow

During SPI serial transfers, SPI receive FIFO requests are made to the DMAC whenever the number of entries in the SPI receive FIFO is at or above the DMA Receive Data Level Register; that is, SSI.DMARDLR+1. This is known as the watermark level. The DMA responds by reading a burst of data from the receive FIFO buffer of length DMA.CTLx.sour_bst_msize.

Data should be fetched by the DMAC often enough for the SPI receive FIFO to accept serial transfers continuously; that is, when the SPI receive FIFO begins to fill, another burst transaction request should be triggered. Otherwise, the SPI receive FIFO fills with data (overflow). To prevent this condition, the user must correctly set the watermark level.

3.7.4.7 Generating Requests for the AHB Master Bus Interface

Each channel has a source state machine and destination state machine running in parallel. These state machines generate the request inputs to the arbiter, which arbitrates for the master bus interface (one arbiterper master bus interface).

When the source/destination state machine is granted control of the master bus interface, and when the master bus interface is granted control of the external AHB bus, then AHB transfers between the peripheral and the DMA (on behalf of the granted state machine) can take place.

AHB transfers from the source peripheral or to the destination peripheral cannot proceed until the channel FIFO is ready. For burst transaction requests and for transfers involving memory peripherals, the criterion for "FIFO readiness" is controlled by the ff_mode_sel field of the CH_CFGx register.

The definition of FIFO readiness is the same for:

- Single transactions
- Burst transactions, where ff_mode_sel = 0
- Transfers involving memory peripherals, where ff_mode_sel = 0

The channel FIFO is deemed ready when the space/data available is sufficient to complete a single AHB transfer of the specified transfer width. FIFO readiness for source transfers occurs when the channel FIFO contains enough room to accept at least a single transfer of sour_ts_width width. FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of dest_ts_width width.

When ff_mode_sel = 1, then the criteria for FIFO readiness for burst transaction requests and transfers involving memory peripherals are as follows:

- A FIFO is ready for a source burst transfer when the FIFO is less than half empty.
- A FIFO is ready for a destination burst transfer when the FIFO is greater than or equal to half full.

Exceptions to this "readiness" occur. During these exceptions, a value of ff_mode_sel = 0 is assumed.

The following are the exceptions:

- Near the end of a burst transaction or block transfer – The channel source state machine does not wait for the channel FIFO to be less than half empty if the number of source data items left to complete the source burst transaction or source block transfer is less than DMAH_CHx_FIFO_DEPTH/2. Similarly, the channel destination state machine does not wait for the channel FIFO to be greater than or equal to half full, if the number of destination data items left to complete the destination burst transaction or destination block transfer is less than DMAH_CHx_FIFO_DEPTH/2.
- When a channel is suspended – The destination state machine does not wait for the FIFO to become half empty to flush the FIFO, regardless of the value of the ff_mode_sel field.

- When the source/destination peripheral is not memory, the source/destination state machine waits for a single/burst transaction request. Upon receipt of a transaction request and only if the channel FIFO is“ready” for source/destination AHB transfers, a request for the master bus interface is made by the source/destination state machine.

3.7.4.8 Interrupt

There are five interrupt registers in the DMA:

- INT_BLOK_TC – Block Transfer Complete Interrupt
  This interrupt is generated on DMA block transfer completion to the destination peripheral.
- INT_DEST_TC – Destination Transaction Complete Interrupt
  This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the destination side.
- INT_ERR – Error Interrupt
  This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.
- INT_SOUR_TC – Source Transaction Complete Interrupt
  This interrupt is generated after completion of the last AHB transfer of the requested single/burst transaction from the handshaking interface (either the hardware or software handshaking interface) on the source side.
- INT_DMA_TC– DMA Transfer Complete Interrupt
  This interrupt is generated on DMA transfer completion to the destination peripheral.

When a channel has been enabled to generate interrupts, the following is true:

- Interrupt events are stored in the Raw Status registers.
- The contents of the Raw Status registers are masked with the contents of the Mask registers.
- The masked interrupts are stored in the Status registers.
- The contents of the Status registers are used to drive the int_* port signals.
- Writing to the appropriate bit in the Clear registers clears an interrupt in the Raw Status registers and the Status registers on the same clock cycle.

Upon occurrence of an error in an AHB transfer, the following occurs:

- DMA transfer in progress stops immediately
- Relevant channel is disabled,

- An interrupt is issued (if not masked)

If multiple channels are enabled, only the one where the AHB error was detected is disabled.

The contents of the FIFO are not cleared, but they become inaccessible and are overwritten once the channel is re-enabled to start a new sequence.

There is no support for automatically resuming the transfer from the point where the error occurred, and the full block transfer has to be re-initiated in order to be successfully completed.

The DMA does not use the hardware handshaking interface to signal the error occurrence in any way, nor does it signal the end of a transfer. In practice, this means that if a request from a peripheral is active when the error occurs—dma_req is high if peripheral is the flow controller; dma_req or dma_single are high if peripheral is not the flow controller—the channel is disabled without the DMA ever asserting dma_ack (or dma_finish).

The hardware handshaking interface on the peripheral side has to be re-initiated by the CPU upon detection of the error interrupt. The dma_req signal needs to be brought low before the channel is re-enabled and thenbrought high when the channel has been enabled.

3.7.4.9 Arbitration for AHB Master Interface

Each DMAC channel has two request lines that request ownership of a particular master bus interface: channel source and channel destination request lines.

Source and destination arbitrate separately for the bus. Once a source/destination state machine gains ownership of the master bus interface and the master bus interface has ownership of the AHB bus, then AHB transfers can proceed between the peripheral and DMA. Figure 3-26 illustrates the arbitration flow of the master bus interface.

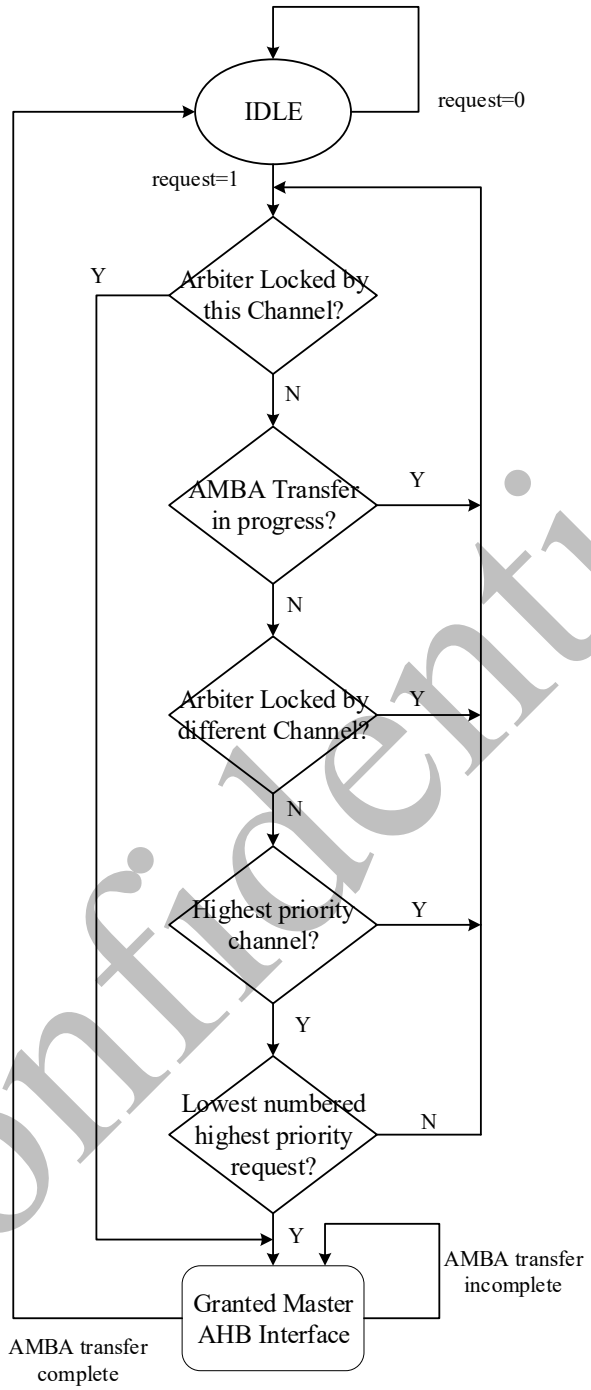


Figure 3-26 Arbitration Flow for Master Bus Interface

An arbitration scheme is used to decide which of the request lines (2*DMAH_chan_num) is granted the particular master bus interface. Each channel has a programmable priority. A request for the master bus interface can be made at any time, but is granted only after the current AHB transfer (burst or single) has completed. Therefore, if the master interface is transferring data for a lower priority channel and a higher priority channel requests service, then the master

interface will complete the current burst for the lower priority channel before switching to transfer data for the higher priority channel.

To prevent a channel from saturating the master bus interface, it can be given a maximum AMBA burst length (MAX_ABRST field in CH_CFGx register) at channel setup time. This also prevents the master bus interface from saturating the AHB bus where the system arbiter cannot change the grant lines until the end of an undefined length burst.

3.7.4.10 IP DMA Control

Table 3-10 IP DMA Control Map

IP request	Channel number
I2C0_TX	0
I2C0_RX	1
SPI0_TX	2
SPI0_RX	3
UART0_TX	4
UART0_RX	5
I2S_TX(M/S)	6
I2S_RX(M/S)	7
SPI1_TX	8
SPI1_RX	9
UART1_TX	10
UART1_RX	11
ADC	12

3.7.5 DMA Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
DMA Base Address: DMA_BA = 0x5000_1000				
CH_SOUR_ADDR0_L	DMA_BA+0x000	R/W	Source Address Register for Channel 0	0x0
CH_SOUR_ADDR0_H	DMA_BA+0x004	R/W	Source Address Register for Channel 0	0x0
CH_DEST_ADDR0_L	DMA_BA+0x008	R/W	Destination Address Register for Channel 0	0x0
CH_DEST_ADDR0_H	DMA_BA+0x00C	R/W	Destination Address Register for Channel 0	0x0
CTL0_L	DMA_BA+0x018	R/W	Control Register for Channel 0	Configuration dependent
CTL0_H	DMA_BA+0x01C	R/W	Control Register for Channel 0	Configuration dependent
CH_CFG0_L	DMA_BA+ 0x040	R/W	Configuration Register for Channel 0	0x0000_0E00
CH_CFG0_H	DMA_BA+ 0x044	R/W	Configuration Register for Channel 0	0x0000_0004
CH_SOUR_ADDR1_L	DMA_BA+0x058	R/W	Source Address Register for Channel 1	0x0
CH_SOUR_ADDR1_H	DMA_BA+0x05C	R/W	Source Address Register for Channel 1	0x0
CH_DEST_ADDR1_L	DMA_BA+0x060	R/W	Destination Address Register for Channel 1	0x0
CH_DEST_ADDR1_H	DMA_BA+0x064	R/W	Destination Address Register for Channel 1	0x0
CTL1_L	DMA_BA+0x070	R/W	Control Register for Channel 1	Configuration dependent
CTL1_H	DMA_BA+0x074	R/W	Control Register for Channel 1	Configuration dependent
CH_CFG1_L	DMA_BA+ 0x098	R/W	Configuration Register for Channel 1	0x0000_0E20
CH_CFG1_H	DMA_BA+ 0x09C	R/W	Configuration Register for Channel 1	0x0000_0004
CH_SOUR_ADDR2_L	DMA_BA+0x0B0	R/W	Source Address Register for Channel 2	0x0
CH_SOUR_ADDR2_H	DMA_BA+0x0B4	R/W	Source Address Register for Channel 2	0x0
CH_DEST_ADDR2_L	DMA_BA+0x0B8	R/W	Destination Address Register for Channel 2	0x0
CH_DEST_ADDR2_H	DMA_BA+0x0BC	R/W	Destination Address Register for Channel 2	0x0
CTL2_L	DMA_BA+0x0C8	R/W	Control Register for Channel 2	Configuration dependent
CTL2_H	DMA_BA+0x0CC	R/W	Control Register for Channel 2	Configuration dependent
CH_CFG2_L	DMA_BA+ 0x0F0	R/W	Configuration Register for Channel 2	0x0000_0E40
CH_CFG2_H	DMA_BA+ 0x0F4	R/W	Configuration Register for Channel 2	0x0000_0004
INTS_DMA_TC_RAW_L	DMA_BA+0x2C0	R/W	Interrupt Raw Status Registers	0x0
INTS_DMA_TC_RAW_H	DMA_BA+0x2C4	R/W	Interrupt Raw Status Registers	0x0
INTS_BLOK_TC_RAW_L	DMA_BA+0x2C8	R/W	Interrupt Raw Status Registers	0x0
INTS_BLOK_TC_RAW_H	DMA_BA+0x2CC	R/W	Interrupt Raw Status Registers	0x0

INTS_SOUR_TC_RAW_L	DMA_BA+ 0x2D0	R/W	Interrupt Raw Status Registers	0x0
INTS_SOUR_TC_RAW_H	DMA_BA+ 0x2D4	R/W	Interrupt Raw Status Registers	0x0
INTS_DEST_TC_RAW_L	DMA_BA+0x2D8	R/W	Interrupt Raw Status Registers	0x0
INTS_DEST_TC_RAW_H	DMA_BA+0x2DC	R/W	Interrupt Raw Status Registers	0x0
INTS_ERR_RAW_L	DMA_BA+0x2E0	R/W	Interrupt Raw Status Registers	0x0
INTS_ERR_RAW_H	DMA_BA+0x2E4	R/W	Interrupt Raw Status Registers	0x0
INTS_DMA_TC_L	DMA_BA+0x2E8	R	Interrupt Status Registers	0x0
INTS_DMA_TC_H	DMA_BA+0x2EC	R	Interrupt Status Registers	0x0
INTS_BLOK_TC_L	DMA_BA+0x2F0	R	Interrupt Status Registers	0x0
INTS_BLOK_TC_H	DMA_BA+0x2F4	R	Interrupt Status Registers	0x0
INTS_SOUR_TC_L	DMA_BA+0x2F8	R	Interrupt Status Registers	0x0
INTS_SOUR_TC_H	DMA_BA+0x2FC	R	Interrupt Status Registers	0x0
INTS_DEST_TC_L	DMA_BA+0x300	R	Interrupt Status Registers	0x0
INTS_DEST_TC_H	DMA_BA+0x304	R	Interrupt Status Registers	0x0
INTS_ERR_L	DMA_BA+0x308	R	Interrupt Status Registers	0x0
INTS_ERR_H	DMA_BA+0x30C	R	Interrupt Status Registers	0x0
INT_DMA_TC_MSK_L	DMA_BA+0x310	R/W	Interrupt Mask Registers	0x0
INT_DMA_TC_MSK_H	DMA_BA+0x314	R/W	Interrupt Mask Registers	0x0
INT_BLOK_TC_MSK_L	DMA_BA+0x318	R/W	Interrupt Mask Registers	0x0
INT_BLOK_TC_MSK_H	DMA_BA+0x31C	R/W	Interrupt Mask Registers	0x0
INT_SOUR_TC_MSK_L	DMA_BA+0x320	R/W	Interrupt Mask Registers	0x0
INT_SOUR_TC_MSK_H	DMA_BA+0x324	R/W	Interrupt Mask Registers	0x0
INT_DEST_TC_MSK_L	DMA_BA+0x328	R/W	Interrupt Mask Registers	0x0
INT_DEST_TC_MSK_H	DMA_BA+0x32C	R/W	Interrupt Mask Registers	0x0
INT_ERR_MSK_L	DMA_BA+0x330	R/W	Interrupt Mask Registers	0x0
INT_ERR_MSK_H	DMA_BA+0x334	R/W	Interrupt Mask Registers	0x0
INTSCL_DMA_TC_L	DMA_BA+0x338	W	Interrupt Clear Registers	0x0
INTSCL_DMA_TC_H	DMA_BA+0x33C	W	Interrupt Clear Registers	0x0
INTSCL_BLOK_TC_L	DMA_BA+0x340	W	Interrupt Clear Registers	0x0
INTSCL_BLOK_TC_H	DMA_BA+0x344	W	Interrupt Clear Registers	0x0
INTSCL_SOUR_TC_L	DMA_BA+0x348	W	Interrupt Clear Registers	0x0
INTSCL_SOUR_TC_H	DMA_BA+0x34C	W	Interrupt Clear Registers	0x0
INTSCL_DEST_TC_L	DMA_BA+0x350	W	Interrupt Clear Registers	0x0
INTSCL_DEST_TC_H	DMA_BA+0x354	W	Interrupt Clear Registers	0x0
INTSCL_ERR_L	DMA_BA+0x358	W	Interrupt Clear Registers	0x0
INTSCL_ERR_H	DMA_BA+0x35C	W	Interrupt Clear Registers	0x0
INTS_COMB_L	DMA_BA+0x360	R	Combined Interrupt Status Register	0x0
INTS_COMB_H	DMA_BA+0x364	R	Combined Interrupt Status Register	0x0
SW_SOUR_TS_REQ_L	DMA_BA+0x368	R/W	Source Software Transaction Request Register	0x0
SW_SOUR_TS_REQ_H	DMA_BA+0x36C	R/W	Source Software Transaction Request Register	0x0
DEST_TS_REQ_L	DMA_BA+0x370	R/W	Destination Software Transaction Request Register	0x0
DEST_TS_REQ_H	DMA_BA+0x374	R/W	Destination Software Transaction Request Register	0x0

SIG_SOUR_TS_REQ_L	DMA_BA+0x378	R/W	Destination Software Transaction Request Register	0x0
SIG_SOUR_TS_REQ_H	DMA_BA+0x37C	R/W	Destination Software Transaction Request Register	0x0
SIG_DEST_TS_REQ_L	DMA_BA+0x380	R/W	Destination Software Transaction Request Register	0x0
SIG_DEST_TS_REQ_H	DMA_BA+0x384	R/W	Destination Software Transaction Request Register	0x0
LST_SOUR_TS_REQ_L	DMA_BA+0x388	R/W	Last Source Transaction Request Register	0x0
LST_SOUR_TS_REQ_H	DMA_BA+0x38C	R/W	Last Source Transaction Request Register	0x0
LST_DEST_TS_REQ_L	DMA_BA+0x390	R/W	Last Destination Transaction Request Register	0x0
LST_DEST_TS_REQ_H	DMA_BA+0x394	R/W	Last Destination Transaction Request Register	0x0
DMA_CFG_L	DMA_BA+0x398	R	DMA Configuration Register	0x0
DMA_CFG_H	DMA_BA+0x39C	R	DMA Configuration Register	0x0
CHANNEL_EN_L	DMA_BA+0x3A0	R/W	DMAC Channel Enable Register	0x0
CHANNEL_EN_H	DMA_BA+0x3A4	R/W	DMAC Channel Enable Register	0x0
DMA_ID_REG_L	DMA_BA+0x3A8	R/W	DMA ID Register	0x0
DMA_ID_REG_H	DMA_BA+0x3AC	R/W	DMA ID Register	0x0
DMA_TEST_L	DMA_BA+0x3B0	/R/W	DMA·Test Register-	0x0
DMA_TEST_L	DMA_BA+0x3B4	R/W	DMA·Test Register-	0x0
DMA_COMP_PARMS_3_L	DMA_BA+0x3E0	R	DMAC Component Parameters Register 3	Depend on user configuration
DMA_COMP_PARMS_3_H	DMA_BA+0x3E4	R	DMAC Component Parameters Register 3	Depend on user configuration
DMA_COMP_PARMS_2_L	DMA_BA+0x3E8	R	DMAC Component Parameters Register 2	Depend on user configuration
DMA_COMP_PARMS_2_H	DMA_BA+0x3EC	R	DMAC Component Parameters Register 2	Depend on user configuration
DMA_COMP_PARMS_1_L	DMA_BA+0x3F0	R	DMAC Component Parameters Register 1	Depend on user configuration
DMA_COMP_PARMS_1_H	DMA_BA+0x3F4	R	DMAC Component Parameters Register 1	Depend on user configuration
DMA_COM_ID_L	DMA_BA+0x3F8	R	DMA Component ID Register	0x4457_1110
DMA_COM_ID_H	DMA_BA+0x3FC	R	DMA Component ID Register	0x0

3.7.6 DMA Register Description

3.7.6.1 Source Address Register for Channel_0 Register Low (CH_SOUR_ADDR0_L)

Register	Offset	R/W	Description	Reset Value
CH_SOUR_ADDR0_L	DMA_BA+0x000	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[31:0]	CH0_SOUR_ADDR	Current Source Address of DMA transfer. Updated after each source transfer. The sour_addr_inc field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

3.7.6.2 Source Address Register for Channel_0 Register High (CH_SOUR_ADDR0_H)

Register	Offset	R/W	Description	Reset Value
CH_SOUR_ADDR0_H	DMA_BA+0x004	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.3 Destination Address Register for Channel 0 Register Low(CH_DEST_ADDR0_L)

Register	Offset	R/W	Description	Reset Value
CH_DEST_ADDR0_L	DMA_BA+0x008	R/W	Destination Address Register for Channel 0	0x0

Bits	Description	
[31:0]	CH0_DEST_ADDR	Current Destination address of DMA transfer. Updated after each destination transfer. The DINC field in the CTL0 register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

3.7.6.4 Destination Address Register for Channel 0 Register High(CH_DEST_ADDR0_H)

Register	Offset	R/W	Description	Reset Value
CH_DEST_ADDR0_H	DMA_BA+0x00c	R/W	Destination Address Register for Channel 0	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.5 Control Register for Channel_0 Low (CTL0_L)

Register	Offset	R/W	Description	Reset Value
CTL0_L	DMA_BA+0x018	R/W	Control Register for Channel 0	0x0206_1201

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:23]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[22:20]</td><td>CH0_TS_FW_CTL</td><td>Transfer Type and Flow Control.
The following transfer types are supported.
Memory to Memory
Memory to Peripheral
Peripheral to Memory
Peripheral to Peripheral
Flow Control can be assigned to the DMA, the source peripheral, or the destination peripheral.
<table>
<tr><th>CTLx.ts_fw_ctl Field</th><th>Transfer Type</th><th>Flow Controller</th></tr>
<tr><td>000</td><td>Memory to Memory</td><td>DMA</td></tr>
<tr><td>001</td><td>Memory to Peripheral</td><td>DMA</td></tr>
<tr><td>010</td><td>Peripheral to Memory</td><td>DMA</td></tr>
<tr><td>011</td><td>Peripheral to Peripheral</td><td>DMA</td></tr>
<tr><td>100</td><td>Peripheral to Memory</td><td>Peripheral</td></tr>
<tr><td>101</td><td>Peripheral to Peripheral</td><td>Source Peripheral</td></tr>
<tr><td>110</td><td>Memory to Peripheral</td><td>Peripheral</td></tr>
<tr><td>111</td><td>Peripheral to Peripheral</td><td>Destination Peripheral</td></tr>
</table>
Reset value: 3’h011</td></tr>
<tr><td>[19:17]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[16:14]</td><td>CH0_SOUR_BST_MSIZE</td><td>Source Burst Transaction Length.
Number of data items, each of width sour_ts_width, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface.
Reset value: 3’h001</td></tr>
<tr><td>[13:11]</td><td>CH0_DEST_BST_MSIZE</td><td>Destination Burst Transaction Length.
Number of data items, each of width dest_ts_width, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface.
<table>
<tr><th>CTLx.sour_bst_msize / CTLx.dest_bst_msize</th><th>Number of data items to be transferred (of width CTLx.sour_ts_width or CTLx.dest_ts_width)</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>4</td></tr>
<tr><td>010</td><td>8</td></tr>
<tr><td>011</td><td>16</td></tr>
<tr><td>100</td><td>32</td></tr>
<tr><td>101</td><td>64</td></tr>
<tr><td>110</td><td>128</td></tr>
</table></td></tr>
</table>

		111 | 256 Reset value: 3'b001
[10:9]	CH0_SOUR_ADDR_INC	Source Address Increment. Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change Reset value: 2'b0
[8:7]	CH0_DEST_ADDR_INC	Destination Address Increment. Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change Reset value: 2'b0
[6:4]	CH0_SOUR_TS_WIDTH	Source Transfer Width. Mapped to AHB bus "hsize." For a non-memory peripheral, typically the peripheral (source) FIFO width. This value must be less than or equal to 32. Reset value: 3'b000
[3:1]	CH0_DEST_TS_WIDTH	Destination Transfer Width. For a non-memory peripheral, typically rgw peripheral (destination) FIFO width. This value must be less than or equal to 32. CTLx.sour_ts_width / CTLx.dest_ts_width | Size (bits) 000 | 8 001 | 16 010 | 32 011 | 64 100 | 128 101 | 256 11X | 256 Reset value: 3'b000
[0]	CH0_INT_ENABLE	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; raw* interrupt registers still assert if int_enable = 0. Reset value: 1'b1

3.7.6.6 Control Register for Channel_0 High (CTL0_H)

Register	Offset	R/W	Description	Reset Value
CTL0_H	DMA_BA+0x01c	R/W	Control Register for Channel 0	0x0000_0000

Bits	Description	
[31:13]	Reserved	Reserved.
[12]	CH0_DONE	Done bit If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set. Software can poll the LLI CTLx.done bit to see when a block transfer is complete. The LLI CTLx.done bit should be cleared when the linked lists are set up in memory prior to enabling the channel. LLI accesses are always 32-bit accesses (Hsize = 2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit. Reset value: 1'b0
[11:10]	Reserved	Reserved.
[9:0]	CH0_TS_NUM	When the DMA is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into ts_num indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat. **Width**: The width of the single transaction is determined by sour_ts_width. Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller. When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at 1023, but the actual block size can be greater. Reset value: 12'h002

3.7.6.7 Configuration Register for Channel_0 Low (CH_CFG0_L)

Register	Offset	R/W	Description	Reset Value
CH_CFG0_L	DMA_BA+ 0x040	R/W	Configuration Register for Channel 0	0x0000_0E00

Bits	Description	
[31]	CH0_DEST_RELOAD	Automatic Destination Reload. The CH_DEST_ADDRx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.
[30:20]	Reserved	Reserved
[19]	CH0_SOUR_HSH_POL	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
[18]	CH0_DEST_HSH_POL	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
[17:12]	Reserved	Reserved
[11]	CH0_SWSOUR_HWHSH_SEL	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
[10]	CH0_SWDEST_HWHSH_SEL	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware- initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
[9]	CH0_FIFO_EPT	Indicates if there is data left in the channel FIFO. Can be used inconjunction with CH_CFGx.ch_suspd to cleanly disable a channel. For more information. 1 = Channel FIFO empty 0 = Channel FIFO not empty
[8]	CH0_CH_SUSPD	Channel Suspend. Suspends all DMAC data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CH_CFGx.fifo_ept to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMAC transfer from the source

[7:5]	CH0_CH_PRIORITY	Channel priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range: 0~2 0: channel0 1: channel1 2: channel2 A programmed value outside this range will cause erroneous behavior.
[4:0]	Reserved	Reserved

3.7.6.8 Configuration Register for Channel_0 High(CH_CFG0_H)

Register	Offset	R/W	Description	Reset Value
CH_CFG0_H	DMA_BA+ 0x044	R/W	Configuration Register for Channel 0	0x0000_0004

Bits	Description	
[31:15]	Reserved	Reserved.
[14:11]	CH0_DEST_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the destination of channel x if the CH_CFGx.swdest_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. **Note**: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[10:7]	CH0_SOUR_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the source of channel x if the CH_CFGx.swsour_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface. Note: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[6:5]	Reserved	Reserved
[4:2]	CH0_PROT_CTL	Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. 1'b1 — HPROT[0] CH_CFGx.prot_ctl[1] — HPROT[1] CH_CFGx.prot_ctl[2]-> — HPROT[2] CH_CFGx.prot_ctl[3]-> — HPROT[3]
[1]	CH0_FF_MODE_SEL	FIFO Mode Select. Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.
[0]	CH0_FW_MODE_CTL	Flow Control Mode. Determines when source transaction requests are serviced when the

		Destination Peripheral is the flow controller. 0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled. 1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

3.7.6.9 Source Address Register for Channel_1Register Low (CH_SOUR_ADDR1_L)

Register	Offset	R/W	Description	Reset Value
CH_SOUR_ADDR1_L	DMA_BA+0x058	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[31:0]	CH1_SOUR_ADDR	Current Source Address of DMA transfer. Updated after each source transfer. The sour_addr_inc field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

3.7.6.10 Source Address Register for Channel_1Register High(CH_SOUR_ADDR1_H)

Register	Offset	R/W	Description	Reset Value
CH_SOUR_ADDR1_H	DMA_BA+0x05c	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.11 Destination Address Register for Channel 1 Register Low(CH_DEST_ADDR1_L)

Register	Offset	R/W	Description	Reset Value
CH_DEST_ADDR1_L	DMA_BA+0x060	R/W	Destination Address Register for Channel 1	0x0

Bits	Description	
[31:0]	CH1_DEST_ADDR	Current Destination address of DMA transfer. Updated after each destination transfer. The DINC field in the CTL1 register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

3.7.6.12 Destination Address Register for Channel 1 Register High(CH_DEST_ADDR1_H)

Register	Offset	R/W	Description	Reset Value
CH_DEST_ADDR1_H	DMA_BA+0x064	R/W	Destination Address Register for Channel 1	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.13 Control Register for Channel_1 Low (CTL1_L)

Register	Offset	R/W	Description	Reset Value
CTL1_L	DMA_BA+0x070	R/W	Control Register for Channel 0	0x0206_1201

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:23]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[22:20]</td><td>CH1_TS_FW_CTL</td><td>Transfer Type and Flow Control.
The following transfer types are supported.
Memory to Memory
Memory to Peripheral
Peripheral to Memory
Peripheral to Peripheral
Flow Control can be assigned to the DMA, the source peripheral, or the destination peripheral.
<table>
<tr><th>CTLx.ts_fw_ctl Field</th><th>Transfer Type</th><th>Flow Controller</th></tr>
<tr><td>000</td><td>Memory to Memory</td><td>DMA</td></tr>
<tr><td>001</td><td>Memory to Peripheral</td><td>DMA</td></tr>
<tr><td>010</td><td>Peripheral to Memory</td><td>DMA</td></tr>
<tr><td>011</td><td>Peripheral to Peripheral</td><td>DMA</td></tr>
<tr><td>100</td><td>Peripheral to Memory</td><td>Peripheral</td></tr>
<tr><td>101</td><td>Peripheral to Peripheral</td><td>Source Peripheral</td></tr>
<tr><td>110</td><td>Memory to Peripheral</td><td>Peripheral</td></tr>
<tr><td>111</td><td>Peripheral to Peripheral</td><td>Destination Peripheral</td></tr>
</table>
Reset value: 3’h011</td></tr>
<tr><td>[19:17]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[16:14]</td><td>CH1_SOUR_BST_MSIZE</td><td>Source Burst Transaction Length.
Number of data items, each of width sour_ts_width, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface.
Reset value: 3’h001</td></tr>
<tr><td>[13:11]</td><td>CH1_DEST_BST_MSIZE</td><td>Destination Burst Transaction Length.
Number of data items, each of width dest_ts_width, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface.
<table>
<tr><th>CTLx.sour_bst_msize / CTLx.dest_bst_msize</th><th>Number of data items to be transferred (of width CTLx.sour_ts_width or CTLx.dest_ts_width)</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>4</td></tr>
<tr><td>010</td><td>8</td></tr>
<tr><td>011</td><td>16</td></tr>
<tr><td>100</td><td>32</td></tr>
<tr><td>101</td><td>64</td></tr>
<tr><td>110</td><td>128</td></tr>
</table></td></tr>
</table>

		111 | 256 Reset value: 3'b001
[10:9]	CH1_SOUR_ADDR_INC	Source Address Increment. Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change Reset value: 2'b0
[8:7]	CH1_DEST_ADDR_INC	Destination Address Increment. Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change Reset value: 2'b0
[6:4]	CH1_SOUR_TS_WIDTH	Source Transfer Width. Mapped to AHB bus "hsize." For a non-memory peripheral, typically the peripheral (source) FIFO width. This value must be less than or equal to 32. Reset value: 3'b000
[3:1]	CH1_DEST_TS_WIDTH	Destination Transfer Width. For a non-memory peripheral, typically rgw peripheral (destination) FIFO width. This value must be less than or equal to 32. CTLx.sour_ts_width / CTLx.dest_ts_width | Size (bits) 000 | 8 001 | 16 010 | 32 011 | 64 100 | 128 101 | 256 11X | 256 Reset value: 3'b000
[0]	CH1_INT_ENABLE	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; raw* interrupt registers still assert if int_enable = 0. Reset value: 1'b1

3.7.6.14 Control Register for Channel_1 High (CTL1_H)

Register	Offset	R/W	Description	Reset Value
CTL1_H	DMA_BA+0x074	R/W	Control Register for Channel 0	0x0000_0000

Bits	Description	
[31:13]	Reserved	Reserved.
[12]	CH1_DONE	Done bit If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set. Software can poll the LLI CTLx.done bit to see when a block transfer is complete. The LLI CTLx.done bit should be cleared when the linked lists are set up in memory prior to enabling the channel. LLI accesses are always 32-bit accesses (Hsize = 2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit. Reset value: 1'b0
[11:10]	Reserved	Reserved.
[9:0]	CH1_TS_NUM	When the DMA is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into ts_num indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat. **Width**: The width of the single transaction is determined by sour_ts_width. Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller. When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at 1023, but the actual block size can be greater. Reset value: 12'h002

3.7.6.15 Configuration Register for Channel_1 Low(CH_CFG1_L)

Register	Offset	R/W	Description	Reset Value
CH_CFG1_L	DMA_BA+ 0x098	R/W	Configuration Register for Channel 1	0x0000_0E20

Bits	Description	
[30:20]	Reserved	Reserved
[19]	CH1_SOUR_HSH_POL	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
[18]	CH1_DEST_HSH_POL	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
[17:12]	Reserved	Reserved
[11]	CH1_SWSOUR_HWHSH_SEL	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
[10]	CH1_SWDEST_HWHSH_SEL	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware- initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
[9]	CH1_FIFO_EPT	Indicates if there is data left in the channel FIFO. Can be used inconjunction with CH_CFGx.ch_suspd to cleanly disable a channel. For more information. 1 = Channel FIFO empty 0 = Channel FIFO not empty
[8]	CH1_CH_SUSPD	Channel Suspend. Suspends all DMAC data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CH_CFGx.fifo_ept to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMAC transfer from the source
[7:5]	CH1_CH_PRIORITY	Channel priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range: 0~2 0: channel0

		1: channel1 2: channel2 A programmed value outside this range will cause erroneous behavior.
[4:0]	Reserved	Reserved

3.7.6.16 Configuration Register for Channel_1 High(CH_CFG1_H)

Register	Offset	R/W	Description	Reset Value
CH_CFG1_H	DMA_BA+0x09c	R/W	Configuration Register for Channel 1	0x0000_0004

Bits	Description	
[31:15]	Reserved	Reserved.
[14:11]	CH1_DEST_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the destination of channel x if the CH_CFGx.swdest_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. **Note**: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[10:7]	CH1_SOUR_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the source of channel x if the CH_CFGx.swsour_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface. Note: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[6:5]	Reserved	Reserved
[4:2]	CH1_PROT_CTL	Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. 1'b1 — HPROT[0] CH_CFGx.prot_ctl[1] — HPROT[1] CH_CFGx.prot_ctl[2]-> — HPROT[2] CH_CFGx.prot_ctl[3]-> — HPROT[3]
[1]	CH1_FF_MODE_SEL	FIFO Mode Select. Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.
[0]	CH1_FW_MODE_CTL	Flow Control Mode. Determines when source transaction requests are serviced when the

		Destination Peripheral is the flow controller. 0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled. 1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

3.7.6.17 Source Address Register for Channel_2 Register Low (CH_SOUR_ADDR2_L)

Register	Offset	R/W	Description	Reset Value
CH_SOUR_ADDR2_L	DMA_BA+0x0b0	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[31:0]	CH2_SOUR_ADDR	Current Source Address of DMA transfer. Updated after each source transfer. The sour_addr_inc field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

3.7.6.18 Source Address Register for Channel_2 Register High(CH_SOUR_ADDR2_H)

Register	Offset	R/W	Description	Reset Value
CH_SOUR_ADDR2_H	DMA_BA+0x0b4	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.19 Destination Address Register for Channel 2 Register Low(CH_DEST_ADDR2_L)

Register	Offset	R/W	Description	Reset Value
CH_DEST_ADDR2_L	DMA_BA+0x0b8	R/W	Destination Address Register for Channel 2	0x0

Bits	Description	
[31:0]	CH2_DEST_ADDR	Current Destination address of DMA transfer. Updated after each destination transfer. The DINC field in the CTL2 register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

3.7.6.20 Destination Address Register for Channel 2 Register High(CH_DEST_ADDR2_H)

Register	Offset	R/W	Description	Reset Value
CH_DEST_ADDR2_H	DMA_BA+0x0bc	R/W	Destination Address Register for Channel 2	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.21 Control Register for Channel_2 Low (CTL2_L)

Register	Offset	R/W	Description	Reset Value
CTL2_L	DMA_BA+0x0C8	R/W	Control Register for Channel 0	0x0206_1201

<table>
<tr><th>Bits</th><th>Description</th><th></th></tr>
<tr><td>[31:23]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[22:20]</td><td>CH2_TS_FW_CTL</td><td>Transfer Type and Flow Control.
The following transfer types are supported.
Memory to Memory
Memory to Peripheral
Peripheral to Memory
Peripheral to Peripheral
Flow Control can be assigned to the DMA, the source peripheral, or the destination peripheral.
<table>
<tr><th>CTLx.ts_fw_ctl Field</th><th>Transfer Type</th><th>Flow Controller</th></tr>
<tr><td>000</td><td>Memory to Memory</td><td>DMA</td></tr>
<tr><td>001</td><td>Memory to Peripheral</td><td>DMA</td></tr>
<tr><td>010</td><td>Peripheral to Memory</td><td>DMA</td></tr>
<tr><td>011</td><td>Peripheral to Peripheral</td><td>DMA</td></tr>
<tr><td>100</td><td>Peripheral to Memory</td><td>Peripheral</td></tr>
<tr><td>101</td><td>Peripheral to Peripheral</td><td>Source Peripheral</td></tr>
<tr><td>110</td><td>Memory to Peripheral</td><td>Peripheral</td></tr>
<tr><td>111</td><td>Peripheral to Peripheral</td><td>Destination Peripheral</td></tr>
</table>
Reset value: 3’h011</td></tr>
<tr><td>[19:17]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[16:14]</td><td>CH2_SOUR_BST_MSIZE</td><td>Source Burst Transaction Length.
Number of data items, each of width sour_ts_width, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface.
Reset value: 3’h001</td></tr>
<tr><td>[13:11]</td><td>CH2_DEST_BST_MSIZE</td><td>Destination Burst Transaction Length.
Number of data items, each of width dest_ts_width, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface.
<table>
<tr><th>CTLx.sour_bst_msize / CTLx.dest_bst_msize</th><th>Number of data items to be transferred (of width CTLx.sour_ts_width or CTLx.dest_ts_width)</th></tr>
<tr><td>000</td><td>1</td></tr>
<tr><td>001</td><td>4</td></tr>
<tr><td>010</td><td>8</td></tr>
<tr><td>011</td><td>16</td></tr>
<tr><td>100</td><td>32</td></tr>
<tr><td>101</td><td>64</td></tr>
<tr><td>110</td><td>128</td></tr>
</table></td></tr>
</table>

		111 | 256 Reset value: 3'b001
[10:9]	CH2_SOUR_ADDR_INC	Source Address Increment. Indicates whether to increment or decrement the source address on every source transfer. If the device is fetching data from a source peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change Reset value: 2'b0
[8:7]	CH2_DEST_ADDR_INC	Destination Address Increment. Indicates whether to increment or decrement the destination address on every destination transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to "No change." 00 = Increment 01 = Decrement 1x = No change Reset value: 2'b0
[6:4]	CH2_SOUR_TS_WIDTH	Source Transfer Width. Mapped to AHB bus "hsize." For a non-memory peripheral, typically the peripheral (source) FIFO width. This value must be less than or equal to 32. Reset value: 3'b000
[3:1]	CH2_DEST_TS_WIDTH	Destination Transfer Width. For a non-memory peripheral, typically rgw peripheral (destination) FIFO width. This value must be less than or equal to 32. CTLx.sour_ts_width / CTLx.dest_ts_width | Size (bits) 000 | 8 001 | 16 010 | 32 011 | 64 100 | 128 101 | 256 11X | 256 Reset value: 3'b000
[0]	CH2_INT_ENABLE	Interrupt Enable Bit. If set, then all interrupt-generating sources are enabled. Functions as a global mask bit for all interrupts for the channel; raw* interrupt registers still assert if int_enable = 0. Reset value: 1'b1

3.7.6.22 Control Register for Channel_2 High (CTL2_H)

Register	Offset	R/W	Description	Reset Value
CTL2_H	DMA_BA+0x0cc	R/W	Control Register for Channel 0	0x0000_0000

Bits	Description	
[31:13]	Reserved	Reserved.
[12]	CH2_DONE	Done bit If status write-back is enabled, the upper word of the control register, CTLx[63:32], is written to the control register location of the Linked List Item (LLI) in system memory at the end of the block transfer with the done bit set. Software can poll the LLI CTLx.done bit to see when a block transfer is complete. The LLI CTLx.done bit should be cleared when the linked lists are set up in memory prior to enabling the channel. LLI accesses are always 32-bit accesses (Hsize = 2) aligned to 32-bit boundaries and cannot be changed or programmed to anything other than 32-bit. Reset value: 1'b0
[11:10]	Reserved	Reserved.
[9:0]	CH2_TS_NUM	When the DMA is the flow controller, the user writes this field before the channel is enabled in order to indicate the block size. The number programmed into ts_num indicates the total number of single transactions to perform for every block transfer; a single transaction is mapped to a single AMBA beat. **Width**: The width of the single transaction is determined by sour_ts_width. Once the transfer starts, the read-back value is the total number of data items already read from the source peripheral, regardless of what is the flow controller. When the source or destination peripheral is assigned as the flow controller, then the maximum block size that can be read back saturates at 1023, but the actual block size can be greater. Reset value: 12'h002

3.7.6.23 Configuration Register for Channel_2 Low(CH_CFG2_L)

Register	Offset	R/W	Description	Reset Value
CH_CFG2_L	DMA_BA+ 0x0f0	R/W	Configuration Register for Channel 2	0x0000_0E40

Bits	Description	
[63:47]	Reserved	Reserved.
[46:43]	CH2_DEST_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the destination of channel x if the CH_CFGx.swdest_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. **Note**: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[42:39]	CH2_SOUR_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the source of channel x if the CH_CFGx.swsour_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface. Note: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[38:37]	Reserved	Reserved
[36:34]	CH2_PROT_CTL	Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. 1'b1 → HPROT[0] CH_CFGx.prot_ctl[1] → HPROT[1] CH_CFGx.prot_ctl[2]-> → HPROT[2] CH_CFGx.prot_ctl[3]-> → HPROT[3]
[33]	CH2_FF_MODE_SEL	FIFO Mode Select. Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.
[32]	CH2_FW_MODE_CTL	Flow Control Mode. Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.

		0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled. 1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.
[31]	CH2_DEST_RELOAD	Automatic Destination Reload. The CH_DEST_ADDRx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.
[30:20]	Reserved	Reserved
[19]	CH2_SOUR_HSH_POL	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
[18]	CH2_DEST_HSH_POL	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
[17:12]	Reserved	Reserved
[11]	CH2_SWSOUR_HWHSH_SEL	Source Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for source requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware-initiated transaction requests are ignored. If the source peripheral is memory, then this bit is ignored.
[10]	CH2_SWDEST_HWHSH_SEL	Destination Software or Hardware Handshaking Select. This register selects which of the handshaking interfaces – hardware or software – is active for destination requests on this channel. 0 = Hardware handshaking interface. Software-initiated transaction requests are ignored. 1 = Software handshaking interface. Hardware- initiated transaction requests are ignored. If the destination peripheral is memory, then this bit is ignored.
[9]	CH2_FIFO_EPT	Indicates if there is data left in the channel FIFO. Can be used inconjunction with CH_CFGx.ch_suspd to cleanly disable a channel. For more information. 1 = Channel FIFO empty 0 = Channel FIFO not empty
[8]	CH2_CH_SUSPD	Channel Suspend. Suspends all DMAC data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with CH_CFGx.fifo_ept to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMAC transfer from the source

[7:5]	CH2_CH_PRIORITY	Channel priority. A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range: 0~2 0: channel0 1: channel1 2: channel2 A programmed value outside this range will cause erroneous behavior.
[4:0]	Reserved	Reserved

3.7.6.24 Configuration Register for Channel_2 High(CH_CFG2_H)

Register	Offset	R/W	Description	Reset Value
CH_CFG2_H	DMA_BA+ 0x0f4	R/W	Configuration Register for Channel 2	0x0000_0004

Bits	Description	
[31:15]	Reserved	Reserved.
[14:11]	CH2_DEST_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the destination of channel x if the CH_CFGx.swdest_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface through the assigned hardware handshaking interface. **Note**: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[10:7]	CH2_SOUR_INTERFACE	Assigns a hardware handshaking interface (0 - 15) to the source of channel x if the CH_CFGx.swsour_hwhsh_sel field is 0; otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface through the assigned hardware handshaking interface. Note: For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.
[6:5]	Reserved	Reserved
[4:2]	CH2_PROT_CTL	Protection Control bits used to drive the AHB HPROT[3:1] bus. The AMBA Specification recommends that the default value of HPROT indicates a non-cached, non-buffered, privileged data access. The reset value is used to indicate such an access. HPROT[0] is tied high because all transfers are data accesses, as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. 1'b1 — HPROT[0] CH_CFGx.prot_ctl[1] — HPROT[1] CH_CFGx.prot_ctl[2]-> — HPROT[2] CH_CFGx.prot_ctl[3]-> — HPROT[3]
[1]	CH2_FF_MODE_SEL	FIFO Mode Select. Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced. 0 = Space/data available for single AHB transfer of the specified transfer width. 1 = Data available is greater than or equal to half the FIFO depth for destination transfers and space available is greater than half the fifo depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.
[0]	CH2_FW_MODE_CTL	Flow Control Mode. Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.

		0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled. 1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode, the amount of data transferred from the source is limited so that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

3.7.6.25 DMA Transfer Complete Interrupt Raw Status Registers Low (INTS_DMA_TC_RAW_L)

Register	Offset	R/W	Description	Reset Value
INTS_DMA_TC_RAW_L	DMA_BA+0x2c0	R/W	DMA Transfer Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_DMA_TC_RAW	DMA Transfer Complete Interrupt Raw Status Registers

3.7.6.26 DMA Transfer Complete Interrupt Raw Status Registers high (INTS_DMA_TC_RAW_H)

Register	Offset	R/W	Description	Reset Value
INTS_DMA_TC_RAW_H	DMA_BA+0x2c4	R/W	DMA Transfer Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.27 Block Transfer Complete Interrupt Raw Status Registers Low (INTS_BLOK_TC_RAW_L)

Register	Offset	R/W	Description	Reset Value
INTS_BLOK_TC_RAW_L	DMA_BA+0x2c8	R/W	Block Transfer Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_BLOK_TC_RAW	Block Transfer Complete Interrupt Raw Status Registers

3.7.6.28 Block Transfer Complete Interrupt Raw Status Registers High (INTS_BLOK_TC_RAW_H)

Register	Offset	R/W	Description	Reset Value
INTS_BLOK_TC_RAW	DMA_BA+0x2cc	R/W	Block Transfer Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.29 Source Transaction Complete Interrupt Raw Status Registers Low

Register	Offset	R/W	Description	Reset Value
INTS_SOUR_TC_RAW_L	DMA_BA+0x2d0	R/W	Source Transaction Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_SOUR_TC_RAW	Source Transaction Complete Interrupt Raw Status Registers

3.7.6.30 Source Transaction Complete Interrupt Raw Status Registers High

Register	Offset	R/W	Description	Reset Value
INTS_SOUR_TC_RAW_H	DMA_BA+0x2d4	R/W	Source Transaction Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.31 Destination Transaction Complete Interrupt Raw Status Registers Low

Register	Offset	R/W	Description	Reset Value
INTS_DETR_TC_RAW_L	DMA_BA+0x2d8	R/W	Destination Transaction Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_DEST_TC_RAW	Destination Transaction Complete Interrupt Raw Status Registers

3.7.6.32 Destination Transaction Complete Interrupt Raw Status Registers High

Register	Offset	R/W	Description	Reset Value
INTS_DETR_TC_RAW_H	DMA_BA+0x2dc	R/W	Destination Transaction Complete Interrupt Raw Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.33 Error Interrupt Raw Status Registers Low

Register	Offset	R/W	Description	Reset Value
INTS_ERR_RAW_L	DMA_BA+0x2e0	R/W	Error Interrupt Raw Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_ERR_RAW	Error Interrupt Raw Status Registers

3.7.6.34 Error Interrupt Raw Status Registers High

Register	Offset	R/W	Description	Reset Value
INTS_ERR_RAW_H	DMA_BA+0x2e4	R/W	Error Interrupt Raw Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.35 DMA Transfer Complete Interrupt Status Registers Low(INTS_DMA_TC_L)

Register	Offset	R/W	Description	Reset Value
INTS_DMA_TC_L	DMA_BA+0x2e8	R/W	DMA Transfer Complete Interrupt Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_DMA_TC	DMA Transfer Complete Interrupt Status Registers

3.7.6.36 DMA Transfer Complete Interrupt Status Registers High(INTS_DMA_TC_H)

Register	Offset	R/W	Description	Reset Value
INTS_DMA_TC_H	DMA_BA+0x2ec	R/W	DMA Transfer Complete Interrupt Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.37 Block Transfer Complete Interrupt Status Registers Low(INTS_BLOK_TC_L)

Register	Offset	R/W	Description	Reset Value
INTS_BLOK_TC_L	DMA_BA+0x2f0	R/W	Block Transfer Complete Interrupt Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_BLOK_TC	Block Transfer Complete Interrupt Status Registers

3.7.6.38 Block Transfer Complete Interrupt Status Registers High(INTS_BLOK_TC_H)

Register	Offset	R/W	Description	Reset Value
INTS_BLOK_TC_H	DMA_BA+0x2f4	R/W	Block Transfer Complete Interrupt Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.39 Source Transaction Complete Interrupt Status Registers Low (INTS_SOUR_TC_L)

Register	Offset	R/W	Description	Reset Value
INTS_SOUR_TC_L	DMA_BA+0x2f8	R/W	Source Transaction Complete Interrupt Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_SOUR_TC	Source Transaction Complete Interrupt Status Registers

3.7.6.40 Source Transaction Complete Interrupt Status Registers High(INTS_SOUR_TC_H)

Register	Offset	R/W	Description	Reset Value
INTS_SOUR_TC_H	DMA_BA+0x2fc	R/W	Source Transaction Complete Interrupt Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.41 Destination Transaction Complete Interrupt Status Registers Low

Register	Offset	R/W	Description	Reset Value
INTS_DETR_TC_L	DMA_BA+0x300	R/W	Destination Transaction Complete Interrupt Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.
[2:0]	INTS_DEST_TC	Destination Transaction Complete Interrupt Status Registers

3.7.6.42 Destination Transaction Complete Interrupt Status Registers High

Register	Offset	R/W	Description	Reset Value
INTS_DETR_TC_H	DMA_BA+0x304	R/W	Destination Transaction Complete Interrupt Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.43 Error Interrupt Status Registers Low(INTS_ERR_L)

Register	Offset	R/W	Description	Reset Value
INTS_ERR_L	DMA_BA+0x308	R/W	Error Interrupt Status Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTS_ERR	Error Interrupt Status Registers

3.7.6.44 Error Interrupt Status Registers High (INTS_ERR_H)

Register	Offset	R/W	Description	Reset Value
INTS_ERR_H	DMA_BA+0x30c	R/W	Error Interrupt Status Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.45 DMA Transfer Complete Interrupt Mask Registers Low(INT_DMA_TC_MSK_L)

Register	Offset	R/W	Description	Reset Value
INT_DMA_TC_MSK_L	DMA_BA+0x310	R/W	DMA Transfer Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	INT_DMA_TC_MSK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	INT_DMA_TC_MSK	Interrupt Mask 0 = masked 1 = unmasked **Note**: A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation.

3.7.6.46 DMA Transfer Complete Interrupt Mask Registers High(INT_DMA_TC_MSK_H)

Register	Offset	R/W	Description	Reset Value
INT_DMA_TC_MSK_H	DMA_BA+0x314	R/W	DMA Transfer Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.47 Block Transfer Complete Interrupt Mask Registers Low(INT_BLOK_TC_MSK_L)

Register	Offset	R/W	Description	Reset Value
INT_BLOK_TC_MSK_L	DMA_BA+0318	R/W	Block Transfer Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	INT_BLOK_TC_MSK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	INT_BLOK_TC_MSK	Interrupt Mask 0 = masked 1 = unmasked **Note**: A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation.

3.7.6.48 Block Transfer Complete Interrupt Mask Registers High(INT_BLOK_TC_MSK_H)

Register	Offset	R/W	Description	Reset Value
INT_BLOK_TC_MSK_H	DMA_BA+031c	R/W	Block Transfer Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.49 Source Transaction Complete Interrupt Mask Registers (INT_SOUR_TC_MSK_L)

Register	Offset	R/W	Description	Reset Value
INT_SOUR_TC_MSK_L	DMA_BA+0x320	R/W	Source Transaction Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	INT_SOUR_TC_MSK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	INT_SOUR_TC_MSK	Interrupt Mask 0 = masked 1 = unmasked **Note**: A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation.

3.7.6.50 Source Transaction Complete Interrupt Mask Registers(INT_SOUR_TC_MSK_H)

Register	Offset	R/W	Description	Reset Value
INT_SOUR_TC_MSK_H	DMA_BA+0x324	R/W	Source Transaction Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.51 Destination Transaction Complete Interrupt Msk Registers(INT_DEST_TC_MSK_L)

Register	Offset	R/W	Description	Reset Value
INT_DEST_TC_MSK_L	DMA_BA+0x328	R/W	Destination Transaction Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	INT_DEST_TC_MSK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	INT_DEST_TC_MSK	Interrupt Mask 0 = masked 1 = unmasked **Note**: A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation.

3.7.6.52 Destination Transaction Complete Interrupt Msk Registers(INT_DEST_TC_MSK_H)

Register	Offset	R/W	Description	Reset Value
INT_DEST_TC_MSK_H	DMA_BA+0x32c	R/W	Destination Transaction Complete Interrupt Mask Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.53 Error Interrupt Raw Mask Registers Low(INT_ERR_MSK_L)

Register	Offset	R/W	Description	Reset Value
INT_ERR_MSK_L	DMA_BA+0x330	R/W	Error Interrupt Mask Registers	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	INT_ERR_MSK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	INT_ERR_MSK	Interrupt Mask 0 = masked 1 = unmasked **Note**: A channel INT_MASK bit will be written only if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AHB write transfer. This allows software to set a mask bit without performing a read-modified write operation.

3.7.6.54 Error Interrupt Raw Mask Registers High(INT_ERR_MSK_H)

Register	Offset	R/W	Description	Reset Value
INT_ERR_MSK_H	DMA_BA+0x334	R/W	Error Interrupt Mask Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.55 DMA Transfer Complete Interrupt Status Clear Registers (INTSCL_DMA_TC_L)

Register	Offset	R/W	Description	Reset Value
INTSCL_DMA_TC_L	DMA_BA+0x338	R/W	DMA Transfer Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTSCL_DMA_TC	DMA Transfer Complete Interrupt Status Registers

3.7.6.56 DMA Transfer Complete Interrupt Status Clear Registers(INTSCL_DMA_TC_H)

Register	Offset	R/W	Description	Reset Value
INTSCL_DMA_TC_H	DMA_BA+0x33c	R/W	DMA Transfer Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.57 Block Transfer Complete Interrupt Status Clear Registers(INTSCL_BLOK_TC_L)

Register	Offset	R/W	Description	Reset Value
INTSCL_BLOK_TC_L	DMA_BA+0x340	R/W	Block Transfer Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTSCL_BLOK_TC	Block Transfer Complete Interrupt Status Registers

3.7.6.58 Block Transfer Complete Interrupt Status Clear Registers(INTSCL_BLOK_TC_H)

Register	Offset	R/W	Description	Reset Value
INTSCL_BLOK_TC_H	DMA_BA+0x344	R/W	Block Transfer Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.59 Source Transaction Complete Interrupt Status Clear Registers

Register	Offset	R/W	Description	Reset Value
INTSCL_SOUR_TC_L	DMA_BA+0x348	R/W	Source Transaction Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTSCL_SOUR_TC	Source Transaction Complete Interrupt Status Clear Registers

3.7.6.60 Source Transaction Complete Interrupt Status Clear Registers

Register	Offset	R/W	Description	Reset Value
INTSCL_SOUR_TC_H	DMA_BA+0x34c	R/W	Source Transaction Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.61 Destination Transaction Complete Interrupt Status Clear Registers

Register	Offset	R/W	Description	Reset Value
INTSCL_DETR_TC_L	DMA_BA+0x350	R/W	Destination Transaction Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTSCL_DEST_TC	Destination Transaction Complete Interrupt Status Clear Registers

3.7.6.62 Destination Transaction Complete Interrupt Status Clear Registers

Register	Offset	R/W	Description	Reset Value
INTSCL_DETR_TC_H	DMA_BA+0x354	R/W	Destination Transaction Complete Interrupt Status Clear Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.63 Error Interrupt Raw Status Clear Registers Low (INTSCL_ERR_L)

Register	Offset	R/W	Description	Reset Value
INTSCL_ERR_L	DMA_BA+0x358	R/W	Error Interrupt Status Clear Registers	0x0

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	INTSCL_ERR	Error Interrupt Status Clear Registers

3.7.6.64 Error Interrupt Raw Status Clear Registers High (INTSCL_ERR_H)

Register	Offset	R/W	Description	Reset Value
INTSCL_ERR_H	DMA_BA+0x35c	R/W	Error Interrupt Status Clear Registers	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.65 Combined Interrupt Status Register Low(INTS_COMB_L)

Register	Offset	R/W	Description	Reset Value
INTS_COMB_L	DMA_BA+0x360	R	Combined Interrupt Status Register	0x0

Bits	Description	
[31:5]	Reserved	Reserved.
[4]	ERR	OR of the contents of INTS_ERR register
[3]	DSTT	OR of the contents of StatusDst register.
[2]	SRCT	OR of the contents of INTS_SOUR_TC register.
[1]	BLOCK	OR of the contents of INTS_BLOK_TC register.
[0]	TFR	OR of the contents of INTS_DMA_TCregister.

3.7.6.66 Combined Interrupt Status Register High(INTS_COMB_H)

Register	Offset	R/W	Description	Reset Value
INTS_COMB_H	DMA_BA+0x364	R	Combined Interrupt Status Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.67 Source Software Transaction Request Register Low(SW_SOUR_TS_REQ_L)

Register	Offset	R/W	Description	Reset Value
SW_SOUR_TS_REQ_L	DMA_BA+0x368	R/W	Source Software Transaction Request Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	SOUR_REQ_WEN	Source request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	SOUR_REQ	Source request **Note:** A channel sour_req bit is written only if the corresponding channel write enable bit in the sour_req_wen field is asserted on the same AHB write transfer, and if the channel is enabled in the CHANNEL_EN register.

3.7.6.68 Source Software Transaction Request Register High(SW_SOUR_TS_REQ_H)

Register	Offset	R/W	Description	Reset Value
SW_SOUR_TS_REQ_H	DMA_BA+0x36c	R/W	Source Software Transaction Request Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.69 Destination Software Transaction Request Register Low(DEST_TS_REQ_L)

Register	Offset	R/W	Description	Reset Value
DEST_TS_REQ_L	DMA_BA+0x370	R/W	Destination Software Transaction Request Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	DEST_REQ_WEN	Destination request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	DEST_REQ	Destination request **Note**: A channel dest_req bit is written only if the corresponding channel write enable bit in the dest_req_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the CHANNEL_EN register.

3.7.6.70 Destination Software Transaction Request Register High(DEST_TS_REQ_H)

Register	Offset	R/W	Description	Reset Value
DEST_TS_REQ_H	DMA_BA+0x374	R/W	Destination Software Transaction Request Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.71 Single Source Transaction Request Register Low (SIG_SOUR_TS_REQ_L)

Register	Offset	R/W	Description	Reset Value
SIG_SOUR_TS_REQ_L	DMA_BA+0x378	R/W	Destination Software Transaction Request Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	SIG_SOUR_WEN	Single write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	SIG_SOUR_REQ	Source single request **Note**: A channel sig_sour_req bit is written only if the corresponding channel write enable bit in the sig_sour_wen field is asserted on the same AHB write transfer, and if the channel is enabled in the CHANNEL_EN register.

3.7.6.72 Single Source Transaction Request Register High(SIG_SOUR_TS_REQ_H)

Register	Offset	R/W	Description	Reset Value
SIG_SOUR_TS_REQ_H	DMA_BA+0x37c	R/W	Destination Software Transaction Request Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.73 Single Destination Transaction Request Register Low(SIG_DEST_TS_REQ_L)

Register	Offset	R/W	Description	Reset Value
SIG_DEST_TS_REQ_L	DMA_BA+0x380	R/W	Destination Software Transaction Request Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	SIG_DEST_WEN	Destination write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	SIGBST_DEST_REQ	Destination single or burst request **Note**: A channel sigbst_dest_req bit is written only if the corresponding channel write enable bit in the sigbst_dest_req_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the CHANNEL_EN register.

3.7.6.74 Single Destination Transaction Request Register High(SIG_DEST_TS_REQ_H)

Register	Offset	R/W	Description	Reset Value
SIG_DEST_TS_REQ_H	DMA_BA+0x384	R/W	Destination Software Transaction Request Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.75 Last Source Transaction Request Register Low(LST_SOUR_TS_REQ_L)

Register	Offset	R/W	Description	Reset Value
LST_SOUR_TS_REQ_L	DMA_BA+0x388	R/W	Last Source Transaction Request Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	LST_SOUR_WEN	Source last transaction request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	LST_SOUR_REQ	Source last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

3.7.6.76 Last Source Transaction Request Register High(LST_SOUR_TS_REQ_H)

Register	Offset	R/W	Description	Reset Value
LST_SOUR_TS_REQ_H	DMA_BA+0x38c	R/W	Last Source Transaction Request Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.77 Last Destination Transaction Request Register Low(LST_DEST_TS_REQ_L)

Register	Offset	R/W	Description	Reset Value
LST_DEST_TS_REQ_L	DMA_BA+0x390	R/W	Last Destination Transaction Request Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10:8]	LST_DEST_WEN	Destination last transaction request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	LST_DEST_REQ	Destination last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

3.7.6.78 Last Destination Transaction Request Register High(LST_DEST_TS_REQ_H)

Register	Offset	R/W	Description	Reset Value
LST_DEST_TS_REQ_H	DMA_BA+0x394	R/W	Last Destination Transaction Request Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.79 DMA Configuration Register Low (DMA_CFG_L)

Register	Offset	R/W	Description	Reset Value
DMA_CFG_L	DMA_BA+0x398	R	DMA Configuration Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	DMA_EN	DMA Enable bit. 0 = DMA Disabled 1 = DMA Enabled.

3.7.6.80 DMA Configuration Register High (DMA_CFG_H)

Register	Offset	R/W	Description	Reset Value
DMA_CFG_H	DMA_BA+0x39c	R	DMA Configuration Register	0x0000_0000

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.81 DMA Channel Enable Register Low(CHANNEL_EN_L)

Register	Offset	R/W	Description	Reset Value
CHANNEL_EN_L	DMA_BA+0x3A0	R/W	DMA Channel Enable Register	0x0

Bits	Description	
[31:11]	Reserved	Reserved.
[10 :8]	CH_WRITE_EN	Channel enable write enable(only writable)
[7:3]	Reserved	Reserved
[2:0]	CHNL_EN	Enables/Disables the channel. Setting this bit enables a channel; clearing this bit disables the channel. 0 = Disable the Channel 1 = Enable the Channel The CHANNEL_EN. chnl_en bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. **Note**: All bits of this register are cleared to 0 when the global DMA channel enable bit, DMA_CFG[0], is 0.

3.7.6.82 DMA Channel Enable Register High(CHANNEL_EN_H)

Register	Offset	R/W	Description	Reset Value
CHANNEL_EN_H	DMA_BA+0x3A4	R/W	DMA Channel Enable Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.83 DMA ID Register Low(DMA_ID_REG_L)

Register	Offset	R/W	Description	Reset Value
DMA_ID_REG_L	DMA_BA+0x3A8	R	DMA ID Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved

3.7.6.84 DMA ID Register High(DMA_ID_REG_H)

Register	Offset	R/W	Description	Reset Value
DMA_ID_REG_H	DMA_BA+0x3ac	R	DMA ID Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved

3.7.6.85 DMA Test Register Low(DMA_TEST_L)

Register	Offset	R/W	Description	Reset Value
DMA_TEST_L	DMA_BA+0x3B0	R/W	DMA Test Register	0x0

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	TEST_SL_INFACE	Puts the AHB slave interface into test mode. In this mode, the readback value of the writable registers always matches the value written. This bit does not allow writing to read-only registers. 0 = Normal mode 1 = Test mode

3.7.6.86 DMA Test Register High(DMA_TEST_H)

Register	Offset	R/W	Description	Reset Value
DMA_TEST_H	DMA_BA+0x3B4	R/W	DMA Test Register	0x0

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.87 DMA Component Parameters Register 3 Low(DMA_CMP_PARMS_3_L)

Register	Offset	R/W	Description	Reset Value
DMA_CMP_PARMS_3_L	DMA_BA+0x3E0	R	DMA Component Parameters Register 3	0x0

Bits	Description	
[31]	Reserved	Reserved
[30:28]	CH2_FIFO_DEPTH	The value of this register is derived from the DMAH_ch2_fifo_depth core Consultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128 Default value: 0x1
[27:19]	Reserved	Reserved
[18:16]	CH2_MAX_MULT_SIZE	The value of this register is derived from the DMAH_CH2_MULT_SIZE core Consultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x1
[15:14]	CH2_FC	The value of this register is derived from the DMAH_ch2_fc core Consultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY Default value: 0x0
[13:12]	Reserved	Reserved
[11]	CH2_MULT_BLK_EN	The value of this register is derived from the DMAH_ch2_mult_blk_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[10]	CH2_FC	The value of this register is derived from the DMAH_ch2_fc core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[9]	CH2_SOUR_GAT_EN	The value of this register is derived from the DMAH_ch2_sour_gat_en core Consultant parameter.

		0x0 = FALSE 0x1 = TRUE Default value: 0x0
[8]	CH2_DEST_SCA_EN	The value of this register is derived from the DMAH_ch2_dest_sca_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[7:6]	Reserved	Reserved
[5:3]	CH2_STW	The value of this register is derived from the DMAH_ch2_stw core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0
[2:0]	CH2_DTW	The value of this register is derived from the DMAH_ch2_dtw core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0

3.7.6.88 DMA Component Parameters Register 3 High(DMA_CMP_PARMS_3_H)

Register	Offset	R/W	Description	Reset Value
DMA_CMP_PARMS_3_H	DMA_BA+0x3E4	R	DMA Component Parameters Register 3	0x0

Bits	Description	
[31]	Reserved	Reserved.
[30:28]	CH1_FIFO_DEPTH	The value of this register is derived from the DMAH_ch1_fifo_depth core Consultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 2x3 = 64 0x4 = 128 Default value: 0x1
[27:19]	Reserved	Reserved.
[18:16]	CH1_MAX_MULT_SIZE	The value of this register is derived from the DMAH_CH1_MULT_SIZE core Consultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x1
[15:14]	CH1_FC	The value of this register is derived from the DMAH_ch1_fc core Consultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY Default value: 0x0
[13:12]	Reserved	Reserved
[11]	CH1_MULT_BLK_EN	The value of this register is derived from the DMAH_ch1_mult_blk_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[10]	CH1_LOCK_EN	The value of this register is derived from the DMAH_ch1_lock_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[9]	CH1_SOUR_GAT_EN	The value of this register is derived from the DMAH_ch1_sour_gat_en core Consultant parameter.

		0x0 = FALSE 0x1 = TRUE Default value: 0x0
[8]	CH1_DEST_SCA_EN	The value of this register is derived from the DMAH_ch1_dest_sca_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[7:6]	Reserved	Reserved
[5:3]	CH1_STW	The value of this register is derived from the DMAH_ch1_stw core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0
[2:0]	CH1_DTW	The value of this register is derived from the DMAH_ch1_dtw core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0

3.7.6.89 DMA Component Parameters Register 2 Low(DMA_CMP_PARMS_2_L)

Register	Offset	R/W	Description	Reset Value
DMA_CMP_PARMS_2_L	DMA_BA+0x3E8	R	DMA Component Parameters Register 2	0x0

Bits	Description	
[31]	Reserved	Reserved
[30:28]	CH0_FIFO_DEPTH	The value of this register is derived from the dma_ch0_fifo_depth core Consultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 0x3 = 64 0x4 = 128 Default value: 0x1
[27:19]	Reserved	Reserved
[18:16]	CH0_MAX_MULT_SIZE	The value of this register is derived from the dma_ch0_max_mult_size core Consultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x1
[15:14]	CH0_FC	The value of this register is derived from the dma_ch0_fc core Consultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY Default value: 0x0
[13]	CH0_HC_LLP	The value of this register is derived from the dma_ch0_hc_llp core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[12]	CH0_CTL_WB_EN	The value of this register is derived from the dmah_ch0_ctl_wb_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[11]	CH0_MULT_BLK_EN	The value of this register is derived from the dmah_ch0_mult_blk_en core Consultant parameter. 0x0 = FALSE

		0x1 = TRUE Default value: 0x0
[10]	CH0_LOCK_EN	The value of this register is derived from the dmah_ch0_lock_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[9]	CH0_SOUR_GAT_EN	The value of this register is derived from the dmah_ch0_sour_gat_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[8]	CH0_DEST_SCA_EN	The value of this register is derived from the dmah_ch0_dest_sca_en core Consultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[7:6]	Reserved	Reserved
[5:3]	CH0_STW	The value of this register is derived from the dmah_ch0_stw core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0
[2:0]	CH0_DTW	The value of this register is derived from the dmah_ch0_dtw core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0

3.7.6.90 DMA Component Parameters Register 2 High(DMA_CMP_PARMS_2_H)

Register	Offset	R/W	Description	Reset Value
DMA_CMP_PARMS_2_H	DMA_BA+0x3EC	R	DMA Component Parameters Register 2	0x0

Bits	Description	
[31:12]	Reserved	Reserved
[11:8]	CH2_MUT_BK_TYPE	The values of these bit fields are derived from the DMAH_CHx_MULTI_BLK_TYPE core Consultant parameter. 0x0 = NO_HARDCODE 0x1 = CONT_RELOAD 0x2 = RELOAD_CONT 0x3 = RELOAD_RELOAD 0x4 = CONT_LLP 0x5 = RELOAD_LLP 0x6 = LLP_CONT 0x7 = LLP_RELOAD 0x8 = LLP_LLP Default value: 0x0
[7:4]	CH1_MUT_BK_TYPE	
[3:0]	CH0_MUT_BK_TYPE	

3.7.6.91 DMA Component Parameters Register 1 Low(DMA_CMP_PARMS_1_L)

Register	Offset	R/W	Description	Reset Value
DMA_CMP_PARMS_1_L	DMA_BA+0x3F0	R	DMA Component Parameters Register 1	0x0

Bits	Description	
[31:12]	Reserved	Reserved
[11:8]	CH2_BLK_MAX_SIZE	The values of these bit fields are derived from the DMAH_chx_blk_max_size core Consultant parameter. 0x0 = 3 0x1 = 7 0x2 =15 0x3 = 31 0x4 = 63 0x5 = 127 0x6 = 255 0x7 = 511 0x8 = 1023 0x9 = 2047 0xa = 4095 Default value: 0x8
[7:4]	CH1_BLK_MAX_SIZE	
[3:0]	CH0_BLK_MAX_SIZE	

3.7.6.92 DMA Component Parameters Register 1 High(DMA_CMP_PARMS_1_H)

Register	Offset	R/W	Description	Reset Value
DMA_CMP_PARMS_1_H	DMA_BA+0x3F4	R	DMA Component Parameters Register 1	0x0

Bits	Description	
[31:30]	Reserved	Reserved
[29]	STATIC_ENDLAN_SEL	The value of this register is derived from the DMAH_static_endlan_sel core Consultant parameter. 0 = FALSE 1 = TRUE Default value: 0x1
[28]	ADD_ENDCODE_PARA	The value of this register is derived from the DMAH_add_endcode_para core Consultant parameter. 0 = FALSE 1 = TRUE Default value: 0x0
[27:23]	NUM_HS_INT	The value of this register is derived from the DMAH_num_hs_int core Consultant parameter. 0x00 = 0 to 0x10 = 16 Default value: 0x10
[22:21]	HDATA_M1_WIDTH	The value of this register is derived from the DMAH_hdata_m1_width core Consultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits Default value: 0x0
[20:15]	Reserved	Reserved
[14:13]	HDATA_S_WIDTH	The value of this register is derived from the DMAH_hdata_s_width core Consultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits Default value: 0x0
[12:11]	MAST_NUM_INT	The value of this register is derived from the DMAH_mast_num_int core Consultant parameter. 0x0 = 1 to 0x3 = 4 Default value: 0x0
[10:8]	CHAN_NUM	The value of this register is derived from the DMAH_chan_num core Consultant parameter. 0x0 = 1

		to 0x7 = 8 Default value: 0x2
[7:4]	Reserved	Reserved
[3]	MAB_RST	The value of this register is derived from the DMAH_mab_rst core Consultant parameter. 0 = FALSE 1 = TRUE Default value: 0x0
[2:1]	IO_INTR	The value of this register is derived from the DMAH_io_intr core Consultant parameter. 0x0 = ALL 0x1 = TYPE 0x2 = COMBINED 0x3 = reserved Default value: 0x2
[0]	ENDLAN_BIG	The value of this register is derived from the DMAH_endlan_big core Consultant parameter. 0 = FALSE 1 = TRUE Default value: 0x0

3.7.6.93 DMA Component ID Register Low(DMA_COM_ID_L)

Register	Offset	R/W	Description	Reset Value
DMA_COM_ID_L	DMA_BA+0x3F8	R	DMA Component ID Register	0x0000_0000

Bits	Description	
[31:0]	Reserved	Reserved.

3.7.6.94 DMA Component ID Register High(DMA_COM_ID_H)

Register	Offset	R/W	Description	Reset Value
DMA_COM_ID_H	DMA_BA+0x3FC	R	DMA Component ID Register	0x0000_0000

Bits	Description	
[31:0]	Reserved	Reserved.

3.8 General Purpose I/O (GPIO)

3.8.1 Overview

The PAN101x series has up to 21 General Purpose I/O pins to be shared with other function pins de-pending on the chip configuration. These 21 pins are arranged in 4 ports named as P0, P1, P2, P3. Each of the 21 pins is independent and has the corresponding register bits to control the pin mode function and data. Different package formats correspond to different GPIO numbers.

The I/O type of each pin can be configured by software individually as Input, Push-pull output, Open-drain output, or Quasi-bidirectional mode. After the chip is reset, the I/O mode of all pins is stay in input mode and each port data register Px_DOUT[n] resets to 1. For Quasi-bidirectional mode, each I/O pin is equipped with a very weak individual pull-up resistor about 110k~ 300kΩ for VDD is from 1.8 V to 3.6 V.

3.8.2 Features

- Four I/O modes:
  - Quasi-bidirectional mode
  - Push-pull output
  - Open-drain output
  - Input-only with high impendence
- Quasi-bidirectional TTL/Schmitt trigger input mode selected by SYS_Px_MFP[23:16]
- I/O pin configured as interrupt source with edge/level setting
- I/O pin internal pull-up resistor enabled only in Quasi-bidirectional I/O mode
- Enabling the pin interrupt function will also enable the pin wake-up function
- High driver and high sink I/O mode suppor

3.8.3 Block Diagram

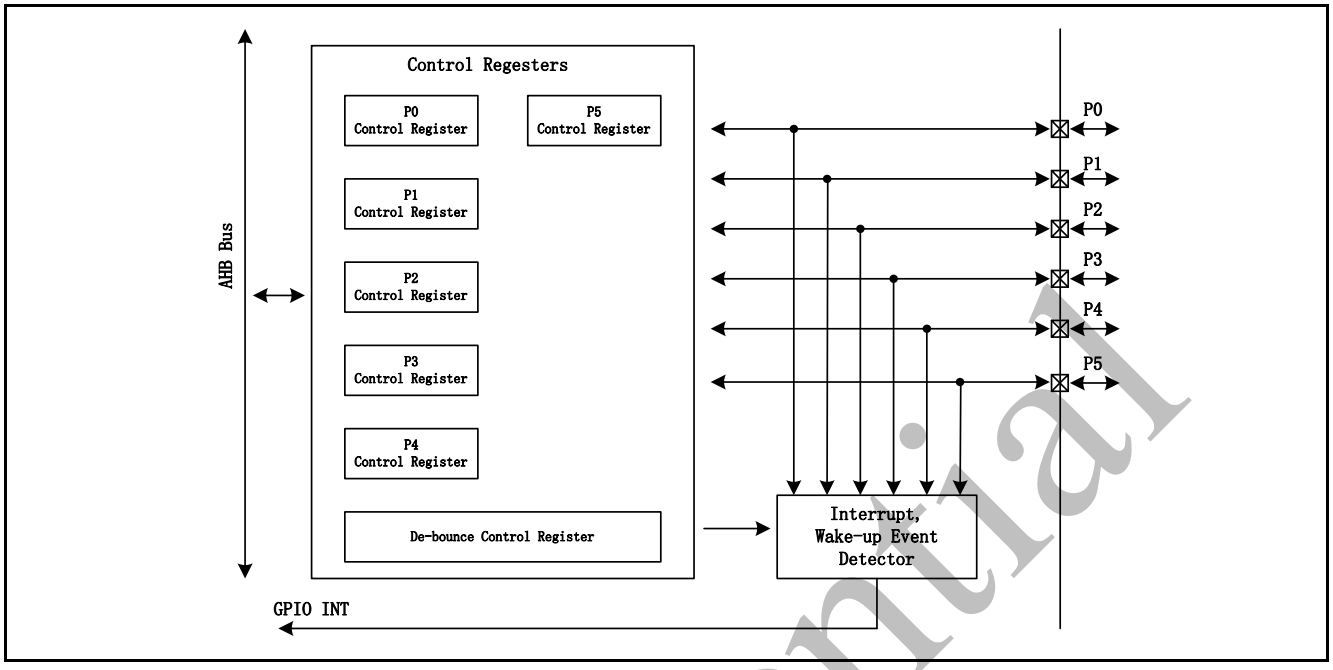


Figure 3-27 GPIO Controller Block Diagram

3.8.4 Basic Configuration

The GPIO pin functions are configured in *SYS_P0_MFP*, *SYS_P1_MFP*, *SYS_P2_MFP*, *SYS_P3_MFP* registers.

3.8.5 Functional Description

The PAD diagram is introduced in Figure 3-28.

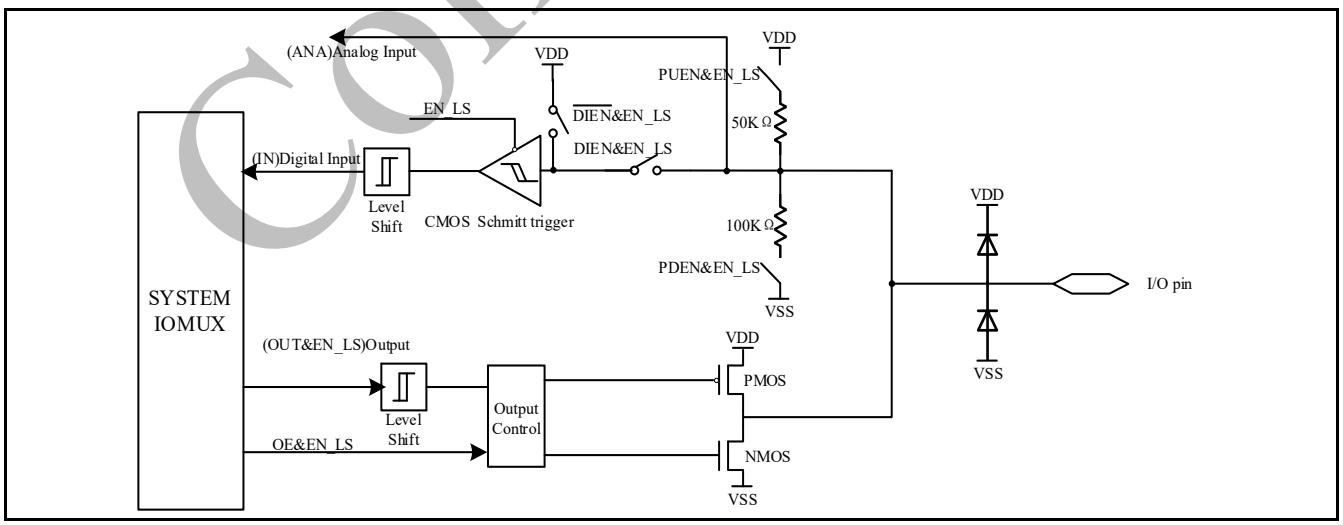


Figure 3-28 PAD Diagram

3.8.5.1 Input Mode

Set *MODEn* (Px_MODE[2n+1:2n]) to 00 as the Px.n pin is in Input mode and the I/O pin is in tri-state (high impedance) without output drive capability. The *PIN* (Px_PIN[n]) value reflects the status of the corresponding port pins.

3.8.5.2 Push-pull Output Mode

Set *MODEn* (Px_MODE[2n+1:2n]) to 01 as Px.n pin is in Push-pull Output mode and the I/O pin supports digital output function with source/sink current capability. The bit value in the corresponding *DOUT* (Px_DOUT[n]) is driven on the pin.

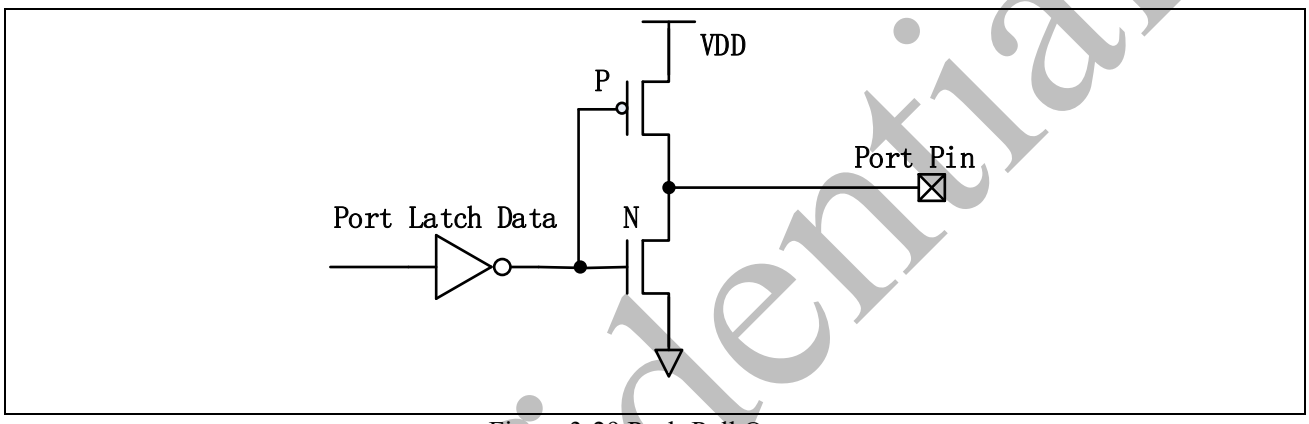


Figure 3-29 Push-Pull Output

3.8.5.3 Open-drain Output Mode

Set *MODEn* (Px_MODE[2n+1:2n]) to 10 as Px.n pin is in Open-drain mode and the digital output function of I/O pin supports only sink current capability, an external pull-up register is needed for driving high state. If the bit value in the corresponding *DOUT* (Px_DOUT[n]) bit is 0, the pin drive a low output on the pin. If the bit value in the corresponding *DOUT* (Px_DOUT[n]) bit is 1, the pin output drives high that is controlled by external pull high resistor.

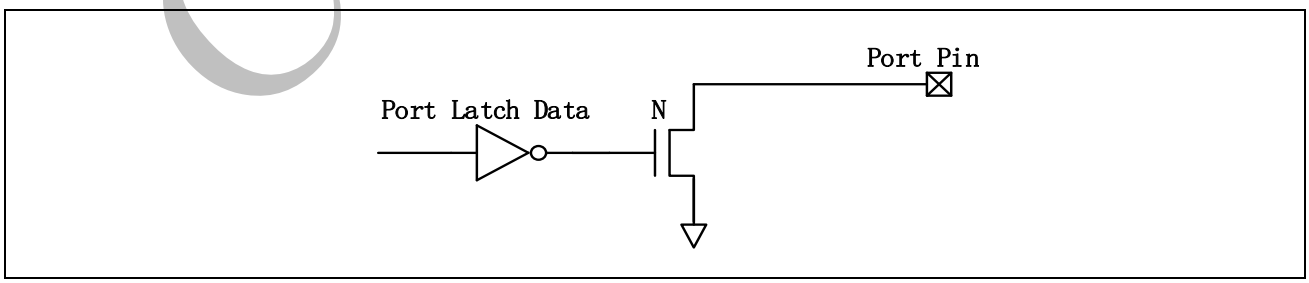


Figure 3-30 Open-Drain Output

3.8.5.4 Quasi-bidirectional Mode

Set *MODEn* (Px_MODE[2n+1:2n]) to 11 as the Px.n pin is in Quasi-bidirectional mode and the I/O pin supports digital output and input function at the same time but the source current is

only up to hundreds uA. Before the digital input function is performed the corresponding *DOUT* (Px_DOUT[n]) bit must be set to 1. If the bit value in the corresponding *DOUT* (Px_DOUT[n]) bit is 0, the pin drive a low output on the pin. If the bit value in the corresponding *DOUT* (Px_DOUT[n]) bit is 1, the pin status is controlled by internal pull-up resistor.

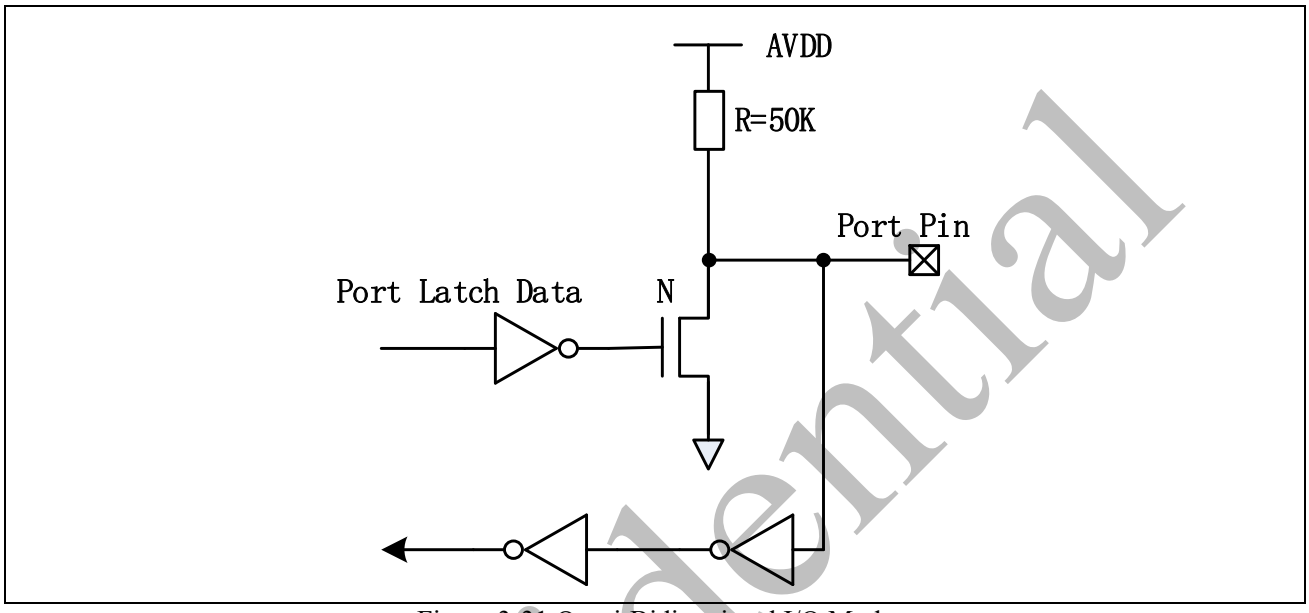


Figure 3-31 Quasi-Bidirectional I/O Mode

3.8.6 GPIO Interrupt and Wake-up Function

Each GPIO pin can be set as chip interrupt source by setting correlative RHIEN (*Px_INTEN*[n+16])/ FLIEN (*Px_INTEN*[n]) bit and TYPE (*Px_INTTYPE*[n]). There are five types of interrupt condition can be selected: low level trigger, high level trigger, falling edge trigger, rising edge trigger and both rising and falling edge trigger. For edge trigger condition, user can enable input signal de-bounce function to prevent unexpected interrupt happened which caused by noise. The de-bounce clock source and sampling cycle period can be set through DBCLKSRC (*GPIO_DBCTL*[4]) and DBCLKSEL (*GPIO_DBCTL*[3:0]) register.

The GPIO can also be the chip wake-up source when chip enters Idle/Power-down mode. The setting of wake-up trigger condition is the same as GPIO interrupt trigger.

3.8.7 GPIO Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
GPIO Base Address: GP_BA = 0x4002_0000				
P0_MODE	GP_BA+0x000	R/W	P0 I/O Mode Control	0x0000_0000
P0_DINOFF	GP_BA+0x004	R/W	P0 Digital Input Path Disable Control	0x00FC_0003
P0_DOUT	GP_BA+0x008	R/W	P0 Data Output Value	0x0000_00FF
P0_DATMSK	GP_BA+0x00C	R/W	P0 Data Output Write Mask	0x0000_0000
P0_PIN	GP_BA+0x010	R	P0 Pin Value	0x0000_00XX
P0_DBEN	GP_BA+0x014	R/W	P0 De-bounce Enable Control	0x0000_0000
P0_INTTYPE	GP_BA+0x018	R/W	P0 Interrupt Mode Control	0x0000_0000
P0_INTEN	GP_BA+0x01C	R/W	P0 Interrupt Enable Control	0x0000_0000
P0_INTSRC	GP_BA+0x020	R/W	P0 Interrupt Source Flag	0x0000_0000
P1_MODE	GP_BA+0x040	R/W	P1 I/O Mode Control	0x0000_0000
P1_DINOFF	GP_BA+0x044	R/W	P1 Digital Input Path Disable Control	0x00FF_0000
P1_DOUT	GP_BA+0x048	R/W	P1 Data Output Value	0x0000_00FF
P1_DATMSK	GP_BA+0x04C	R/W	P1 Data Output Write Mask	0x0000_0000
P1_PIN	GP_BA+0x050	R	P1 Pin Value	0x0000_00XX
P1_DBEN	GP_BA+0x054	R/W	P1 De-bounce Enable Control	0x0000_0000
P1_INTTYPE	GP_BA+0x058	R/W	P1 Interrupt Mode Control	0x0000_0000
P1_INTEN	GP_BA+0x05C	R/W	P1 Interrupt Enable Control	0x0000_0000
P1_INTSRC	GP_BA+0x060	R/W	P1 Interrupt Source Flag	0x0000_0000
P2_MODE	GP_BA+0x080	R/W	P2 I/O Mode Control	0x0000_0000
P2_DINOFF	GP_BA+0x084	R/W	P2 Digital Input Path Disable Control	0x00FF_0000
P2_DOUT	GP_BA+0x088	R/W	P2 Data Output Value	0x0000_00FF
P2_DATMSK	GP_BA+0x08C	R/W	P2 Data Output Write Mask	0x0000_0000
P2_PIN	GP_BA+0x090	R	P2 Pin Value	0x0000_00XX
P2_DBEN	GP_BA+0x094	R/W	P2 De-bounce Enable Control	0x0000_0000
P2_INTTYPE	GP_BA+0x098	R/W	P2 Interrupt Mode Control	0x0000_0000
P2_INTEN	GP_BA+0x09C	R/W	P2 Interrupt Enable Control	0x0000_0000
P2_INTSRC	GP_BA+0x0A0	R/W	P2 Interrupt Source Flag	0x0000_0000
P3_MODE	GP_BA+0x0C0	R/W	P3 I/O Mode Control	0x0000_0000
P3_DINOFF	GP_BA+0x0C4	R/W	P3 Digital Input Path Disable Control	0x00FF_0000
P3_DOUT	GP_BA+0x0C8	R/W	P3 Data Output Value	0x0000_00FF
P3_DATMSK	GP_BA+0x0CC	R/W	P3 Data Output Write Mask	0x0000_0000
P3_PIN	GP_BA+0x0D0	R	P3 Pin Value	0x0000_00XX
P3_DBEN	GP_BA+0x0D4	R/W	P3 De-bounce Enable Control	0x0000_0000
P3_INTTYPE	GP_BA+0x0D8	R/W	P3 Interrupt Mode Control	0x0000_0000
P3_INTEN	GP_BA+0x0DC	R/W	P3 Interrupt Enable Control	0x0000_0000
P3_INTSRC	GP_BA+0x0E0	R/W	P3 Interrupt Source Flag	0x0000_0000
GPIO_DBCTL	GP_BA+0x180	R/W	De-bounce Cycle Control	0x0000_3f00
P00_PDIO	GP_BA+0x200	R/W	GPIO P0.0 Pin Data Input/Output	0x0000_0001
P01_PDIO	GP_BA+0x204	R/W	GPIO P0.1 Pin Data Input/Output	0x0000_0001
P02_PDIO	GP_BA+0x208	R/W	Reserved	0x0000_0001

P03_PDIO	GP_BA+0x20C	R/W	Reserved	0x0000_0001
P04_PDIO	GP_BA+0x210	R/W	GPIO P0.4 Pin Data Input/Output	0x0000_0001
P05_PDIO	GP_BA+0x214	R/W	Reserved	0x0000_0001
P06_PDIO	GP_BA+0x218	R/W	Reserved	0x0000_0001
P07_PDIO	GP_BA+0x21C	R/W	Reserved	0x0000_0001
P10_PDIO	GP_BA+0x220	R/W	GPIO P1.0 Pin Data Input/Output	0x0000_0001
P11_PDIO	GP_BA+0x224	R/W	GPIO P1.1 Pin Data Input/Output	0x0000_0001
P12_PDIO	GP_BA+0x228	R/W	GPIO P1.2 Pin Data Input/Output	0x0000_0001
P13_PDIO	GP_BA+0x22C	R/W	GPIO P1.3 Pin Data Input/Output	0x0000_0001
P14_PDIO	GP_BA+0x230	R/W	GPIO P1.4 Pin Data Input/Output	0x0000_0001
P15_PDIO	GP_BA+0x234	R/W	GPIO P1.5 Pin Data Input/Output	0x0000_0001
P16_PDIO	GP_BA+0x238	R/W	Reserved	0x0000_0001
P17_PDIO	GP_BA+0x23C	R/W	Reserved	0x0000_0001
P20_PDIO	GP_BA+0x240	R/W	Reserved	0x0000_0001
P21_PDIO	GP_BA+0x244	R/W	GPIO P2.1 Pin Data Input/Output	0x0000_0001
P22_PDIO	GP_BA+0x248	R/W	GPIO P2.2 Pin Data Input/Output	0x0000_0001
P23_PDIO	GP_BA+0x24C	R/W	GPIO P2.3 Pin Data Input/Output	0x0000_0001
P24_PDIO	GP_BA+0x250	R/W	GPIO P2.4 Pin Data Input/Output	0x0000_0001
P25_PDIO	GP_BA+0x254	R/W	GPIO P2.5 Pin Data Input/Output	0x0000_0001
P26_PDIO	GP_BA+0x258	R/W	GPIO P2.6 Pin Data Input/Output	0x0000_0001
P27_PDIO	GP_BA+0x24C	R/W	Reserved	0x0000_0001
P30_PDIO	GP_BA+0x260	R/W	Reserved	0x0000_0001
P31_PDIO	GP_BA+0x264	R/W	GPIO P3.1 Pin Data Input/Output	0x0000_0001

3.8.8 GPIO Register Description

3.8.8.1 Port 0-5 I/O Mode Control (Px_MODE)

Register	Offset	R/W	Description	Reset Value
P0_MODE	GP_BA+0x000	R/W	P0 I/O Mode Control	0x0000_0000
P1_MODE	GP_BA+0x040	R/W	P1 I/O Mode Control	0x0000_0000
P2_MODE	GP_BA+0x080	R/W	P2 I/O Mode Control	0x0000_0000
P3_MODE	GP_BA+0x0C0	R/W	P3 I/O Mode Control	0x0000_0000

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[2n+1:2n] n=0,1..7	MODEn	Port 0-5 I/O Pin[N] Mode Control Determine each I/O mode of Px.n pins. 00 = Px.n is in Input mode. 01 = Px.n is in Push-pull Output mode. 10 = Px.n is in Open-drain Output mode. 11 = Px.n is in Quasi-bidirectional mode. **Note:** Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3, Max. n=7 for port 4 Max. n=7 for port 5

3.8.8.2 Port 0-5 Digital Input Path Disable Control (Px_DINOFF)

Register	Offset	R/W	Description	Reset Value
P0_DINOFF	GP_BA+0x004	R/W	P0 Digital Input Path Disable Control	0x00FC_000C
P1_DINOFF	GP_BA+0x044	R/W	P1 Digital Input Path Disable Control	0x00FF_0000
P2_DINOFF	GP_BA+0x084	R/W	P2 Digital Input Path Disable Control	0x00FF_0000
P3_DINOFF	GP_BA+0x0C4	R/W	P3 Digital Input Path Disable Control	0x00FF_0000

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[n+16] n=0,1..7	DINOFF[n]	Port 0-5 Pin[N] Digital Input Path Disable Control Each of these bits is used to control if the digital input path of corresponding Px.n pin is disabled. If input is analog signal, users can disable Px.n digital input path to avoid input current leakage. 0 = Px.n digital input path Enabled. 1 = Px.n digital input path Disabled (digital input tied to low). **Note1:** Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5 **Note2**: The DINOFF of P4[7:6], P5[6] need 3V sync, but not support auto_3V sync
[15:8]	PDEN[n]	Port 0-5 Pin[N] Digital Pull Down Path Enable Control Each of these bits is used to control if the digital pull down path of corresponding Px.n pin is enabled. 0 = Px.n digital pull down path Disabled. 1 = Px.n digital pull down path Enabled. **Note:** The PDEN of P4[7:6], P5[6] need 3V sync, but not support auto_3V sync
[n] n=0,1..7	PUEN[n]	Port 0-5 Pin[N] Digital Pull Up Path Enable Control Each of these bits is used to control if the digital pull up path of corresponding Px.n pin is enabled. 0 = Px.n digital pull up path Disabled. 1 = Px.n digital pull up path Enabled. **Note:** The PDEN of P4[7:6], P5[6] need 3V sync, but not support auto_3V sync

3.8.8.3 Port 0-5 Data Output Value (Px_DOUT)

Register	Offset	R/W	Description	Reset Value
P0_DOUT	GP_BA+0x008	R/W	P0 Data Output Value	0x0000_00FF
P1_DOUT	GP_BA+0x048	R/W	P1 Data Output Value	0x0000_00FF
P2_DOUT	GP_BA+0x088	R/W	P2 Data Output Value	0x0000_00FF
P3_DOUT	GP_BA+0x0C8	R/W	P3 Data Output Value	0x0000_00FF

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[n] n=0,1..7	DOUT[n]	Port 0-5 Pin[N] Output Value Each of these bits controls the status of a Px.n pin when the Px.n is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 0 = Px.n will drive Low if the Px.n pin is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 1 = Px.n will drive High if the Px.n pin is configured as Push-pull output or Quasi-bidirectional mode. **Note**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.8.8.4 Port 0-5 Data Output Write Mask (Px_DATMSK)

Register	Offset	R/W	Description	Reset Value
P0_DATMSK	GP_BA+0x00C	R/W	P0 Data Output Write Mask	0x0000_0000
P1_DATMSK	GP_BA+0x04C	R/W	P1 Data Output Write Mask	0x0000_0000
P2_DATMSK	GP_BA+0x08C	R/W	P2 Data Output Write Mask	0x0000_0000
P3_DATMSK	GP_BA+0x0CC	R/W	P3 Data Output Write Mask	0x0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[n] n=0,1..7	DATMSK[n]	Port 0-5 Pin[N] Data Output Write Mask These bits are used to protect the corresponding DOUT (Px_DOUT[n]) bit. When the DATMSK (Px_DATMSK[n]) bit is set to 1, the corresponding DOUT (Px_DOUT[n]) bit is protected. If the write signal is masked, writing data to the protect bit is ignore. 0 = Corresponding DOUT (Px_DOUT[n]) bit can be updated. 1 = Corresponding DOUT (Px_DOUT[n]) bit protected. **Note1**: This function only protects the corresponding DOUT (Px_DOUT[n]) bit, and will not protect the corresponding PDIO (Pxn_PDIO[0]) bit. **Note2**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.8.8.5 Port 0-5 Pin Value (Px_PIN)

Register	Offset	R/W	Description	Reset Value
P0_PIN	GP_BA+0x010	R	P0 Pin Value	0x0000_00XX
P1_PIN	GP_BA+0x050	R	P1 Pin Value	0x0000_00XX
P2_PIN	GP_BA+0x090	R	P2 Pin Value	0x0000_00XX
P3_PIN	GP_BA+0x0D0	R	P3 Pin Value	0x0000_00XX

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[n] n=0,1..7	PIN[n]	Port 0-5 Pin[N] Pin Value Each bit of the register reflects the actual status of the respective Px.n pin. If the bit is 1, it indicates the corresponding pin status is high; else the pin status is low. **Note:** Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.8.8.6 Port 0-5 De-bounce Enable Control (Px_DBEN)

Register	Offset	R/W	Description	Reset Value
P0_DBEN	GP_BA+0x014	R/W	P0 De-bounce Enable Control	0x0000_0000
P1_DBEN	GP_BA+0x054	R/W	P1 De-bounce Enable Control	0x0000_0000
P2_DBEN	GP_BA+0x094	R/W	P2 De-bounce Enable Control	0x0000_0000
P3_DBEN	GP_BA+0x0D4	R/W	P3 De-bounce Enable Control	0x0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[n] n=0,1..7	DBEN[n]	Port 0-5 Pin[N] Input Signal De-bounce Enable Bit The DBEN[n] bit is used to enable the de-bounce function for each corresponding bit. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the interrupt. The de-bounce clock source is controlled by DBCLKSRC (GPIO_DBCTL [4]), one de-bounce sample cycle period is controlled by DBCLKSEL (GPIO_DBCTL [3:0]). 0 = Px.n de-bounce function Disabled. 1 = Px.n de-bounce function Enabled. The de-bounce function is valid only for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ignore. **Note1:** If Px.n pin is chosen as Power-down wake-up source, user should be disable the de-bounce function before entering Power-down mode to avoid the second interrupt event occurred after system waken up which caused by Px.n de-bounce function. **Note2:** Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.8.8.7 Port 0-5 Interrupt Mode Control (Px_ INTTYPE)

Register	Offset	R/W	Description	Reset Value
P0_INTTYPE	GP_BA+0x018	R/W	P0 Interrupt Mode Control	0x0000_0000
P1_INTTYPE	GP_BA+0x058	R/W	P1 Interrupt Mode Control	0x0000_0000
P2_INTTYPE	GP_BA+0x098	R/W	P2 Interrupt Mode Control	0x0000_0000
P3_INTTYPE	GP_BA+0x0D8	R/W	P3 Interrupt Mode Control	0x0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[n] n=0,1..7	TYPE[n]	Port 0-5 Pin[N] Edge Or Level Detection Interrupt Trigger Type Control TYPE (Px_INTTYPE[n]) bit is used to control the triggered interrupt is by level trigger or by edge trigger. If the interrupt is by edge trigger, the trigger source can be controlled by de-bounce. If the interrupt is by level trigger, the input source is sampled by one AHB_CLK clock and generates the interrupt. 0 = Edge trigger interrupt. 1 = Level trigger interrupt. If the pin is set as the level trigger interrupt, only one level can be set on the registers RHIEN (Px_INTEN[n+16])/FLIEN (Px_INTEN[n]). If both levels to trigger interrupt are set, the setting is ignored and no interrupt will occur. The de-bounce function is valid only for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ignore. **Note**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.8.8.8 Port 0-5 Interrupt Enable Control (Px_INTEN)

Register	Offset	R/W	Description	Reset Value
P0_INTEN	GP_BA+0x01C	R/W	P0 Interrupt Enable Control	0x0000_0000
P1_INTEN	GP_BA+0x05C	R/W	P1 Interrupt Enable Control	0x0000_0000
P2_INTEN	GP_BA+0x09C	R/W	P2 Interrupt Enable Control	0x0000_0000
P3_INTEN	GP_BA+0x0DC	R/W	P3 Interrupt Enable Control	0x0000_0000

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[n+16] n=0,1..7	RHIEN[n]	Port 0-5 Pin[N] Rising Edge Or High Level Interrupt Trigger Type Enable Bit The RHIEN (Px_INTEN[n+16]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function. When setting the RHIEN (Px_INTEN[n+16]) bit to 1: If the interrupt is level trigger (TYPE (Px_INTTYPE[n]) bit is set to 1), the input Px.n pin will generate the interrupt while this pin state is at high level. If the interrupt is edge trigger (TYPE (Px_INTTYPE[n]) bit is set to 0), the input Px.n pin will generate the interrupt while this pin state changed from low to high. 0 = Px.n level high or low to high interrupt Disabled. 1 = Px.n level high or low to high interrupt Enabled. **Note**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5
[15:8]	Reserved	Reserved.
[n] n=0,1..7	FLIEN[n]	Port 0-5 Pin[N] Falling Edge Or Low Level Interrupt Trigger Type Enable Bit The FLIEN (Px_INTEN[n]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function. When setting the FLIEN (Px_INTEN[n]) bit to 1: If the interrupt is level trigger (TYPE (Px_INTTYPE[n]) bit is set to 1), the input Px.n pin will generate the interrupt while this pin state is at low level. If the interrupt is edge trigger (TYPE (Px_INTTYPE[n]) bit is set to 0), the input Px.n pin will generate the interrupt while this pin state changed from high to low. 0 = Px.n level low or high to low interrupt Disabled. 1 = Px.n level low or high to low interrupt Enabled. **Note**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.8.8.9 Port 0-5 Interrupt Source Flag (Px_INTSRC)

Register	Offset	R/W	Description	Reset Value
P0_INTSRC	GP_BA+0x020	R/W	P0 Interrupt Source Flag	0x0000_0000
P1_INTSRC	GP_BA+0x060	R/W	P1 Interrupt Source Flag	0x0000_0000
P2_INTSRC	GP_BA+0x0A0	R/W	P2 Interrupt Source Flag	0x0000_0000
P3_INTSRC	GP_BA+0x0E0	R/W	P3 Interrupt Source Flag	0x0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[n] n=0,1..7	INTSRC[n]	Port 0-5 Pin[N] Interrupt Source Flag Write Operation: 0 = No action. 1 = Clear the corresponding pending interrupt. Read Operation: 0 = No interrupt at Px.n. 1 = Px.n generates an interrupt. **Note**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.8.8.10 Interrupt De-bounce Cycle Control (GPIO_DBCTL)

Register	Offset	R/W	Description	Reset Value
GPIO_DBCTL	GP_BA+0x180	R/W	De-bounce Cycle Control	0x0000_3f00

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[n+8] n=0,1…5	Pn_EN	P0-P5 GPIO enable control 0: Pn port is disable 1: Pn port is enable **Note:** n=0 for p0 n=1 for p1 n=2 for p2 n=3 for p3 n=4 for p4 n=5 for p5
[7:5]	Reserved	Reserved.
[4]	DBCLKSRC	De-bounce Counter Clock Source Selection 0 = De-bounce counter clock source is AHB_CLK. 1 = De-bounce counter clock source is 32 KHz internal low speed RC oscillator (RCL).
[3:0]	DBCLKSEL	De-bounce Sampling Cycle Selection 0000 = Sample interrupt input once per 1 clock. 0001 = Sample interrupt input once per 2 clocks. 0010 = Sample interrupt input once per 4 clocks. 0011 = Sample interrupt input once per 8 clocks. 0100 = Sample interrupt input once per 16 clocks. 0101 = Sample interrupt input once per 32 clocks. 0110 = Sample interrupt input once per 64 clocks. 0111 = Sample interrupt input once per 128 clocks. 1000 = Sample interrupt input once per 256 clocks. 1001 = Sample interrupt input once per 2*256 clocks. 1010 = Sample interrupt input once per 4*256 clocks. 1011 = Sample interrupt input once per 8*256 clocks. 1100 = Sample interrupt input once per 16*256 clocks. 1101 = Sample interrupt input once per 32*256 clocks. 1110 = Sample interrupt input once per 64*256 clocks. 1111 = Sample interrupt input once per 128*256 clocks.

3.8.8.11 GPIO Px.n Data Input/Output (Pxn_PDIO)

Register	Offset	R/W	Description	Reset Value
P00_PDIO	GP_BA+0x200	R/W	GPIO P0.0 Pin Data Input/Output	0x0000_0001
P01_PDIO	GP_BA+0x204	R/W	GPIO P0.1 Pin Data Input/Output	0x0000_0001
P02_PDIO	GP_BA+0x208	R/W	Reserved	0x0000_0001
P03_PDIO	GP_BA+0x20C	R/W	Reserved	0x0000_0001
P04_PDIO	GP_BA+0x210	R/W	GPIO P0.4 Pin Data Input/Output	0x0000_0001
P05_PDIO	GP_BA+0x214	R/W	Reserved	0x0000_0001
P06_PDIO	GP_BA+0x218	R/W	Reserved	0x0000_0001
P07_PDIO	GP_BA+0x21C	R/W	Reserved	0x0000_0001
P10_PDIO	GP_BA+0x220	R/W	GPIO P1.0 Pin Data Input/Output	0x0000_0001
P11_PDIO	GP_BA+0x224	R/W	GPIO P1.1 Pin Data Input/Output	0x0000_0001
P12_PDIO	GP_BA+0x228	R/W	GPIO P1.2 Pin Data Input/Output	0x0000_0001
P13_PDIO	GP_BA+0x22C	R/W	GPIO P1.3 Pin Data Input/Output	0x0000_0001
P14_PDIO	GP_BA+0x230	R/W	GPIO P1.4 Pin Data Input/Output	0x0000_0001
P15_PDIO	GP_BA+0x234	R/W	GPIO P1.5 Pin Data Input/Output	0x0000_0001
P16_PDIO	GP_BA+0x238	R/W	Reserved	0x0000_0001
P17_PDIO	GP_BA+0x23C	R/W	Reserved	0x0000_0001
P20_PDIO	GP_BA+0x240	R/W	Reserved	0x0000_0001
P21_PDIO	GP_BA+0x244	R/W	GPIO P2.1 Pin Data Input/Output	0x0000_0001
P22_PDIO	GP_BA+0x248	R/W	GPIO P2.2 Pin Data Input/Output	0x0000_0001
P23_PDIO	GP_BA+0x24C	R/W	GPIO P2.3 Pin Data Input/Output	0x0000_0001
P24_PDIO	GP_BA+0x250	R/W	GPIO P2.4 Pin Data Input/Output	0x0000_0001
P25_PDIO	GP_BA+0x254	R/W	GPIO P2.5 Pin Data Input/Output	0x0000_0001
P26_PDIO	GP_BA+0x258	R/W	GPIO P2.6 Pin Data Input/Output	0x0000_0001
P27_PDIO	GP_BA+0x25C	R/W	Reserved	0x0000_0001
P30_PDIO	GP_BA+0x260	R/W	Reserved	0x0000_0001
P31_PDIO	GP_BA+0x264	R/W	GPIO P3.1 Pin Data Input/Output	0x0000_0001

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	PDIO	GPIO Px.N Pin Data Input/Output Writing this bit can control one GPIO pin output value. 0 = Corresponding GPIO pin set to low. 1 = Corresponding GPIO pin set to high. Read this register to get GPIO pin status. For example, writing P00_PDIO will reflect the written value to bit DOUT (P0_DOUT[0]), reading P00_PDIO will return the value of PIN (P0_PIN[0]). **Note1**: The writing operation will not be affected by register DATMSK (Px_DATMSK[n]). **Note2**: Max. n=7 for port 0. Max. n=7 for port 1. Max. n=7 for port 2. Max. n=7 for port 3. Max. n=7 for port 4. Max. n=7 for port 5

3.9 Universal Serial Bus(USB)

3.9.1 Overview

The USB device controller is compatible with USB 2.0 Full-speed(12 Mbps) function.

3.9.2 Features

- Conforms to 1.1 and 2.0 revision of the USB specification.
- Support 4 endpoints (include endpoint 0).
- EP0 FIFO: 32Bytes
- BULK Transactions: EP1/2/3 IN FIFO: 64 Bytes, EP1/2/3 OUT FIFO: 64Bytes
- ISOCHRONOUS Transactions: EP1/2/3 IN FIFO: 128 Bytes, EP1/2/3 OUT FIFO: 128 Bytes
- Serial Interface Engine
- Supports full speed devices
- NRZI decoding/encoding
- Bit stuffing/stripping
- 32bytes CRC checking/generation
- On-chip pull-up resistor on USB_DP(1.5kΩ)/USB_DM(150kΩ)
- Support plug in interrupt and plug out interrupt
- Data toggle synchronization mechanism
- Suspend and resume power management functions Conforms to 1.1 and 2.0 revision of the USB specification
- Replace the internal DMA controller of USB with the external public DMA controller
- The number of data received and transmitted by USB must be an integral multiple of the data bit width of DMA controller

3.9.3 Block Diagram

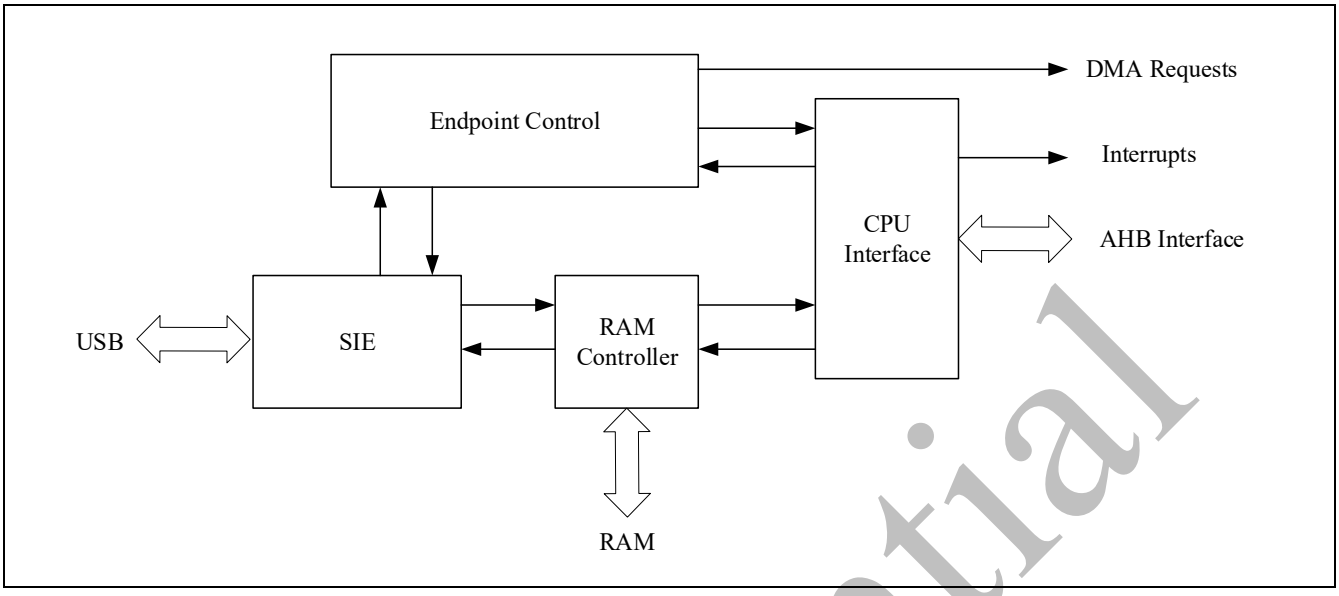


Figure 3-32 USB Block Diagram

3.9.4 Functional Description

3.9.4.1 Support BULK/ISOCHRONOUS transactions

This USB complies with the usb2.0 full_speed protocol and supports bulk and isochronous transmission.

3.9.4.2 USB DMA Operation

SOFTWARE DAM REG: OUT/RX ENDPOINTS

For operation in software dma_req and use DMA_REQ_MODE0, A OUT endpoint should be programmed as follows:

- The relevant interrupt enable bit in the OUT_IER/IN_IER register should be set to 1.
- The CLR_AUTO(D7), OUT_DMA_EN(D5) and OUT_DMA_MODE(D4) bit of the appropriate OUT_EPx_CSR2 register should be set to 0.
- Set EP_INDEX and OUT_MAX_PKT.

When a packet has been received by USB, it will interrupt the processor. The processor should then program the DMA controller as follows:

- DMA_CNTL : DMA_REQ(D0) = 1, DIRECTION(D1) = 0, DMA_REQ_MODE(D2) = 0, ENDPOINT(D4-D7).

The DMA controller will then request bus master-ship and transfer the packet to memory. It will interrupt the processor when it has completed the transfer. The processor can then clear

the OUT_PKT_PREP bit.When the data received by the slave does not reach OUT_MAX_PKT, it is necessary to manually set OUT_PKT_PREP to 1.

SOFTWARE DAM REG : IN/TX ENDPOINTS

For operation in software dma_req and use DMA_REQ_MODE0, a IN endpoint should be programmed as follows:

- The relevant interrupt enable bit in the IN_IER/OUT_IER register should be set to 1.
- The IN_DMA_EN(D4) and SET_IN_PKT_PREP(D7) bit of the appropriate IN_EPx_CSR2 register should be set to 0.
- Set EP_INDEX and IN_MAX_PKT.

When the FIFO becomes available in USB, the USB will interrupt the processor. The processor should then program the DMA controller as follows:

- ADDR : Memory address of packet to send.
- COUNT : Size of packet to be sent.
- DMA_CNTL : DMA_REQ(D0) = 1, DIRECTION (D1) = 1, DMA_REQ_MODE(D2) = 0, ENDPOINT(D4-D7).

The DMA controller will then request bus master-ship and transfer the packet to USB FIFO. It will interrupt the processor when it has completed the transfer and the processor can set the IN_PKT_PREP/OUT_PKT_PREP bit.When the data sent by the slave does not reach IN_MAX_PKT, IN_PKT_PREP needs to be manually set to 1.

HARDWARE DMA REG : OUT/RX ENDPOINTS

For operation in hardware dma_req and use DMA_REQ_MODE1 with a OUT endpoint, the DMA Controller should be programmed as follows:

- DMA_CNTL :DIRECTION(D1)=0, DMA_REQ_MODE(D2)=1, ENDPOINT(D4-D7).
  The OUT/ IN endpoint should be programmed as follows:
- The relevant interrupt enable bit in the OUT_IER/IN_IER register should be set to 1.
- The CLR_AUTO(D7), OUT_DMA_EN(D5) and OUT_DMA_MODE(D4) bits of the appropriate OUT_EPx_CSR2 register should be set to 1.
- In general, the amount of data received by the slave needs to be an integer multiple of OUT_MAX_PKT. When the last packet received by the slave does not reach an integer multiple of OUT_MAX_PKT, it is necessary to set OUT_DMA_EN(D5) and OUT_DMA_MODE(D4) to 0 in advance, which becomes a software DMA request, or use the CPU to transport the received data.

- For ISO DMA transfer, it is required to guarantee(OUT_EPx_CSR2[4]=0).

When a packet is received by the USB, the DMA controller will request bus mastership and transfer the packet to memory. The USB will automatically clear the OUT_PKT_PREP bit in the appropriate OUT_EPx_CSR1/IN_EPx_CSR1 register. This process will continue automatically until the USB receives a 'short packet'(one of less than the maximum packet size for the endpoint) signifying the end of the transfer. This 'short packet' will not be transferred by the DMA controller: instead the USB will interrupt the processor. The processor can then read the OUT_BYTE_NUM/EP0_CNT registers to see the size of the 'short packet' and either unload it manually or reprogram the DMA controller in Mode 0 to unload the packet. The DMA controller ADDR register will have been incremented as the packets were unloaded so the processor can compare the current value of ADDR with the start address of the memory buffer to determine the size of the transfer.

Note: If the size of the transfer exceeds the data buffer size, the DMA controller will stop unloading the FIFO and interrupt the processor.

HARDWARE DMA REG : IN/TX ENDPOINTS

For operation in hardware dma_req and use DMA_REQ_MODE1 with a IN EndPoint, the DMA controller should be programmed as follows:

- DMA_CNTL : DIRECTION(D1)=1,DMA_REQ_MODE(D2)=1, ENDPOINT(D4-D7).
  The IN /OUT endpoint should be programmed as follows:
- The relevant interrupt enable bit in the IN_IER /OUT_IER register should be set to 1 (so that any errors will cause an interrupt to be generated).
- The IN_DMA_EN(D4) and SET_IN_PKT_PREP(D7) bits of the appropriate IN_EPx_CSR2/OUT_EPx_CSR2 register should be set to 1.
- When the data sent by the slave does not reach IN_MAX_PKT, IN_PKT_PREP needs to be manually set to 1.

When the FIFO becomes available in the USB, the DMA controller will request bus mastership and transfer a packet to the FIFO. The USB will automatically set the IN_PKT_PREP/OUT_PKT_PREP bit in the appropriate IN_EPx_CSR1/ OUT_EPx_CSR1 register. This process will continue until the entire data block has been transferred to the USB. The DMA controller will then interrupt the processor. If the last packet to be loaded was less than the maximum packet size for the endpoint, the IN_PKT_PREP/OUT_PKT_PREP bit will not have been set. So the processor should set the IN_PKT_PREP/OUT_PKT_PREP bit

to allow the last 'short packet' to be sent. If the last packet to be loaded was of the maximum packet size, the processor should still set the IN_PKT_PREP/OUT_PKT_PREP bit in order to send a null packet signifying the end of the transfer.

3.9.4.3 Support plug in/out interrupt

Enable USB DM pull-up resistor (ANA_MISC[28]=1), When the USB is plugged in, {USB DP,USB DM}={1,0} will generate a PLUG_IRQ interrupt for more than 1ms. Reading the USB_IRQ, if {USB DP,USB DM}={1,0}, it means that the USB is plugged in; If {USB DP,USB DM}={1,1} exceeds 1ms, PLUG_IRQ interrupt will be generated. Reading the USB_IRQ, if {USB DP,USB DM}={1,1}, it means USB is plugged out.

3.9.5 USB Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
USB Base Address: USB_BA				
FADDR	USB_BA+0x00	R/W	Function address register	0x00
POWER	USB_BA+0x01	R/W	Power management register	0x00
INTRIN1	USB_BA+0x02	R	Interrupt register for Endpoint 0 plus IN Endpoints 1 to 3	0x00
INTRIN2	USB_BA+0x03	R	reserved	0x00
INTROUT1	USB_BA+0x04	R	Interrupt register for OUT Endpoints 1 to 3	0x00
INTROUT2	USB_BA+0x05	R	reserved	0x00
INTRUSB	USB_BA+0x06	R	Interrupt register for common USB interrupts	0x00
INTRIN1E	USB_BA+0x07	R/W	Interrupt enable register for IntrIn1	0xFF
INTRIN2E	USB_BA+0x08	R/W	reserved	0xFF
INTROUT1E	USB_BA+0x09	R/W	Interrupt enable register for IntrOut1	0xFE
INTROUT2E	USB_BA+0x0A	R/W	reserved	0xFF
INTRUSBE	USB_BA+0x0B	R/W	Interrupt enable register for IntrUSB	0x06
FRAME1	USB_BA+0x0C	R/W	Frame number bits 0 to 7	0x0000_0000
FRAME2	USB_BA+0x0D	R	Frame number bits 8 to 10	0x0000_0000
INDEX	USB_BA+0x0E	R/W	Index register for selecting the endpoint status and control registers	0x0000_0000
INMAXP	USB_BA+0x10	R/W	Maximum packet size for IN endpoint (Index register set to select Endpoints 1 – 3 only)	0x0000_0000
CSR0	USB_BA+0x11		Control Status register for Endpoint 0 (Index register set to select Endpoint 0)	
INCSR1		R/W	Control Status register 1 for IN endpoint (Index register set to select Endpoints 1 – 3)	0x0000_0000
INCSR2	USB_BA+0x12	R/W	Control Status register 2 for IN endpoint (Index register set to select Endpoints 1 – 3 only)	0x0000_0000
OUTMAXP	USB_BA+0x13	R/W	Maximum packet size for OUT endpoint (Index register set to select Endpoints 1 – 3 only)	0x0000_0000
OUTCSR1	USB_BA+0x14	R/W	Control Status register 1 for OUT endpoint (Index register set to select Endpoints 1 – 3 only)	0x00FF_0000
OUTCSR2	USB_BA+0x15	R/W	Control Status register 2 for OUT endpoint (Index register set to select Endpoints 1 – 3 only)	0x0000_0000
COUNT0	USB_BA+0x16	R/W	Number of received bytes in Endpoint 0 FIFO (Index register set to select Endpoint 0)	0x0000_0000
OUTCOUNT1			Number of bytes in OUT endpoint FIFO (lower byte) (Index register set to select Endpoints 1 – 3)	0x0000_0000
OUTCOUNT2	USB_BA+0x17	R	Number of bytes in OUT endpoint FIFO (upper byte) (Index register set to select Endpoints 1 – 3 only)	0x0000_0000
FIFOx	USB_BA+0x20~2F	R/W	FIFOs for Endpoints 0 to 3	0x0000_0000
INTR	USB_BA+0x200	R/W	Pending interrupts register	0x0000_0000
CNTL	USB_BA+0x204	R/W	DMA Control register	0x0000_0000
ADDR	USB_BA+0x208	R/W	DMA AHB Memory Address register	0x0000_0000

COUNT	USB_BA+0x20C	R/W	DMA Byte Count register	0x0000_0000

3.9.6 USB Register Description

3.9.6.1 FADDR

Register	Offset	R/W	Description	Reset Value
FADDR	USB_BA+0x00	R/W	Function address register	0x00

Bits	Descriptions	
[7]	UPDATE	Set when FADDR is written. Cleared when the new address takes effect (at the end of the current transfer).
[6:0]	FUNC ADDR	The function address.

3.9.6.2 POWER

Register	Offset	R/W	Description	Reset Value
POWER	USB_BA+0x01	R/W	Power management register.	0x00

Bits	Descriptions	
[7]	ISO UPDATE	When set by the CPU the USB will wait for an SOF token from the time InPktRdy is set before sending the packet. If an IN token is received before an SOF token, then a zero length data packet will be sent. This bit is only used by endpoints performing Isochronous transfers.
[6:4]	Reserved	Reserved.
[3]	RESET	This read only bit is set while Reset signaling is present on the bus.
[2]	RESUME	Set by the CPU to generate Resume signaling when the function is in Suspend mode. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling.
[1]	SUSPEND MODE	Set by the USB when Suspend mode is entered. Cleared when the CPU reads the interrupt register, or sets the Resume bit of this register.
[0]	ENABLE SUSPEND	Set by the CPU to enable entry into Suspend mode when Suspend signaling is received on the bus.

3.9.6.3 INTRIN1

Register	Offset	R/W	Description	Reset Value
INTRIN1	USB_BA+0x02	R	Interrupt register for Endpoint 0 plus IN Endpoints 1 to 3. **Note**: All active interrupts will be cleared when this register is read.	0x00

Bits	Descriptions	
[7:4]	Reserved	Reserved
[3]	EP3	IN Endpoint 3 interrupt.
[2]	EP2	IN Endpoint 2 interrupt.
[1]	EP1	IN Endpoint 1 interrupt.
[0]	EP0	Endpoint 0 interrupt.

3.9.6.4 INTRIN2

Register	Offset	R/W	Description	Reset Value
INTRIN2	USB_BA+0x03	R	Reserved	0x00

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.9.6.5 INTROUT1

Register	Offset	R/W	Description	Reset Value
INTROUT1	USB_BA+0x04	R	Interrupt register for OUT Endpoints 1 to 3. **Note:** All active interrupts will be cleared when this register is read.	0x00

Bits	Descriptions	
[7:4]	Reserved	Reserved
[3]	EP3	OUT Endpoint 3 interrupt.
[2]	EP2	OUT Endpoint 2 interrupt.
[1]	EP1	OUT Endpoint 1 interrupt.
[0]	Reserved	Reserved

3.9.6.6 INTROUT2

Register	Offset	R/W	Description	Reset Value
INTROUT2	USB_BA+0x05	R	Reserved	0x00

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.9.6.7 INTRUSB

Register	Offset	R/W	Description	Reset Value
INTRUSB	USB_BA+0x06	R	Interrupt register for common USB interrupts	0x00

Bits	Descriptions	
[7]	DP	D+ status
[6]	DM	D- status
[5]	Reserved	Reserved
[4]	PUPIF	Plug in or plug out interrupt happens.
[3]	SOF	Set at the start of each frame.
[2]	RESET	Set when Reset signaling is detected on the bus.
[1]	RESUME	Set when Resume signaling is detected on the bus while the USB is in Suspend mode.
[0]	SUSPEND	Set when Suspend signaling is detected on the bus.

3.9.6.8 INTRIN1E

Register	Offset	R/W	Description	Reset Value
INTRIN1E	USB_BA+0x07	R/W	Interrupt enable register for INTRIN1. On reset, the bits corresponding to Endpoint 0 and the IN endpoints included in the design are set to 1, while the remaining bits are set to 0.	0xFF

Bits	Descriptions	
[7:4]	Reserved	Reserved
[3]	EP3	IN Endpoint 3 interrupt enable bit.
[2]	EP2	IN Endpoint 2 interrupt enable bit.
[1]	EP1	IN Endpoint 1 interrupt enable bit.
[0]	EP0	IN Endpoint 0 interrupt enable bit.

3.9.6.9 INTRIN2E

Register	Offset	R/W	Description	Reset Value
INTRIN2E	USB_BA+0x08	R/W	Reserved	0xFF

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.9.6.10 INTROUT1E

Register	Offset	R/W	Description	Reset Value
INTROUT1E	USB_BA+0x09	R/W	Interrupt enable register for IntrOut1.On reset, the bits corresponding to the OUT endpoints included in the design are set to 1, while the remaining bits are set to 0.	0xFE

Bits	Descriptions	
[7:4]	Reserved	Reserved
[3]	EP3	OUT Endpoint 3 interrupt enable bit.
[2]	EP2	OUT Endpoint 2 interrupt enable bit.
[1]	EP1	OUT Endpoint 1 interrupt enable bit.
[0]	Reserved	Reserved

3.9.6.11 INTROUT2E

Register	Offset	R/W	Description	Reset Value
INTROUT2E	USB_BA+0x0A	R/W	reserved	0xFF

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.9.6.12 INTRUSBE

Register	Offset	R/W	Description	Reset Value
INTRUSBE	USB_BA+0x0B	R/W	Interrupt enable register for IntrUSB	0x06

Bits	Descriptions	
[7:5]	Reserved	Reserved.
[4]	PUPIF	Plug in or plug out interrupt enable bit
[3]	SOF	Sof interrupt enable bit.
[2]	RESET	Reset interrupt enable bit.

[1]	RESUME	Resume interrupt enable bit.
[0]	SUSPEND	Suspend interrupt enable bit.

3.9.6.13 FRAME1

Register	Offset	R/W	Description	Reset Value
FRAME1	USB_BA+0x0C	R	Frame number register.	0x00

Bits	Descriptions	
[7:0]	FRAME1	Lower 8 bits of the last received frame number.

3.9.6.14 FRAME2

Register	Offset	R/W	Description	Reset Value
FRAME2	USB_BA+0x0D	R	Frame number register.	0x00

Bits	Descriptions	
[7:3]	Reserved	Reserved
[2:0]	FRAME2	Upper 3 bits of the last received frame number.

3.9.6.15 INDEX

Register	Offset	R/W	Description	Reset Value
INDEX	USB_BA+0x0E	R/W	Choose EndPoints.	0x00

Bits	Descriptions	
[7:4]	Reserved	Reserved
[3:0]	INDEX	Index is a 4-bit register that determines which endpoint control/status registers are accessed. Each IN endpoint and each OUT endpoint have their own set of control/status registers. Only one set of IN control/status and one set of OUT control/status registers appear in the memory map at any one time. Before accessing an endpoint's control/status registers, the endpoint number should be written to the Index register to ensure that the correct control/status registers appear in the memory map.

3.9.6.16 INMAXP

Register	Offset	R/W	Description	Reset Value
INMAXP	USB_BA+0x10	R/W	Maximum Packet Size/transaction	0x00

Bits	Descriptions	
[7:0]	INMAXP	InMaxP is an 8-bit register that holds the maximum packet size for transactions through the currently-selected IN endpoint – in units of 8 bytes. The value of InMaxP should be less than 8，because max FIFO size is 64。There is an InMaxP register for each IN endpoint (except Endpoint 0).

3.9.6.17 CSR0

Register	Offset	R/W	Description	Reset Value
CSR0	USB_BA+0x11	R/W	CSR0 is an 8-bit register that provides control and status bits for Endpoint 0.	0x00

Bits	Descriptions	
[7]	ServicedSetupEnd	R/W. The CPU writes a 1 to this bit to clear the SetupEnd bit. It is cleared automatically.
[6]	ServicedOutPktRdy	R/W. The CPU writes a 1 to this bit to clear the OutPktRdy bit. It is cleared automatically.
[5]	SendStall	R/W. The CPU writes a 1 to this bit to terminate the current transaction. The STALL handshake will be transmitted and then this bit will be cleared automatically.
[4]	SetupEnd	This bit will be set when a control transaction ends before the DataEnd bit has been set. An interrupt will be generated and the FIFO flushed at this time. The bit is cleared by the CPU writing a 1 to the ServicedSetupEnd bit.
[3]	DataEnd	The CPU sets this bit: 1. When setting InPktRdy for the last data packet. 2. When clearing OutPktRdy after unloading the last data packet. 3. When setting InPktRdy for a zero length data packet. It is cleared automatically.
[2]	SentStall	R/CLEAR. This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.
[1]	InPktRdy	R/W. The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is generated when the bit is cleared. It is cleared automatically.
[0]	OutPktRdy	R. This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU clears this bit by setting the ServicedOutPktRdy bit.

3.9.6.18 COUNT0

Register	Offset	R/W	Description	Reset Value
COUNT0	USB_BA+0x16	R	It indicates the number of received data bytes in the Endpoint 0 FIFO	0x00

Bits	Descriptions	
[7:0]	Count0	EndPoint0 OUT Count.

3.9.6.19 INCSR1

Register	Offset	R/W	Description	Reset Value
INCSR1	USB_BA+0x11	R	InCSR1 is an 8-bit register that provides control and status bits for transfers through the currently-selected IN endpoint. There is an InCSR1 register for each IN endpoint (not including Endpoint 0).	0x00

Bits	Descriptions	
[7]	Reserved	Reserved
[6]	ClrDataTog	R/W. The CPU writes a 1 to this bit to reset the endpoint IN data toggle to 0.
[5]	SentStall	R/CLEAR. This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the InPktRdy bit is cleared (see below). The CPU should clear this bit by reading.
[4]	SendStall	R/W. The CPU writes a 1 to this bit to issue a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition. This bit has no effect if the IN endpoint is in ISO mode.
[3]	FlushFIFO	R/W. Self-clearing.The CPU writes a 1 to this bit to flush the next packet to be transmitted from the endpoint INFIFO. The FIFO pointer is reset and the InPktRdy bit (below) is cleared. Note: If the FIFO contains two packets, FlushFIFO will need to be set twice to completely clear the FIFO. It is cleared automatically.
[2]	UnderRun	R/CLEAR. In ISO mode, this bit is set when a zero length data packet is sent after receiving an IN token with the InPktRdy bit not set. In Bulk/Interrupt mode, this bit is set when a NAK is returned in response to an IN token. The CPU should clear this bit by reading.
[1]	FIFONotEmpty	This bit is set when there is at least 1 packet in the IN FIFO.
[0]	InPktRdy	The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated (if enabled) when the bit is cleared.

3.9.6.20 INCSR2

Register	Offset	R/W	Description	Reset Value
INCSR2	USB_BA+0x12	R/W	InCSR2 is an 8-bit register that provides further control bits for transfers through the currently-selected IN endpoint. There is an InCSR2 register for each IN endpoint (not including Endpoint 0).	0x00

Bits	Descriptions	
[7]	AutoSet	If the CPU sets this bit then InPktRdy will be automatically set when data of the maximum packet size (value in InMaxP) is loaded into the IN FIFO. If a packet of less than the maximum packet size is loaded, then InPktRdy will have to be set manually
[6]	ISO	The CPU sets this bit to enable the IN endpoint for isochronous transfers, and clears it to enable the IN endpoint for bulk or interrupt transfers.
[5]	Mode	The CPU sets this bit to enable the endpoint direction as IN, and clears it to enable the endpoint direction as OUT. Valid only where the same endpoint FIFO is used for both IN and OUT transactions.
[4]	Reserved	Reserved
[3]	FrcDataTog	The CPU sets this bit to force the endpoint IN data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by interrupt IN endpoints that are used to communicate rate feedback for isochronous endpoints.
[2:0]	Reserved	Reserved

3.9.6.21 OUTMAXP

Register	Offset	R/W	Description	Reset Value
OUTMAXP	USB_BA+0x13	R/W	Maximum Packet Size/transaction for EndPoint1/2/3	0x00

Bits	Descriptions	
[7:0]	OUTMAXP	OutMaxP is an 8-bit register that holds the maximum packet size for transactions through the currently-selected OUT endpoint – in units of 8 bytes(expect EndPoint0). The value of OutMaxP should be less than 8, because max FIFO size is 64。

3.9.6.22 OUTCSR1

Register	Offset	R/W	Description	Reset Value
OUTCSR1	USB_BA+0x14	R/W	It provides control and status bits for transfers through the currently-selected OUT endpoint.	0x00

Bits	Descriptions	
[7]	ClrDataTog	The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
[6]	SentStall	R/CLEAR. This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.
[5]	SendStall	R/W. The CPU writes a 1 to this bit to issue a STALL handshake. The CPU clears this bit to terminate the stall condition. This bit has no effect if the OUT endpoint is in ISO mode.
[4]	FlushFIFO	The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint OUT FIFO. Note: If the FIFO contains two packets, FlushFIFO will need to be set twice to completely clear the FIFO. It is cleared automatically.
[3]	DataError	R. This bit is set when OutPktRdy is set if the data packet has a CRC or bit-stuff error. it is cleared when OutPktRdy is cleared. The bit is only valid in ISO mode.
[2]	OverRun	R/CLEAR. This bit is set if an OUT packet cannot be loaded into the OUT FIFO. The CPU should clear this bit. The bit is only valid in ISO mode.
[1]	FIFOFull	R. This bit is set when no more packets can be loaded into the OUT FIFO. It is cleared automatically.
[0]	OutPktRdy	R/CLEAR. This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the OUT FIFO. An interrupt is generated when the bit is set.

3.9.6.23 OUTCSR2

Register	Offset	R/W	Description	Reset Value
OUTCSR2	USB_BA+0x16	R/W	OutCSR2 is an 8-bit register that provides further control bits for transfers through the currently-selected OUT endpoint.	0x00

Bits	Descriptions	
[7]	AutoClear	If the CPU sets this bit then the OutPktRdy bit will be automatically cleared when a packet of OutMaxP bytes has been unloaded from the OUT FIFO. When packets of less than the maximum packet size are unloaded, OutPktRdy will have to be cleared manually.
[6]	ISO	The CPU sets this bit to enable the OUT endpoint for isochronous transfers, and clears it to enable the OUT endpoint for bulk or interrupt transfers.
[5]	DMAEnab	The CPU sets this bit to enable the DMA request for the OUT endpoint.
[4]	DMAMode	Two modes of DMA operation are supported: DMA Mode 0 in which a DMA request is generated for all received packets, together with an interrupt (if enabled); and DMA Mode 1 in which a DMA request (but no interrupt) is generated for OUT

		packets of size OutMaxP bytes and an interrupt (but no DMA request) is generated for OUT packets of any other size. The CPU sets this bit to select DMA Mode 1 and clears this bit to select DMA Mode 0.
[3:0]	Reserved	Reserved

3.9.6.24 OUTCOUNT1

Register	Offset	R/W	Description	Reset Value
OUTCOUNT1	USB_BA+0x16	R	Endpoint OUT Count – lower 8 bits	0x00

Bits	Descriptions	
[7:0]	OUTCOUNT1	OutCount1 is a 8-bit read-only register that holds the lower 8 bits of the number of received data bytes in the packet in the FIFO associated with the currently-selected OUT endpoint. The value returned is valid while OutPktRdy (OutCSR1.D0) is set

3.9.6.25 OUTCOUNT2

Register	Offset	R/W	Description	Reset Value
OUTCOUNT2	USB_BA+0x17	R	Endpoint OUT Count – upper 3 bits	0x00

Bits	Descriptions	
[7:3]	Reserved	Reserved
[2:0]	OUTCOUNT2	OutCount2 is a 3-bit read-only register that holds the upper 3 bits of the number of received data bytes in the packet in the FIFO associated with the currently-selected OUT endpoint. The value returned is valid while OutPktRdy (OutCSR1.D0) is set.

3.9.6.26 FIFOx

Register	Offset	R/W	Description	Reset Value
FIFOx	USB_BA+0x20	R/W		0x00

FIFOx	Descriptions
FIFO3	FIFO3 for EndPoint3. Address: USB_BA+0x2C, SIZE:128byte
FIFO2	FIFO2 for EndPoint2. Address: USB_BA+0x28, SIZE:128byte
FIFO1	FIFO1 for EndPoint1. Address: USB_BA+0x24, SIZE:128byte
FIFO0	FIFO0 for EndPoint0. Address: USB_BA+0x20, SIZE:64byte

3.9.6.27 INTR

Register	Offset	R/W	Description	Reset Value
INTR	USB_BA+0x200	R/W	USB DMA channel interrupt register	0x00

Bits	Descriptions	
[31:1]	Reserved	Reserved
[0]	INTR	When USB DMA transimits or receive datas are finished, USB DMA interrupt happens.

3.9.6.28 CNTL

Register	Offset	R/W	Description	Reset Value
CNTL1	USB_BA+0x204	R/W	USB DMA control register	0x00

Bits	Descriptions	
[15]	Bus error	If a bus error occurs while the DMA controller is accessing memory on the AHB, the DMA controller will immediately terminate the DMA transfer and interrupt the processor with the Bus Error (D15) bit of the CNTL register set. Note: The DMA controller does not support split retries and will treat them as bus errors.
[14:8]	Max packect size	Max packect size in unit of 8 bytes. The values is set in the corresponding InMaxP or OUTMAXP.
[7:4]	Endpoint number	Choose which EndPoint to transmit or receive data
[3]	Interrupt enable	Enable interrupt in register INTR
[2]	Dma_mode	0:Dma_mode0 1:dma_mode1
[1]	Direction	0: wirte-out endpoint 1: read-IN endpoint
[0]	Enable DMA	Enable DMA transfer.

3.9.6.29 ADDR

Register	Offset	R/W	Description	Reset Value
ADDR1	USB_BA+0x208	R/W	Endpoint OUT Count – upper 3 bits	0x00

Bits	Descriptions	
[31:0]	AHB Memory Address	Memory address of packet to send for IN-EndPointx(x=1/2/3) or Memory address of buffer to store transfer for OUT-EndPointx(x=1/2/3)

3.9.6.30 COUNT

Register	Offset	R/W	Description	Reset Value
COUNT1	USB_BA+0x20C	R/W	DMA Byte Count	0x00

Bits	Descriptions	
[31:0]	DMA Byte count	Size of packet to be sent by EndPointx(x=1/2/3) or Maximum size of data buffer

3.10 Enhanced PWM Generator (PWM)

3.10.1 Overview

The PAN101x series has built in three PWM unit (following named as PWM0, PWM1 and PWM2) which is specially designed for motor driving control applications. Each PWM unit supports eight PWM generators which can be configured as eight independent outputs, from CH0 to CH7 (CH as an abbreviation for channel). Also, eight CHs can be acted as four complementary output pairs, (CH0, CH1), (CH2, CH3), (CH4, CH5) and (CH6, CH7) with four programmable dead-time generators, or as four synchronous output pairs, (CH0, CH1), (CH2, CH3), (CH4, CH5)and (CH6, CH7).

Every complementary PWM pairs share one 8-bit prescaler. There are eight clock dividers providing five divided frequencies (1, 1/2, 1/4, 1/8, 1/16) for each channel. Each PWM output has independent 16-bit counter for PWM period control, and 16-bit comparators for PWM duty control. The eight PWM generators provide sixteen independent PWM interrupt flags which are set by hardware when the corresponding PWM period counter comparison matched period and duty. Each PWM interrupt source with its corresponding enable bit can request PWM interrupt. The PWM generators works in Auto-reload mode to output PWM waveform continuously.

To prevent PWM driving output pin with unsteady waveform, the 16-bit period down counter and 16-bit comparator are implemented with double buffer. When user writes data to counter/comparator buffer registers, the updated value will be loaded into the 16-bit down counter/comparator at the end of current period. The double buffering feature avoids glitch at PWM outputs.

Besides PWM, Motor controlling also need Timer and ADC to work together. In order to control motor more precisely, we provide some registers that not only configure PWM but also Timer and ADC, by doing so, it can save more CPU time and control motor with ease especially in BLDC.

3.10.2 Features

- Support 1 sets of PWM， each set PWM has 8 channels which is described as below
- Eight independent 16-bit PWM duty control units with maximum eight port pins:
  - Eight independent PWM outputs – CH0, CH1, CH2, CH3, CH4, CH5,CH6 and CH7
  - Four complementary PWM pairs, with each pin in a pair mutually complement to each

other and capable of programmable dead-time insertion – (CH0, CH1), (CH2, CH3), (CH4, CH5) and (CH6, CH7)

- Four synchronous PWM pairs, with each pin in a pair in-phase – (CH0, CH1), (CH2, CH3), (CH4, CH5)and (CH6, CH7)

- Group control bit – CH2, CH4 and CH6 are synchronized with CH0, CH3, CH5 and CH7 are synchronized with CH1
- Auto-reload mode PWM
- Up to 16-bit resolution
- Supports edge-aligned, center-aligned and precise center-aligned mode
- Supports asymmetric PWM generating in center-aligned and precise center-aligned mode
- Supports center loading in center-aligned and precise center-aligned mode
- Programmable dead-time insertion between complementary paired PWMs
- Each pin of CH0 to CH7 has independent polarity setting control
- The PWM signals before polarity control stage are defined in the view of low logic. The PWM ports is active high or active low are controlled by polarity control register
- Supports mask aligned function
- Supports independently rising CMP matching, PERIOD matching, falling CMP matching (in Center-aligned type), PERIOD matching to trigger ADC conversion
- Timer comparing matching event trigger PWM to do phase change in BLDC application
- Provides interrupt accumulation function
- Support work at deepsleep mode with 32K source clock

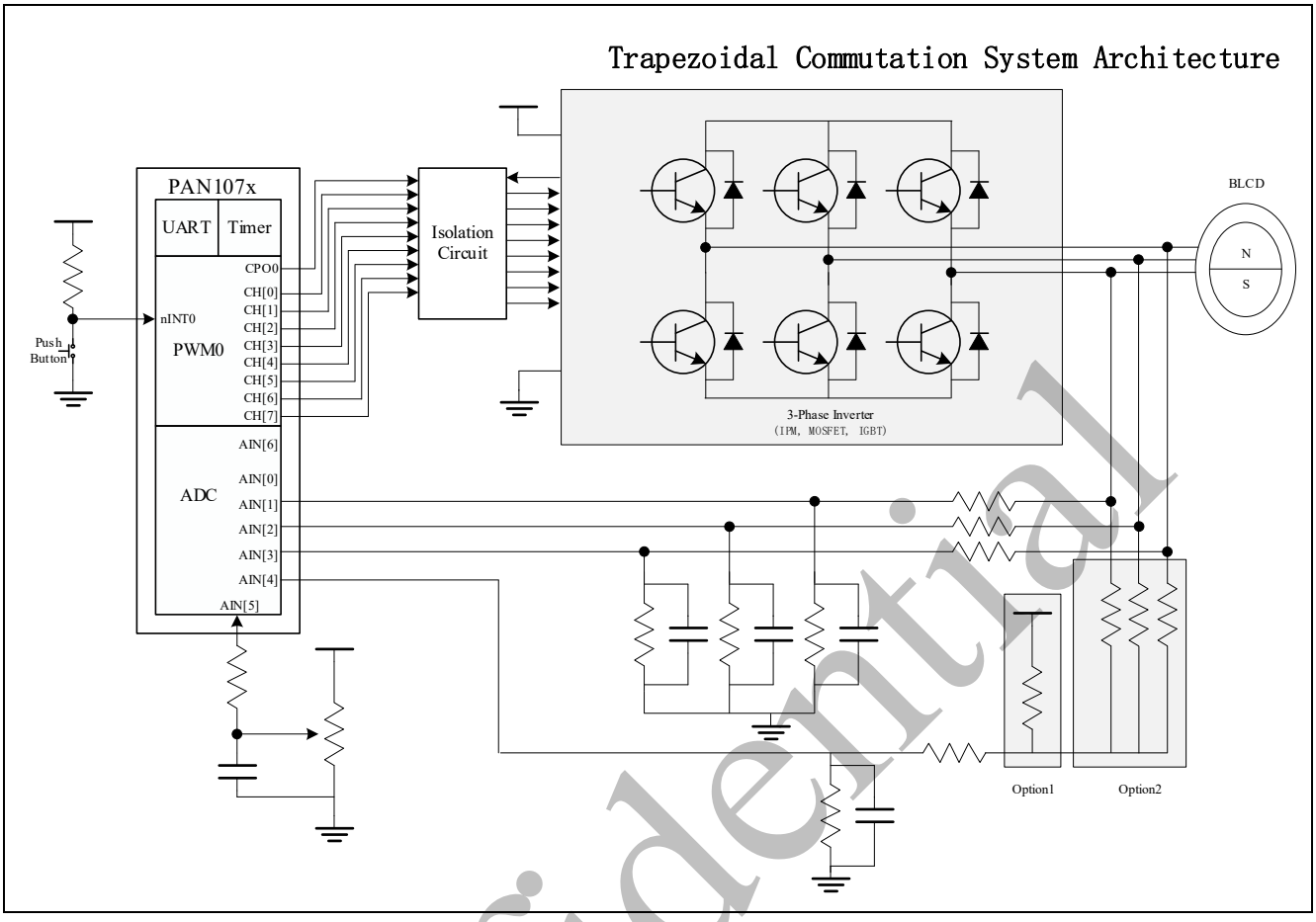


Figure 3-33 Application Circuit Diagram

3.10.3 Block Diagram

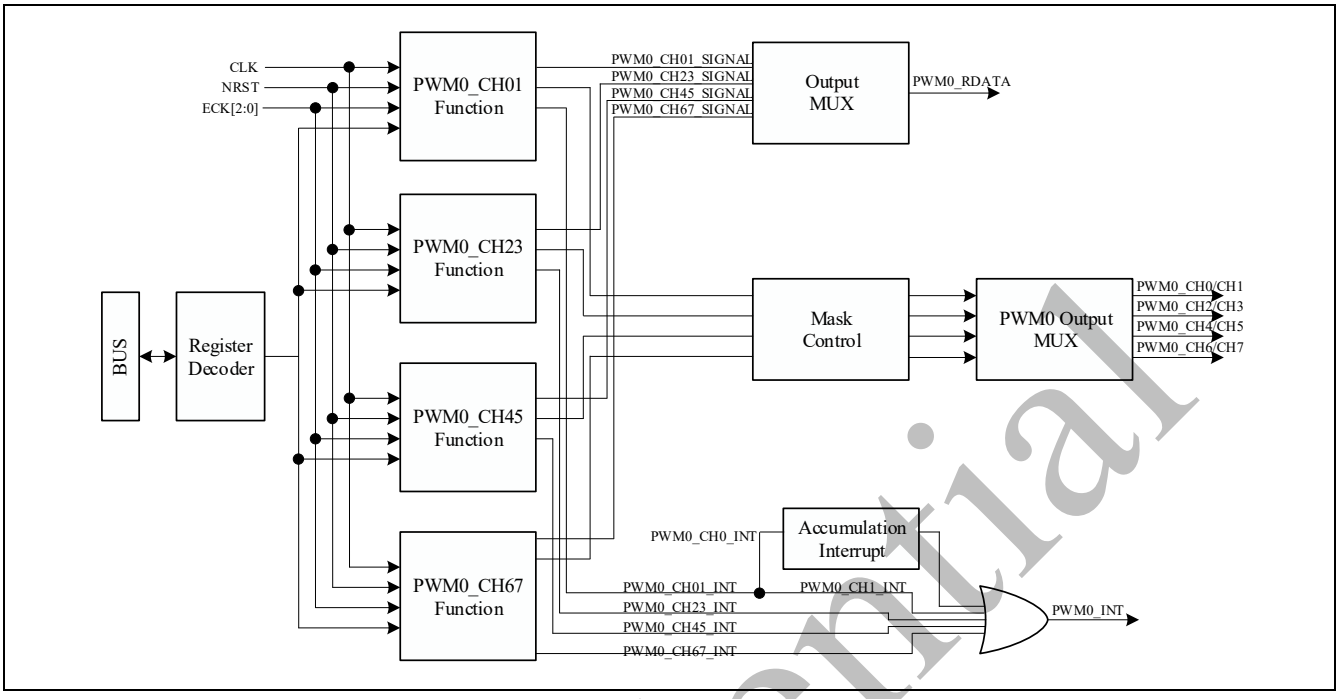


Figure 3-34 PWM0 Block Diagram

Figure 3-34 shows the architecture of PWM0 in pair (e.g. PWM Counter 0/1 are in one pair and PWM Counter 2/3 are in another one, and so on).

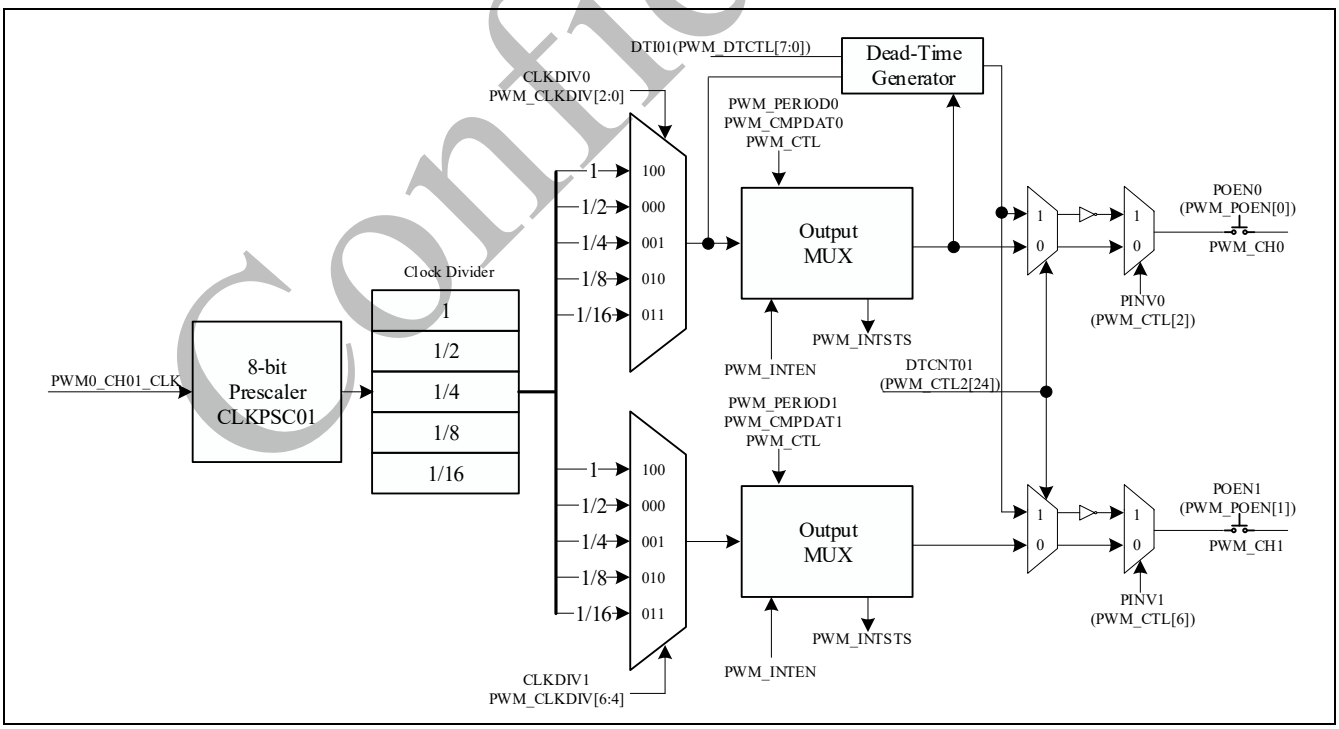


Figure 3-35 PWM0 Generator 0 Architecture Diagram

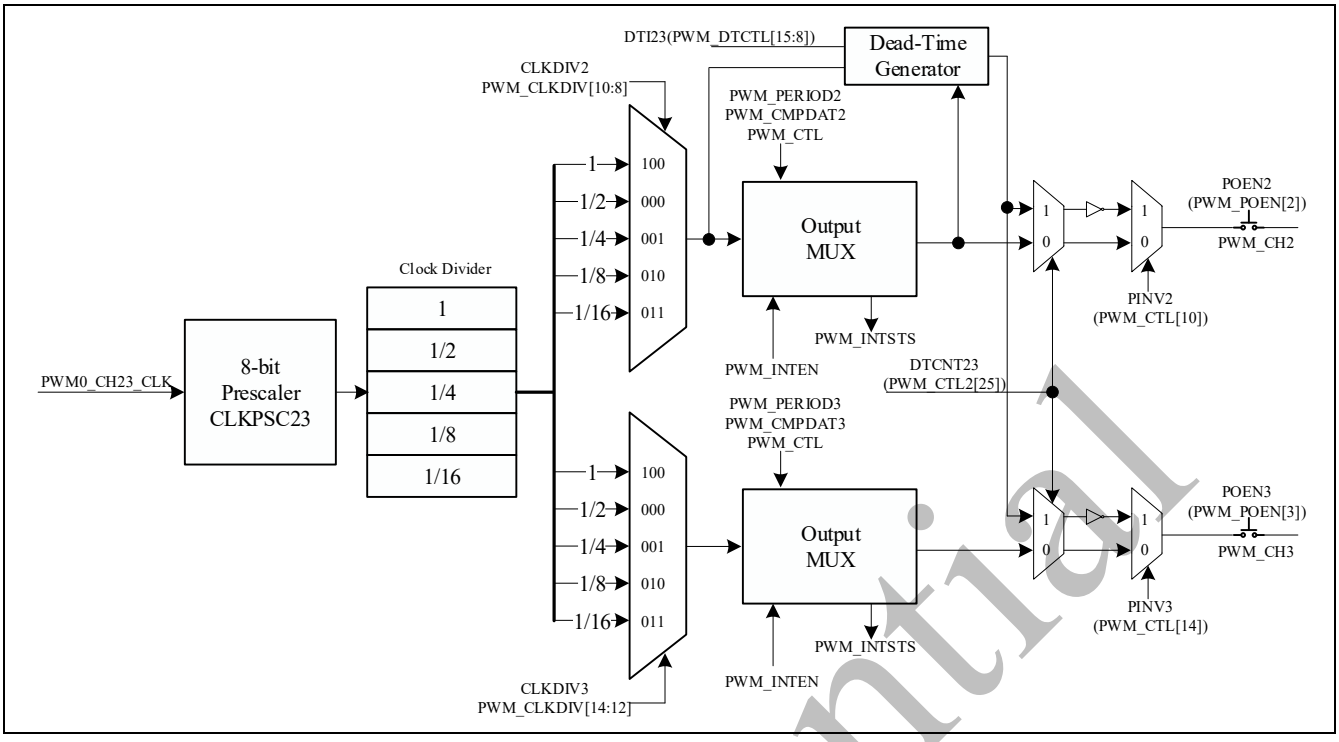


Figure 3-36 PWM0 Generator 2 Architecture Diagram

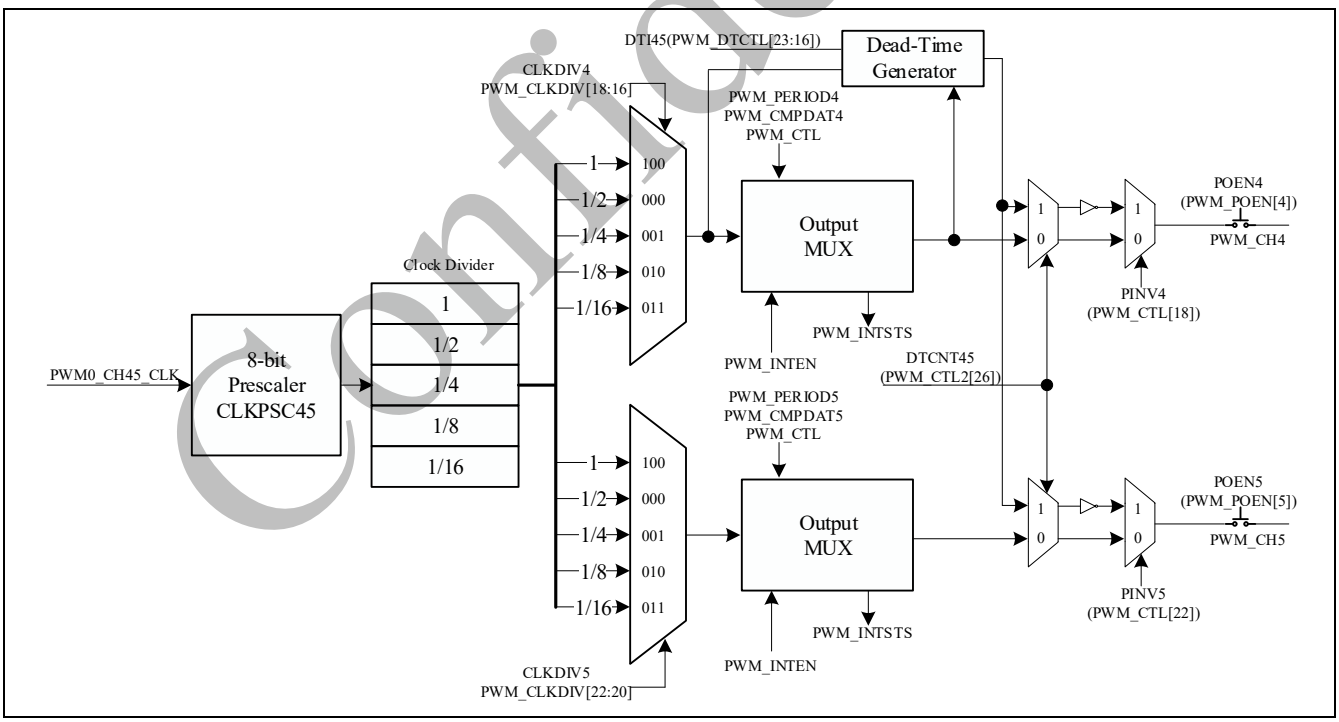


Figure 3-37 PWM0 Generator 4 Architecture Diagram

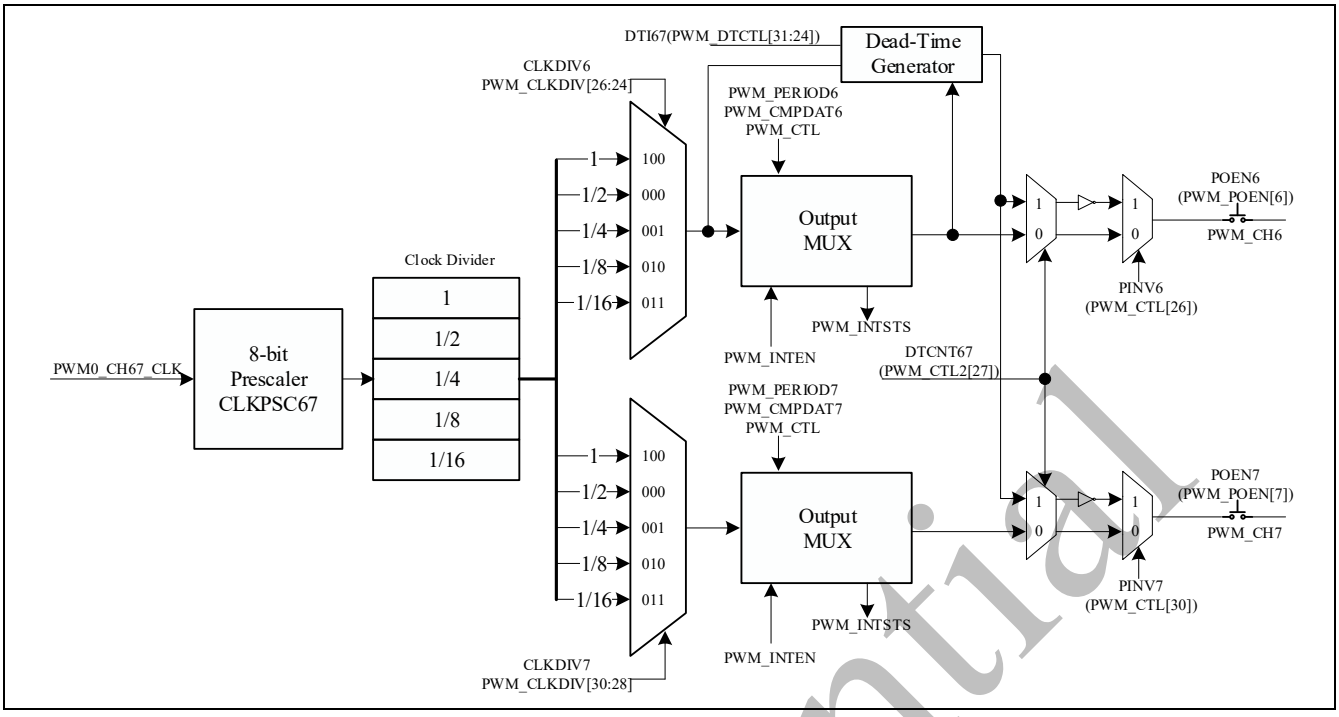


Figure 3-38 PWM0 Generator 6 Architecture Diagram

3.10.4 Basic Configuration.

The PWM clock is configured by AHB_CLK_CTRL[3] register and APB1_CLK_CTRL0 register, also can be reset by IPRST1 register .

The PWM pin functions are configured in SYS_P0_MFP, SYS_P1_MFP, SYS_P2_MFP, SYS_P3_MFP, SYS_P4_MFP and SYS_P5_MFP registers.

3.10.5 Functional Description

3.10.5.1 PWM Counter Type

This device supports three operation types: Edge-aligned, Center-aligned and Precise center-aligned type.

Following equations show the formula for period and duty for each PWM counter operation type:

Edge-aligned (Down counter):

Duty ratio = (CMP + 1) / (PERIOD + 1)

*Duty = (CMP + 1) * (clock period)*

*Period = (PERIOD + 1) * (clock period)*

Center-aligned (Up and Down Counter):

Duty ratio = (PERIOD - CMP) / (PERIOD + 1)

*Duty = (PERIOD - CMP) * 2 * (clock period)*

*Period = (PERIOD + 1) * 2 * (clock period)*

Precise Center-aligned (Up and Down Counter):

*Duty ratio = (PERIOD - (CMP + 1) * 2) / PERIOD*

*Duty = (PERIOD - (CMP + 1) * 2) * (clock period)*

*Period = (PERIOD) * (clock period)*

Note: Clock period is configured by CLKPSC and CLKDIV register.

Edge-aligned PWM (Down-counter)

In Edge-aligned PWM Output type, the 16-bit PWM counter will start counting-down from PERIODn to match with the value of the duty cycle CMPn (old); when this happens it will toggle the CHn output to high and set up CMPDIF compare down match interrupt flag. The counter will continue counting-down to zero; at this moment, it toggles the CHn output to low and CMPn (new) and PERIODn (new) are updated with CNTMODEn=1 and set PIF period interrupt flag.

Figure 3-39, Figure 3-40 and Figure 3-41 show the Edge-aligned PWM timing and operation flow.

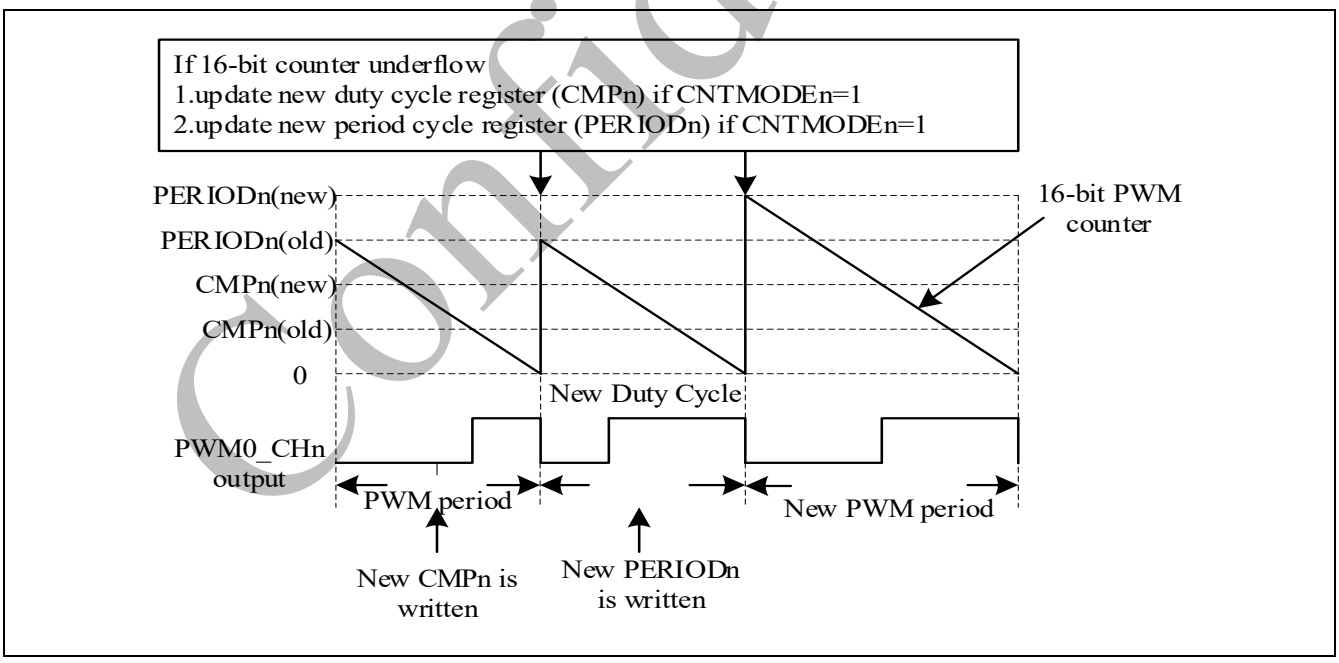


Figure 3-39 Edge-aligned Type PWM

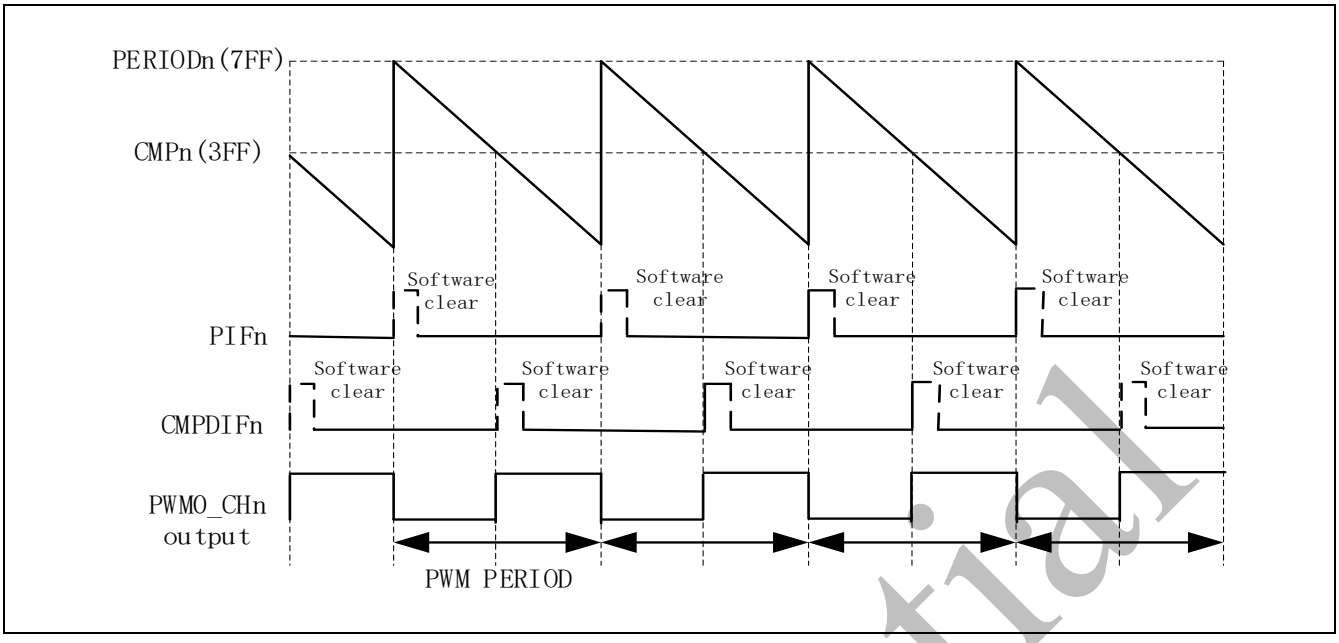


Figure 3-40 PWM Edge-aligned Waveform Timing Diagram

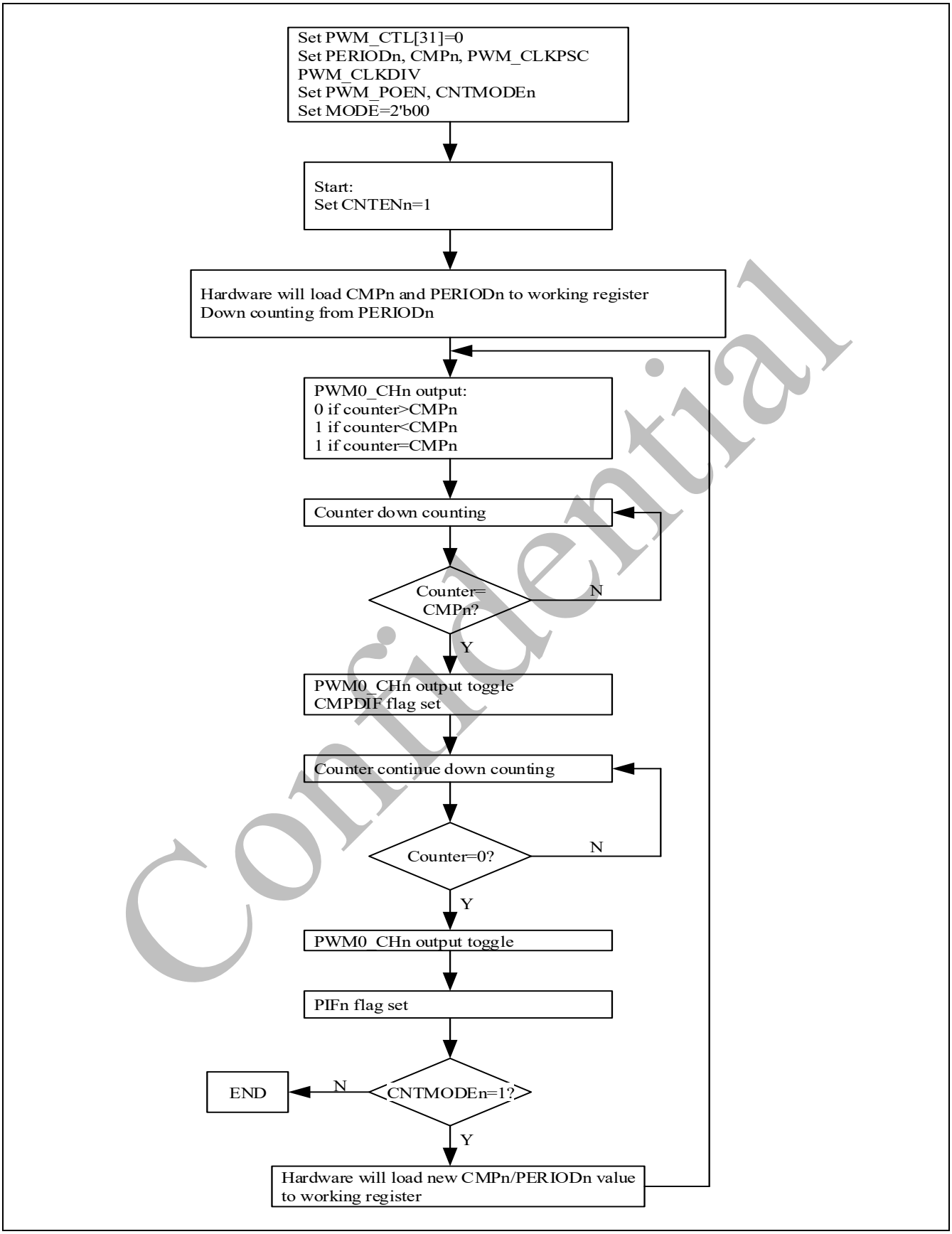


Figure 3-41 Edge-aligned Flow Diagram

The PWM period and duty control are decided by PWM down-counter period register *(PERIODn)* and PWM comparator register (*CMPn*). The PWM counter timing operation is shown in Figure 3-43. The pulse width modulation follows the formula below and the legend of PWM counter Comparator is shown in Figure 3-42. Note that the corresponding GPIO pins must be configured as PWM function (enable *PWM_POEN*) for the corresponding PWM channel.

PWM frequency = *APBCLK / ((CLKPSCnm + 1) * (clock divider)) / (PERIOD + 1)*; where nm, could be 01, 23, 45 or 67 depending on the selected PWM channel, clock divider is configured by CLKDIV register.

Duty ratio = (CMP + 1) / (PERIOD + 1)

PERIOD =0: PWM output is always low.

When PERIOD!=0, PWM output is as follow:

a. CMP ≥ PERIOD: PWM output is always high
b. CMP < PERIOD: PWM low width = (PERIOD - CMP) unit[1]; PWM high width = (CMP+1) unit
c. CMP = 0: PWM is always low

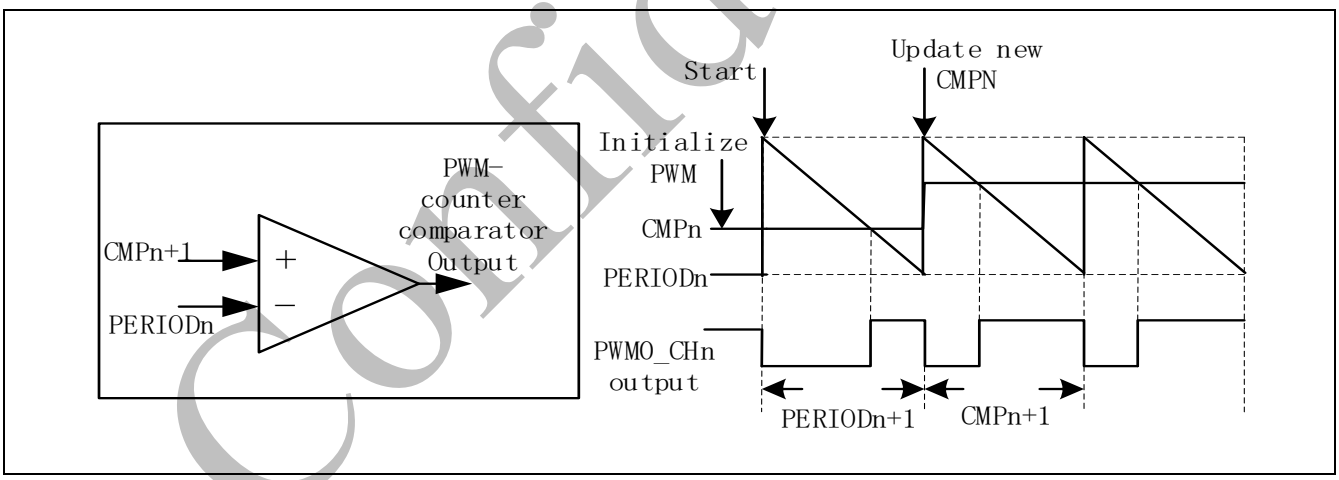


Figure 3-42 Legend of Internal Comparator Output of PWM Counter

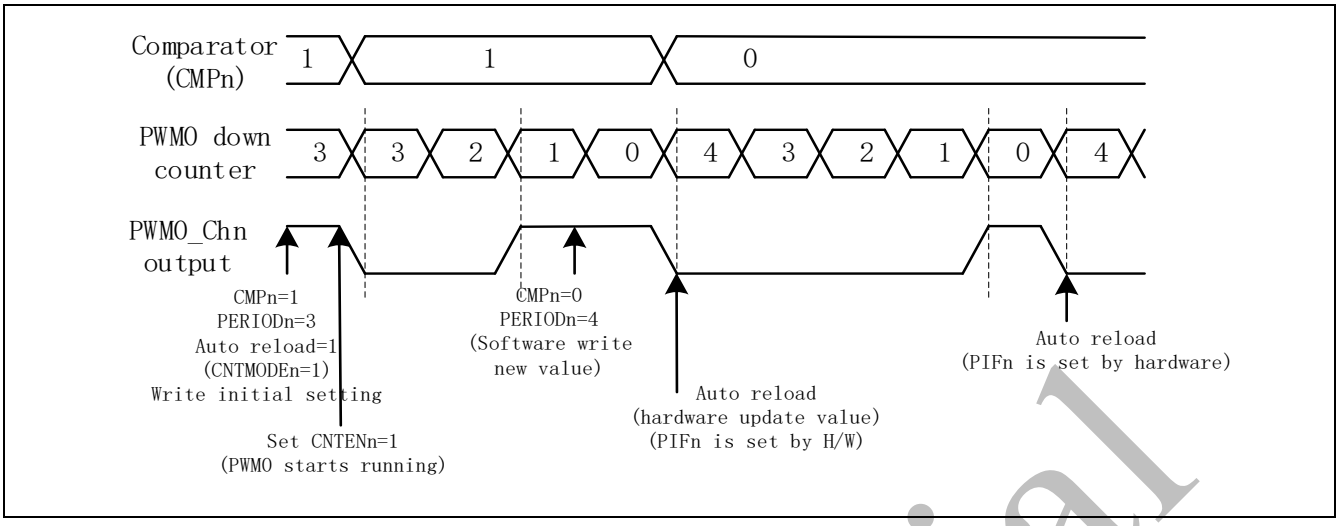


Figure 3-43 PWM Counter Operation Timing

Center-Aligned PWM (up/down counter)

The center-aligned PWM signals are produced by the module when the PWM time base is configured in an Up/Down Counting type. The PWM counter will start counting-up from 0 to match the value of CMPn (old); this will cause the toggling of the CHn generator output to high and set up *CMPUIF* compare up match interrupt flag. The counter will continue counting to match with the PERIODn (old). Upon reaching this state counter is configured automatically to down counting and set up PIF period interrupt flag, when PWM counter matches the CMPn (old) value again the CHn generator output toggles to low and set up *CMPDIF* compare down match interrupt flag. Once the PWM counter underflows it will update the PWM period register PERIODn (new) and duty cycle register CMPn (new) with CNTMODEn = 1.

In Center-aligned type, the PWM period interrupt can also be requested at down-counter underflow if *PINTTYPE* (PWM_CTL2[17]) =0, i.e. at start (end) of each PWM cycle or at up-counter matching with PERIODn if PINTTYPE (PWM_CTL2[17]) =1, i.e. at center point of PWM cycle.

*PWM frequency = HCLK / ((CLKPSCnm + 1) * (clock divider)) / (PERIOD + 1)*; where nm, could be 01, 23, 45 or 67 depending on the selected PWM channel, clock divider is configured by CLKDIV register.

Duty ratio = (PERIOD - CMP) / (PERIOD + 1)

PERIOD =0: PWM output is always low

When PERIOD!=0, PWM output is as follow:

a. CMP ≥ PERIOD: PWM output is always low

b. CMP < PERIOD: PWM low width = (CMP + 1) * 2 units; PWM high width= (PERIOD - CMP) * 2 units[1]

c. CMP = 0: PWM is always high

Note: 1. Unit = one PWM clock cycle.

Figure 3-44, Figure 3-45, Figure 3-46 and Figure 3-47 show the Center-aligned PWM timing and operation flow.

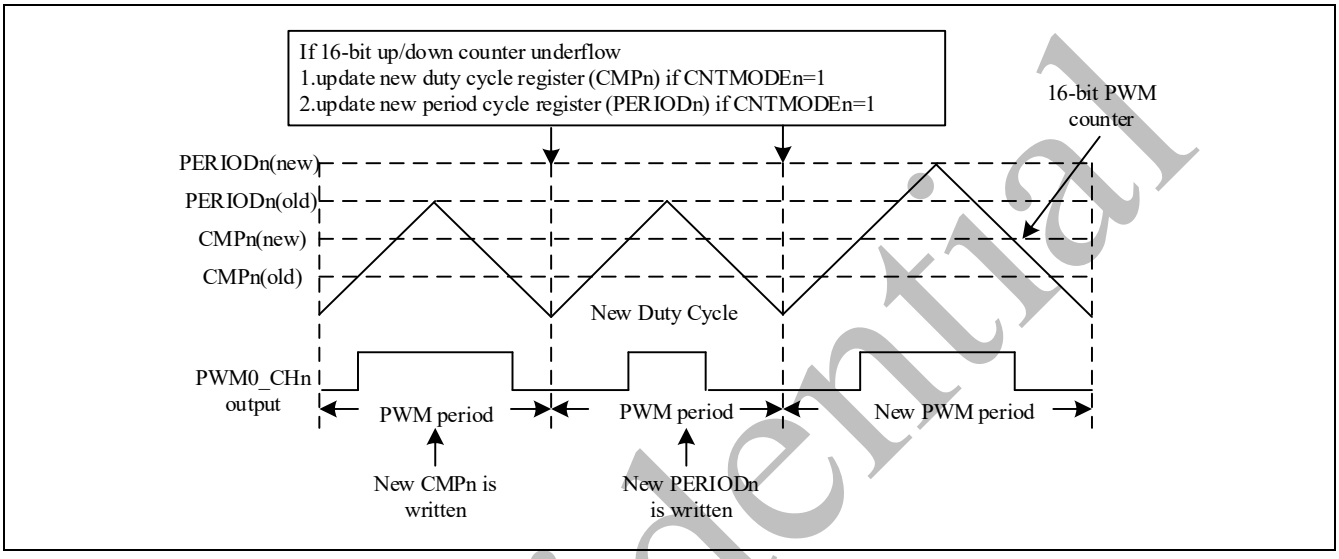


Figure 3-44 Center-aligned Type PWM

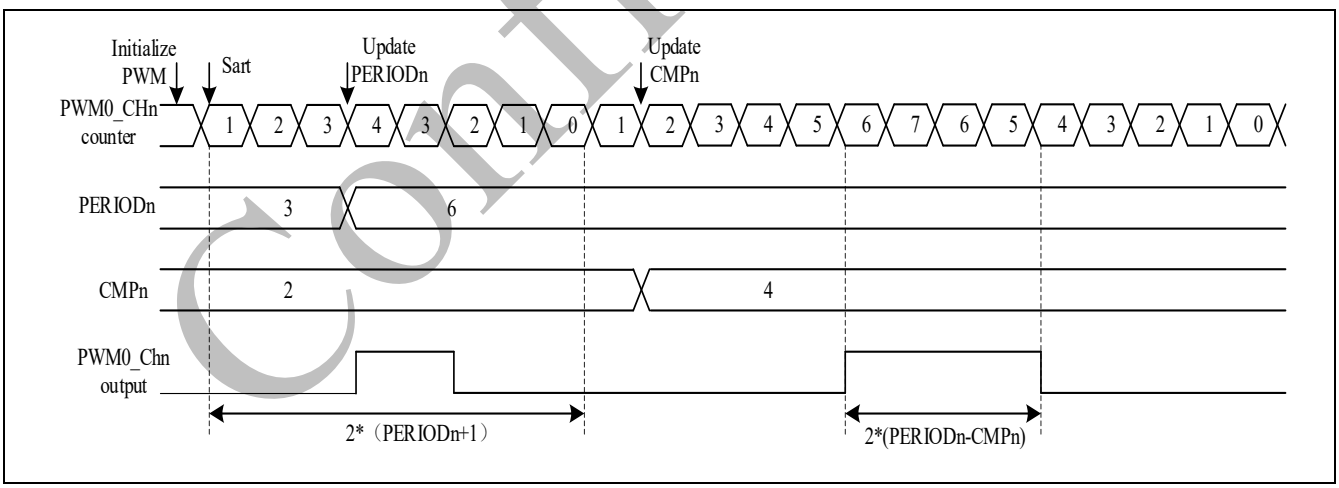


Figure 3-45 Center-aligned Type Operation Timing

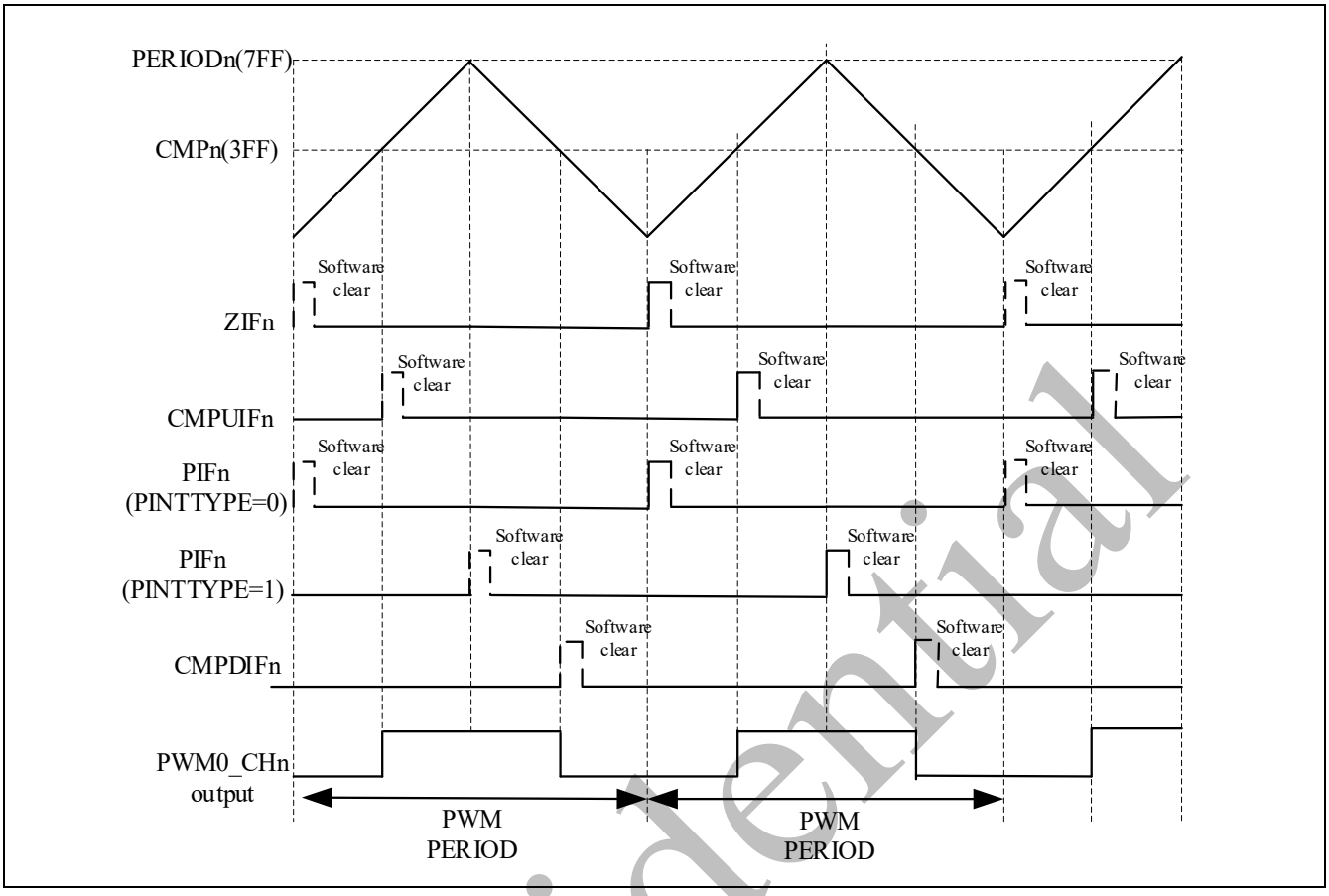


Figure 3-46 PWM Center-aligned Waveform Timing Diagram

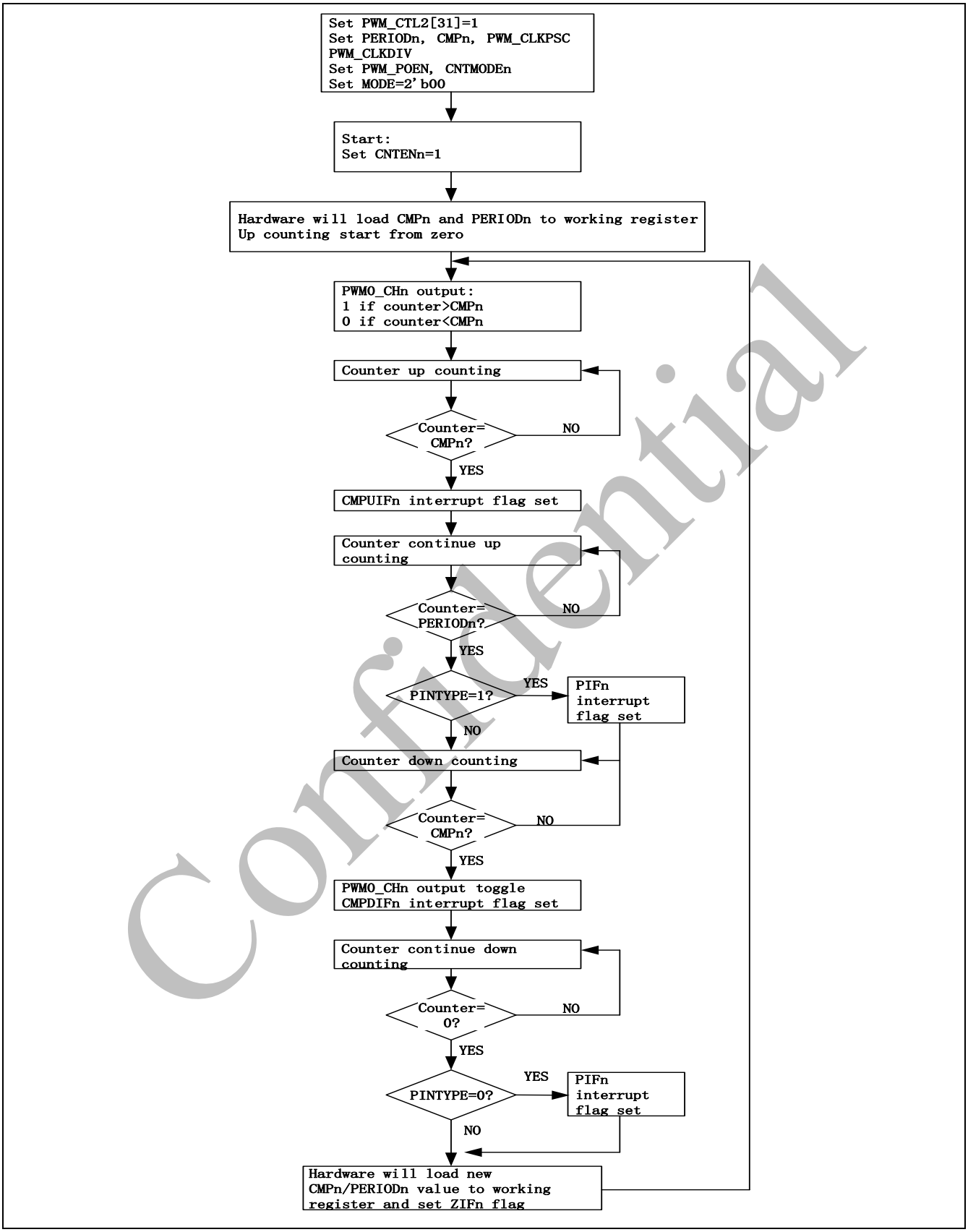


Figure 3-47 Center-aligned Flow Diagram

Precise Center-Aligned PWM (Up/Down Counter)

The precise center-aligned PWM signals are produced by the module when the PWM time base is configured in an Up/Down Counting type and enable *PCAEN* (PWM_PCACTL[0]). The PWM counter will start counting-up from 0 to match the value of CMPn (old); this will cause the toggling of the PWM0_CHn output to high. The counter will continue counting to match with the half of the PERIODn (old). If PERIODn is an odd number, the counter will continue counting to match the integer of lower boundary of the half of the PERIODn (old) and keep the counter value for two clock cycles. Upon reaching this state counter is configured automatically to down counting, when PWM counter matches the CMPn (old) value again the PWM0_CHn output toggles to low. Once the PWM counter underflows it will update the PWM period register PERIODn (new) and duty cycle register CMPn (new) with CNTMODEn = 1.

In Precise Center-aligned type, the PWM period interrupt can also be requested at down-counter underflow if *PINTTYPE* (PWM_CTL2[17]) =0, i.e. at start (end) of each PWM cycle or at up-counter matching with PERIODn if PINTTYPE (PWM_CTL2[17]) =1, i.e. at center point of PWM cycle.

PWM frequency = *HCLK / ((CLKPSCnm + 1) * (clock divider)) / (PERIOD + 1)*; where nm, could be 01, 23, 45 or 67 depending on the selected PWM channel, clock divider is configured by CLKDIV register.

*Duty ratio = (PERIOD – (CMP+1)*2) / PERIOD*

PERIOD =0: PWM output is always low

When PERIOD!=0, PWM output is as follow:

a. CMP ≥ PERIOD: PWM output is always low
b. CMP < PERIOD: PWM low width = (CMP + 1) * 2 units; PWM high width= (PERIOD – (CMP+1)*2) units[1]
c. CMP = 0: PWM is always high

Note: 1. Unit = one PWM clock cycle.

Figure 3-48, Figure 3-49, Figure 3-50 show the Precise Center-aligned PWM timing and operation flow.

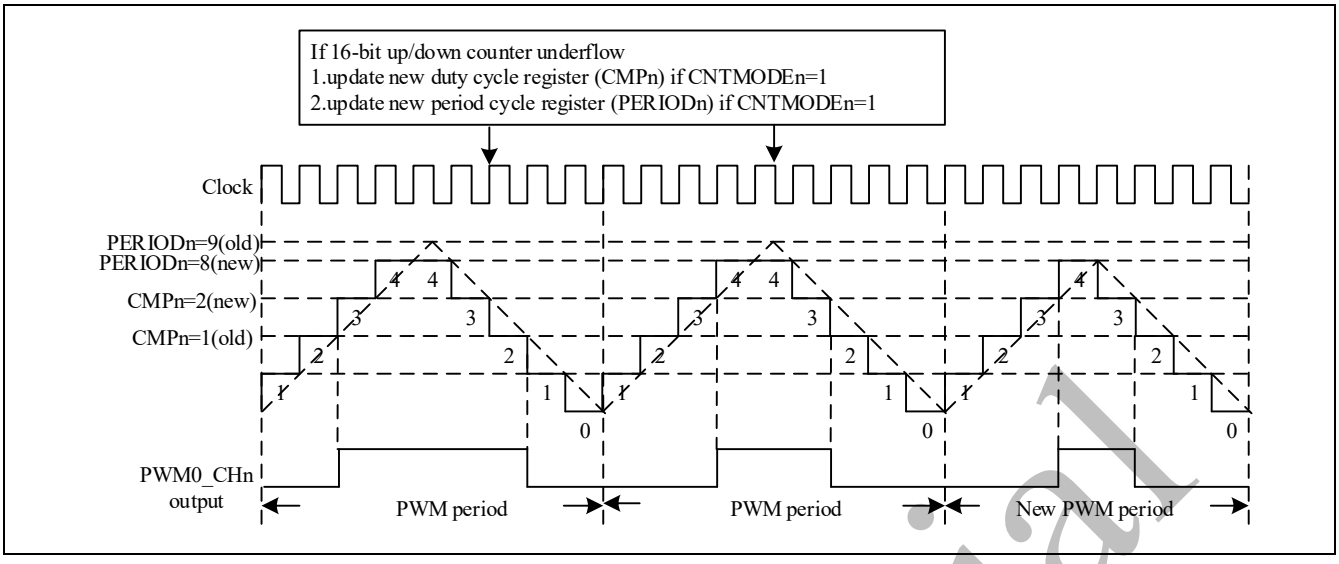


Figure 3-48 Precise Center-aligned Type PWM

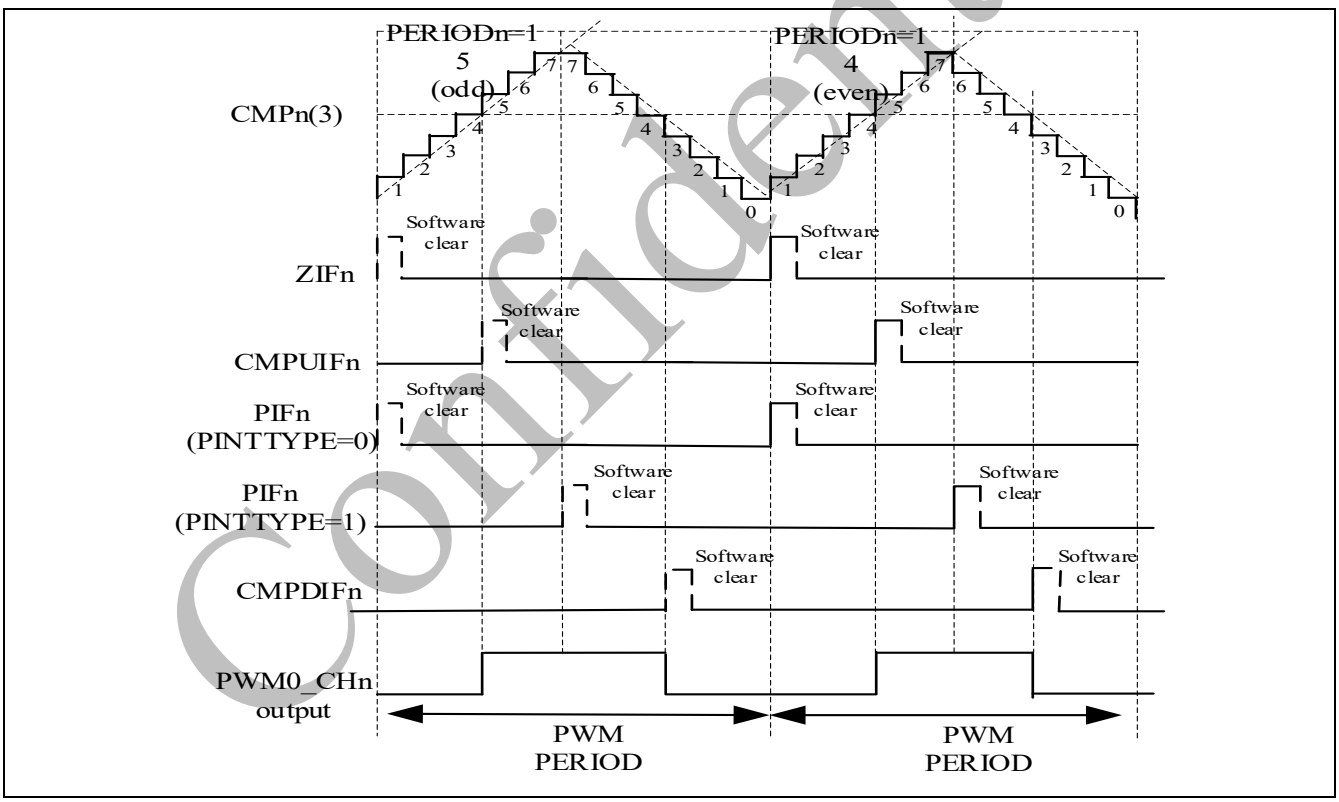


Figure 3-49 PWM Precise Center-aligned Waveform Timing Diagram

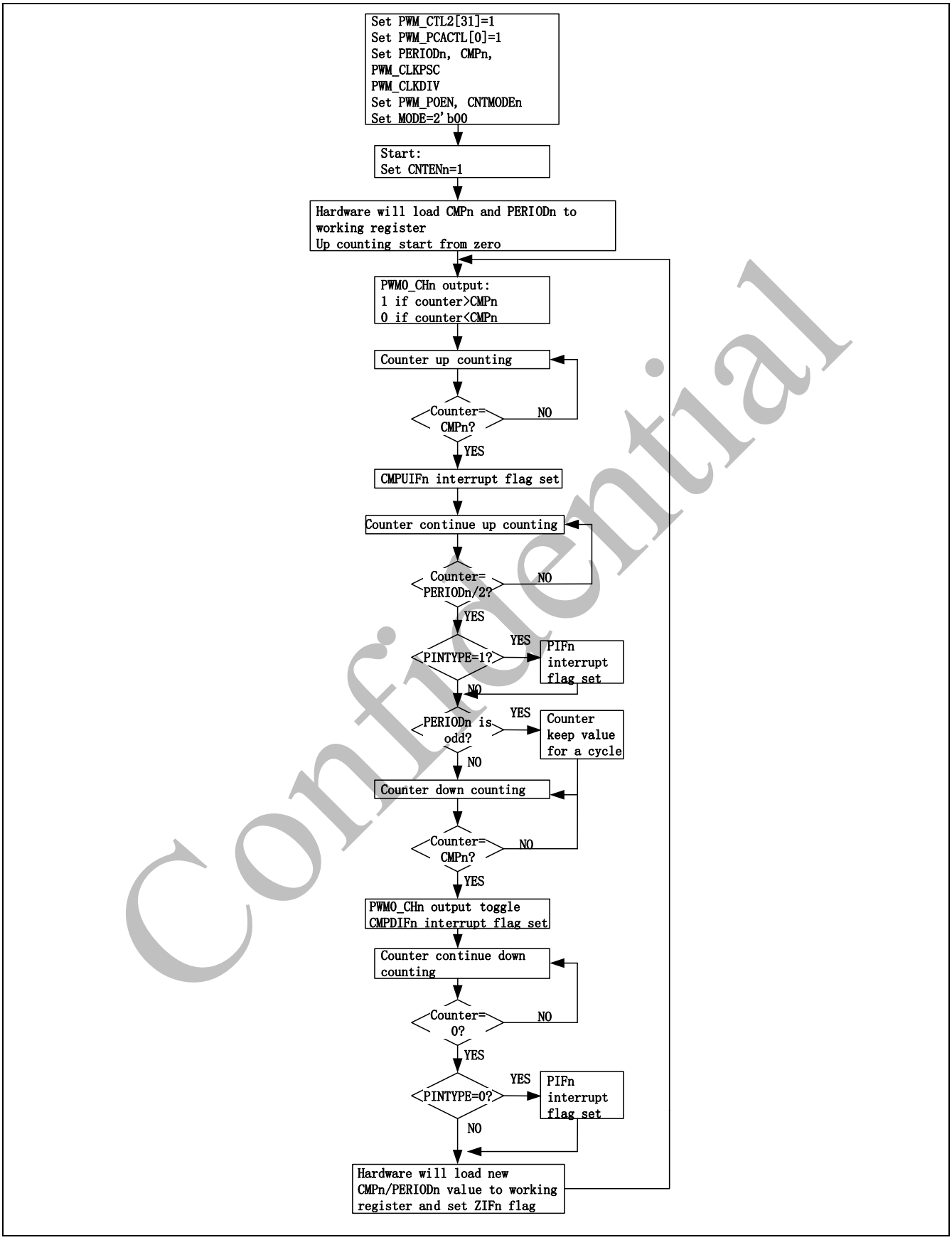


Figure 3-50 Precise Center-aligned Flow Diagram

Asymmetric Mode

Asymmetric mode only works under Center-aligned type. Asymmetric mode is enabled when *ASYMEN* (PWM_CTL[21]) = 1. In this mode PWM counter will compare with another compared value *CMPDn* (PWM_CMPDATn[31:16]) when counting down. If CMRDn is not equal to the CMRn, the PWM will generate asymmetric waveform and set *CMPDIFn* (PWM_INTSTS[13:8]) of the corresponding channel n.

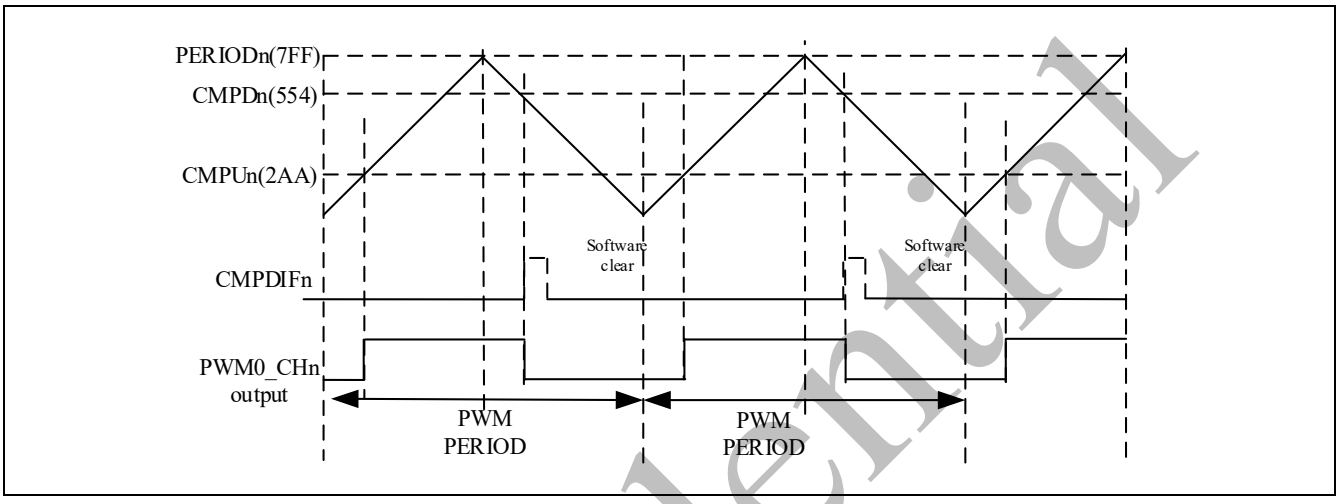


Figure 3-51 Asymmetric Mode Timing Diagram

3.10.5.2 PWM Center Loading Operation

In center-aligned or precise center-aligned type, PWM also supports loading new PERIODn, CMPn. If operating in asymmetric mode, CMPDn will also supports center loading operation.When counter counting to center of PWM period. By setting *HCUPDT* (PWM_CTL[5]) to enable this function. Figure 3-52 shows an example of center loading operation, when counter counts to original center 4; it updates its value to PERIODn 6 then continues counting down.

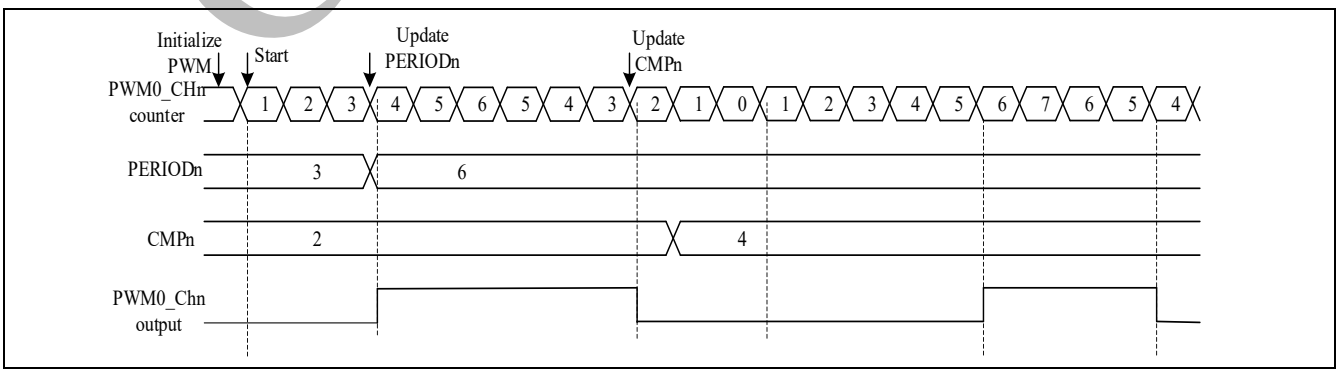


Figure 3-52 PWM Center Loading Timing Diagram

3.10.5.3 PWM Double Buffering and Auto-reload Operation

The PAN101x series PWM have double buffering function, the reload value is updated at the start of next period without affecting current counter operation. The PWM counter value can be written into PERIODn.

CH0 will operate in Auto-reload mode if *CNTMODE0* bit is set to 1. It is recommended that switch CH0 operating mode before set *CNTEN0* bit to 1 to enable CH0 counter to start running because the content of PERIOD0 and CMP0 will be cleared to 0 to reset the CH0 period and duty setting when CH0 operating mode is changed. As CH0 operates at auto-reload mode, CMP0 and PERIOD0 should be written first and then set CNTEN0 bit to 1 to enable CH0 counter to start running. The PERIOD0 value will be reloaded to CH0 counter when the down counting reaches 0. If the PERIOD0 is set to 0, CH0 counter will be held. CH1~CH7 performs the same function as CH0.

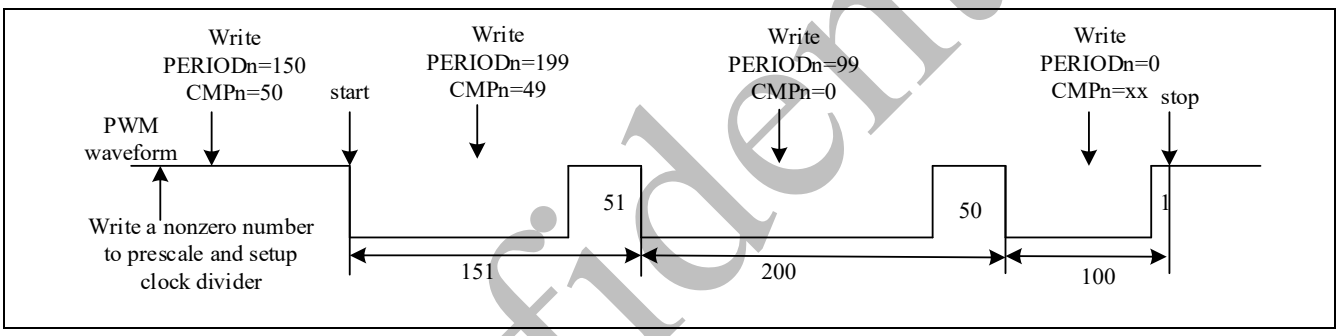


Figure 3-53 PWM Double Buffering Illustration

The double buffering function allows CMPn to be written at any point in the current cycle. The loaded value will take effect from the next cycle which can control output duty ratio easily.

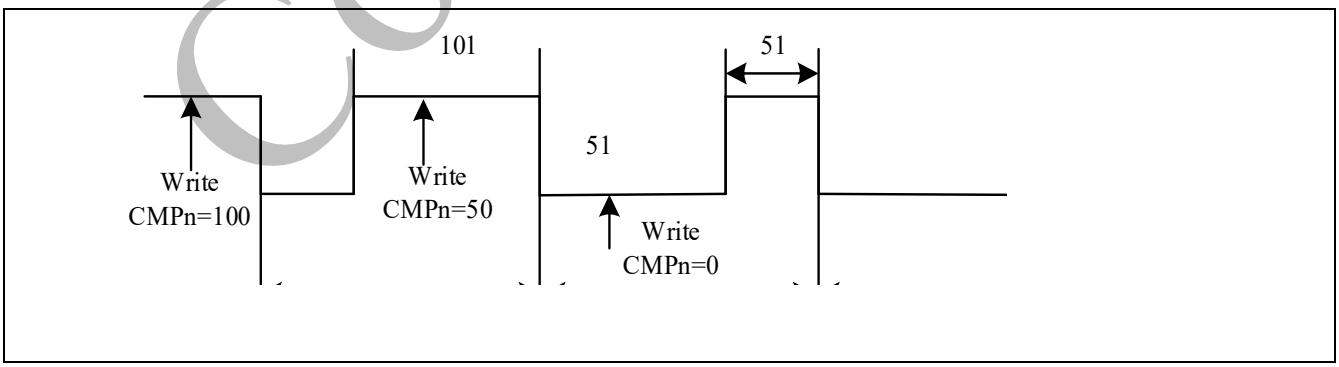


Figure 3-54 PWM Controller Output Duty Ratio

3.10.5.4 PWM Operation Modes

This powerful PWM unit supports independent mode which may be applied to DC or BLDC motor system, Complementary mode with dead-time insertion which may be used in the application of AC induction motor and synchronous motor, and Synchronous mode that makes both pins of each pair are in phase. Besides, the Group mode, which forces the CH2, CH4 and CH6 synchronous with CH0 generator, may simplify updating duty control in DC and BLDC motor applications and Asymmetric mode, to generate asymmetric PWM waveform and interrupt timing.

Independent Mode

Independent mode is enabled when *MODE* (PWM_CTL2[29:28]) = 00.

By default, the PWM is operated in independent mode, with eight PWM channels outputs. Each channel is running off its own duty-cycle generator module.

Complementary Mode

Complementary mode is enabled when MODE (PWM_CTL2[29:28]) = 01.

In this module there are four duty-cycle generators utilized for complementary mode, with total of four PWM output pair pins in this module. The total eight PWM outputs are grouped into output pairs of even and odd numbered outputs. In complimentary modes, the internal odd PWM signal CHn, always be the complement of the corresponding even PWM signal. For example, CH1 will be the complement of CH0. CH3 will be the complement of CH2, CH5 will be the complement of CH4, and CH7 will be the complement of CH6. The time base for the PWM module is provided by its own 16-bit counter, which also incorporates selectable pre-scalar options.

The dead-time generator inserts an “off” period called “dead-time” between the turnings off of one pin to the turning on of the complementary pin of the paired pins. This is to prevent damage to the power switching devices that will be connected to the PWM output pins. The complementary output pair mode has an 8-bit down counter used to produce the dead-time insertion. The complementary outputs are delayed until the counter counts down to zero.

The dead-time can be calculated from the following formula:

*dead-time = PWM_CLK * (DTInm+1)*, where nm, could be 01, 23, 45,67

The timing diagram as shown below indicates the dead-time insertion for one pair of PWM signals.

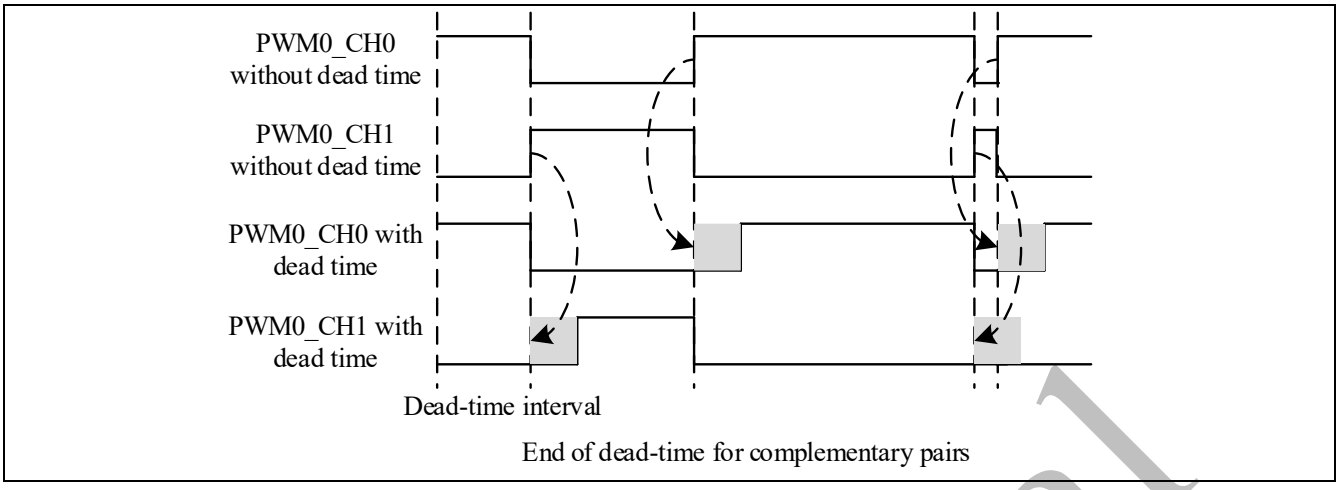


Figure 3-55 Dead-time Insertion

In Power inverter applications, a dead-time insertion avoids the upper and lower switches of the half bridge from being active at the same time. Hence the dead-time control is crucial to proper operation of a system. Some amount of time must be provided between turning off of one PWM output in a complementary pair and turning on the other transistor as the power output devices cannot switch instantaneously.

Synchronous Mode

Synchronous mode is enabled when MODE (PWM_CTL2[29:28]) = 10.

In the synchronization mode the PWM pair signals from PWM Generator are in-phase: CH1=CH0, CH3=CH2, CH5=CH4, and CH7=CH6.

Group Mode

Group mode is enabled when GROUPEN (PWM_CTL2[30]) = 1.

This device supports Group mode control which allows all even PWM channels output to be duty controllable by PWM0_CH0 duty register.

If GROUPEN = 1, All CH2, CH4 and CH6 pairs will follow CH0 output, also CH3, CH5 and CH7 will follow CH1 output. If complementary mode is enabled when MODE (PWM_CTL2[29:28]) = 01, due to CH1 = invert(CH0), so other CHs will follow below formula:

CH6 = CH4 = CH2 = CH0; CH7 = CH5 = CH3 = CH1 = invert (CH0).

Note: For applications, please do not use Group and Synchronous mode simultaneously because the Synchronous mode will be inactive.

3.10.5.5 Polarity Control

Each PWM port from CH0 to CH7 has independent polarity control (PINV0~7) to configure

the polarity of active state of PWM output which are described in the PINVn in *PWM Control Register* (PWM_CTL). By default, the PWM output is active high.

Figure 3-56 shows the initial state before PWM starts with different polarity settings.

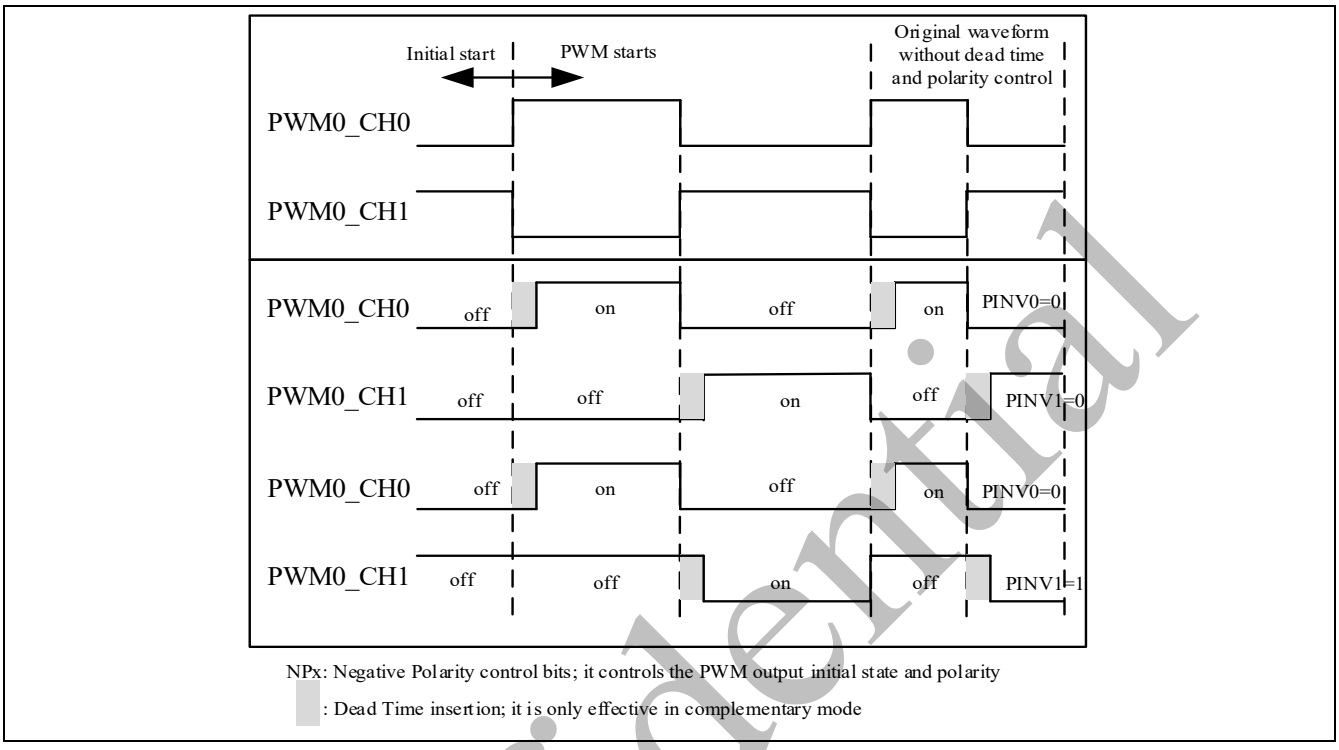


Figure 3-56 Initial State and Polarity Control with Rising Edge Dead-time Insertion

3.10.5.6 PWM Interrupt Architecture

There are four interrupt sources for PWM unit, which are

- ZIFn (PWM_INTSTS[7:0]) PWM counter count to zero interrupt flag;
- CMPUIFn (PWM_INTSTS[31:24]) PWM counter up-counts to CMPn (PWM_CMPDATn[15:0]) interrupt flag;
- PIFn (PWM_INTSTS[23:16]) PWM counter counts to period of edge-aligned type or counts to center of center-aligned type interrupt flag;
- CMPDIFn (PWM_INTSTS[15:8]) PWM counter down-counts to CMPn (PWM_CMPDATn[15:0]) interrupt flag, if operating in asymmetric type it down count to CMPDn (PWM_CMPDATn[31:16]).

The bits *ZIENn (PWM_INTEN[7:0])* control the ZIFn interrupt enable; the bits *CMPUIENn (PWM_INTEN[31:24])* control the CMPUIFn interrupt enable; the bits *PIENn (PWM_INTEN[23:16])* control the PIFn interrupt enable; and the bits *CMPDIENn (PWM_INTEN[15:8])* control the CMPDIFn interrupt enable. Note that all the interrupt flags are set by hardware and must be cleared by software.

Figure 3-57 shows the architecture of Motor Control PWM interrupts.

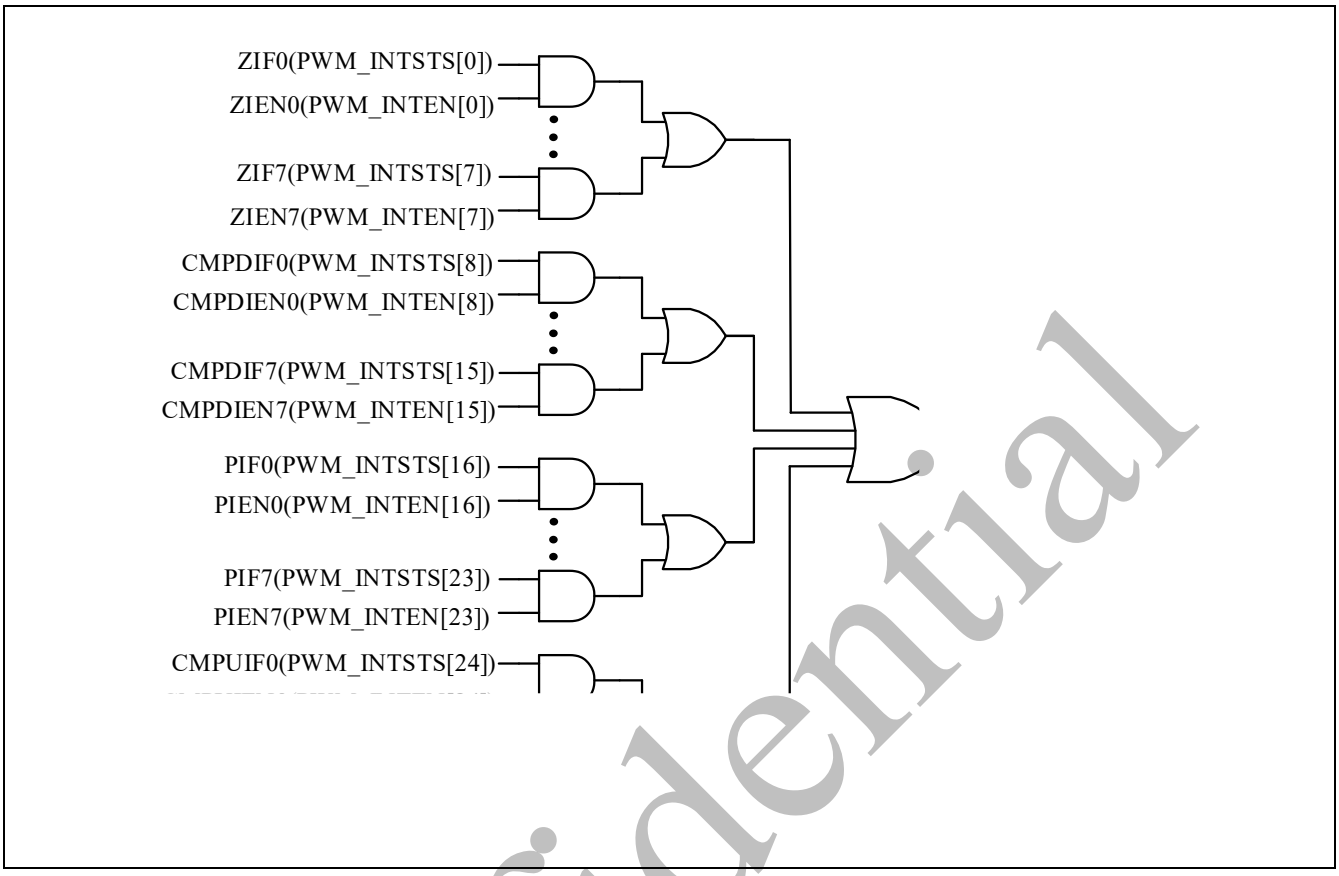


Figure 3-57 Motor Control PWM Interrupt Architecture

3.10.6 PWM Control Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
PWM Base Address: PWM_BA = 0x4004_0000				
PWM_CLKPSC	PWM_BA+0x00	R/W	PWM Clock Pre-scale Register	0x0000_0000
PWM_CLKDIV	PWM_BA+0x04	R/W	PWM Clock Select Register	0x0000_0000
PWM_CTL	PWM_BA+0x08	R/W	PWM Control Register	0x0000_0000
PWM_PERIOD0	PWM_BA+0x0C	R/W	PWM Counter Period Register 0	0x0000_0000
PWM_PERIOD1	PWM_BA+0x10	R/W	PWM Counter Period Register 1	0x0000_0000
PWM_PERIOD2	PWM_BA+0x14	R/W	PWM Counter Period Register 2	0x0000_0000
PWM_PERIOD3	PWM_BA+0x18	R/W	PWM Counter Period Register 3	0x0000_0000
PWM_PERIOD4	PWM_BA+0x1C	R/W	PWM Counter Period Register 4	0x0000_0000
PWM_PERIOD5	PWM_BA+0x20	R/W	PWM Counter Period Register 5	0x0000_0000
PWM_PERIOD6	PWM_BA+0x24	R/W	PWM Counter Period Register 6	0x0000_0000
PWM_PERIOD7	PWM_BA+0x28	R/W	PWM Counter Period Register 7	0x0000_0000
PWM_CMPDAT0	PWM_BA+0x2C	R/W	PWM Comparator Register 0	0x0000_0000
PWM_CMPDAT1	PWM_BA+0x30	R/W	PWM Comparator Register 1	0x0000_0000
PWM_CMPDAT2	PWM_BA+0x34	R/W	PWM Comparator Register 2	0x0000_0000
PWM_CMPDAT3	PWM_BA+0x38	R/W	PWM Comparator Register 3	0x0000_0000
PWM_CMPDAT4	PWM_BA+0x3C	R/W	PWM Comparator Register 4	0x0000_0000
PWM_CMPDAT5	PWM_BA+0x40	R/W	PWM Comparator Register 5	0x0000_0000
PWM_CMPDAT6	PWM_BA+0x44	R/W	PWM Comparator Register 6	0x0000_0000
PWM_CMPDAT7	PWM_BA+0x48	R/W	PWM Comparator Register 7	0x0000_0000
PWM_CTL2	PWM_BA+0x4C	R/W	PWM Control Register	0x0000_0000
PWM_FLAG	PWM_BA+0x50	R/W	PWM Status Register	0x0000_0000
PWM_INTEN	PWM_BA+0x54	R/W	PWM Interrupt Enable Register	0x0000_0000
PWM_INTSTS	PWM_BA+0x58	R/W	PWM Interrupt Status Register	0x0000_0000
PWM_POEN	PWM_BA+0x5C	R/W	PWM Output Enable Register	0x0000_0000
PWM_DTCTL	PWM_BA+0x64	R/W	PWM Dead-time Control Register	0x0000_0000
PWM_ADCTCTL0	PWM_BA+0x68	R/W	PWM Trigger Control Register 0	0x0000_0000
PWM_ADCTCTL1	PWM_BA+0x6C	R/W	PWM Trigger Control Register 1	0x0000_0000
PWM_ADCTSTS0	PWM_BA+0x70	R/W	PWM Trigger Status Register 0	0x0000_0000
PWM_ADCTSTS1	PWM_BA+0x74	R/W	PWM Trigger Status Register 1	0x0000_0000
PWM_PCACTL	PWM_BA+0x88	R/W	PWM Precise Center-Aligned Type Control Register	0x0000_0000

3.10.7 PWM Control Register Description

3.10.7.1 PWM Pre-Scale Register (PWM_CLKPSC)

Register	Offset	R/W	Description	Reset Value
PWM_CLKPSC	PWM_BA+0x00	R/W	PWM Clock Pre-scale Register	0x0000_0000

Bits	Descriptions	
[31:24]	CLKPSC67	Clock Prescaler 6 for PWM Counter 6 and 7 Clock input is divided by (CLKPSC67 + 1) before it is fed to the corresponding PWM counter. If CLKPSC67 = 0, the clock prescaler 6 output clock will be stopped. So the corresponding PWM counter will also be stopped.
[23:16]	CLKPSC45	Clock Prescaler 4 for PWM Counter 4 and 5 Clock input is divided by (CLKPSC45 + 1) before it is fed to the corresponding PWM counter. If CLKPSC45 = 0, the clock prescaler 4 output clock will be stopped. So the corresponding PWM counter will also be stopped.
[15:8]	CLKPSC23	Clock Prescaler 2 for PWM Counter 2 and 3 Clock input is divided by (CLKPSC23 + 1) before it is fed to the corresponding PWM counter. If CLKPSC23 = 0, the clock prescaler 2 output clock will be stopped. So the corresponding PWM counter will also be stopped.
[7:0]	CLKPSC01	Clock Prescaler 0 for PWM Counter 0 and 1 Clock input is divided by (CLKPSC01 + 1) before it is fed to the corresponding PWM counter. If CLKPSC01 = 0, the clock prescaler 0 output clock will be stopped. So the corresponding PWM counter will also be stopped.

3.10.7.2 PWM Clock Selector Register (PWM_CLKDIV)

Register	Offset	R/W	Description	Reset Value
PWM_CLKDIV	PWM_BA+0x04	R/W	PWM Clock Select Register	0x0000_0000

Bits	Descriptions	
[31]	Reserved	Reserved
[30:28]	CLKDIV7	Counter 7 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC67*2). 001 = Clock input / (CLKPSC67*4). 010 = Clock input / (CLKPSC67*8). 011 = Clock input / (CLKPSC67*16). 100 = Clock input / CLKPSC67. Others = Clock input.
[27]	Reserved	Reserved
[26:24]	CLKDIV6	Counter 6 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC67*2). 001 = Clock input / (CLKPSC67*4). 010 = Clock input / (CLKPSC67*8). 011 = Clock input / (CLKPSC67*16). 100 = Clock input / CLKPSC67. Others = Clock input.
[23]	Reserved	Reserved
[22:20]	CLKDIV5	Counter 5 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC45*2). 001 = Clock input / (CLKPSC45*4). 010 = Clock input / (CLKPSC45*8). 011 = Clock input / (CLKPSC45*16). 100 = Clock input / CLKPSC45. Others = Clock input.
[19]	Reserved	Reserved
[18:16]	CLKDIV4	Counter 4 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC45*2). 001 = Clock input / (CLKPSC45*4). 010 = Clock input / (CLKPSC45*8). 011 = Clock input / (CLKPSC45*16). 100 = Clock input / CLKPSC45. Others = Clock input.
[15]	Reserved	Reserved
[14:12]	CLKDIV3	Counter 3 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC23*2). 001 = Clock input / (CLKPSC23*4). 010 = Clock input / (CLKPSC23*8).

		011 = Clock input / (CLKPSC23*16). 100 = Clock input / CLKPSC23. Others = Clock input.
[11]	Reserved	Reserved
[10:8]	CLKDIV2	Counter 2 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC23*2). 001 = Clock input / (CLKPSC23*4). 010 = Clock input / (CLKPSC23*8). 011 = Clock input / (CLKPSC23*16). 100 = Clock input / CLKPSC23. Others = Clock input.
[7]	Reserved	Reserved
[6:4]	CLKDIV1	Counter 1 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC01*2). 001 = Clock input / (CLKPSC01*4). 010 = Clock input / (CLKPSC01*8). 011 = Clock input / (CLKPSC01*16). 100 = Clock input / CLKPSC01. Others = Clock input.
[3]	Reserved	Reserved
[2:0]	CLKDIV0	Counter 0 Clock Divider Selection Select clock input for PWM counter. 000 = Clock input / (CLKPSC01*2). 001 = Clock input / (CLKPSC01*4). 010 = Clock input / (CLKPSC01*8). 011 = Clock input / (CLKPSC01*16). 100 = Clock input / CLKPSC01. Others = Clock input.

3.10.7.3 PWM Control Register (PWM_CTL)

Register	Offset	R/W	Description	Reset Value
PWM_CTL	PWM_BA+0x08	R/W	PWM Control Register	0x0000_0000

Bits	Descriptions	
[31]	Reserved	Reserved
[30]	PINV7	PWM0_CH7 Output Inverter Enable Bit 0 = PWM0_CH7 output inverter Disabled. 1 = PWM0_CH7 output inverter Enabled.
[29]	Reserved	Reserved
[28]	CNTEN7	PWM Counter 7 Enable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.
[27]	Reserved	Reserved
[26]	PINV6	PWM0_CH6 Output Inverter Enable Bit 0 = PWM0_CH6 output inverter Disabled. 1 = PWM0_CH6 output inverter Enabled.
[25]	Reserved	Reserved
[24]	CNTEN6	PWM Counter 6 Enable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.
[23]	Reserved	Reserved
[22]	PINV5	PWM0_CH5 Output Inverter Enable Bit 0 = PWM0_CH5 output inverter Disabled. 1 = PWM0_CH5 output inverter Enabled.
[21]	ASYMEN	Asymmetric Mode In Center-aligned Type 0 = Symmetric mode in center-aligned type. 1 = Asymmetric mode in center-aligned type.
[20]	CNTEN5	PWM Counter 5 Enable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.
[19]	Reserved	Reserved
[18]	PINV4	PWM0_CH4 Output Inverter Enable Bit 0 = PWM0_CH4 output inverter Disabled. 1 = PWM0_CH4 output inverter Enabled.
[17]	Reserved	Reserved
[16]	CNTEN4	PWM Counter 4 Enable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.
[15]	Reserved	Reserved
[14]	PINV3	PWM0_CH 3 Output Inverter Enable Bit 0 = PWM0_CH3 output inverter Disabled. 1 = PWM0_CH3 output inverter Enabled.
[13]	Reserved	Reserved
[12]	CNTEN3	PWM Counter 3 Enable Start Run 0 = Corresponding PWM counter running Stopped.

		1 = Corresponding PWM counter start run Enabled.
[11]	Reserved	Reserved
[10]	PINV2	PWM0_CH2 Output Inverter Enable Bit 0 = PWM0_CH2 output inverter Disabled. 1 = PWM0_CH2 output inverter Enabled.
[9]	Reserved	Reserved
[8]	CNTEN2	PWM Counter 2 Enable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.
[7]	Reserved	Reserved
[6]	PINV1	PWM0_CH1 Output Inverter Enable Bit 0 = PWM0_CH1 output inverter Disable. 1 = PWM0_CH1 output inverter Enable.
[5]	HCUPDT	Half Cycle Update Enable for Center-aligned Type 0 = Disable half cycle update PERIOD & CMP. 1 = Enable half cycle update PERIOD & CMP.
[4]	CNTEN1	PWM Counter 1 Enable/Disable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.
[3]	Reserved	Reserved
[2]	PINV0	PWM0_CH0 Output Inverter Enable Bit 0 = PWM0_CH0 output inverter Disabled. 1 = PWM0_CH0 output inverter Enabled.
[1]	Reserved	Reserved
[0]	CNTEN0	PWM Counter 0 Enable Start Run 0 = Corresponding PWM counter running Stopped. 1 = Corresponding PWM counter start run Enabled.

3.10.7.4 PWM Counter Register 0-7 (PWM_PERIOD0-7)

Register	Offset	R/W	Description	Reset Value
PWM_PERIOD0	PWM_BA+0x0C	R/W	PWM Counter Period Register 0	0x0000_0000
PWM_PERIOD1	PWM_BA+0x10	R/W	PWM Counter Period Register 1	0x0000_0000
PWM_PERIOD2	PWM_BA+0x14	R/W	PWM Counter Period Register 2	0x0000_0000
PWM_PERIOD3	PWM_BA+0x18	R/W	PWM Counter Period Register 3	0x0000_0000
PWM_PERIOD4	PWM_BA+0x1C	R/W	PWM Counter Period Register 4	0x0000_0000
PWM_PERIOD5	PWM_BA+0x20	R/W	PWM Counter Period Register 5	0x0000_0000
PWM_PERIOD6	PWM_BA+0x24	R/W	PWM Counter Period Register 6	0x0000_0000
PWM_PERIOD7	PWM_BA+0x28	R/W	PWM Counter Period Register 7	0x0000_0000

Bits	Descriptions	
[31:16]	Reserved	Reserved
[15:0] n=0,1..7	PERIODn	PWM Counter Period Value PERIODn determines the PWM counter period. **Edge-aligned type**: *PWM frequency = HCLK/((prescale+1)*(clock divider))/(PERIODn+1)*; where xy, could be 01, 23, 45, 67 depending on the selected PWM channel. *Duty ratio = (CMPn+1)/(PERIODn+1).* PERIOD = 0: PWM is always low. When PERIOD!=0, PWM output is as follow: CMPn ≥ PERIODn: PWM output is always high. CMPn < PERIODn: PWM low width = (PERIODn-CMPn) unit; PWM high width = (CMP+1) unit. CMPn = 0: PWM is always low. **Center-aligned type**: PWM frequency = HCLK/((prescale+1)*(clock divider))/ (2*PERIODn+1); where xy, could be 01, 23, 45, 67 depending on the selected PWM channel. *Duty ratio = (PERIODn - CMPn)/(PERIODn+1).* PERIOD = 0: PWM is always low. When PERIOD!=0, PWM output is as follow: CMPn ≥ PERIODn: PWM output is always low. CMPn < PERIODn: PWM low width = (CMPn + 1) * 2 unit; PWM high width = (PERIODn - CMPn) * 2 unit. CMPn = 0: PWM is always high. (Unit = One PWM clock cycle). **Note**: Any write to PERIODn will take effect in the next PWM cycle.

3.10.7.5 PWM Comparator Register 0-7 (PWM_CMPDAT0-7)

Register	Offset	R/W	Description	Reset Value
PWM_CMPDAT0	PWM_BA+0x2C	R/W	PWM Comparator Register 0	0x0000_0000
PWM_CMPDAT1	PWM_BA+0x30	R/W	PWM Comparator Register 1	0x0000_0000
PWM_CMPDAT2	PWM_BA+0x34	R/W	PWM Comparator Register 2	0x0000_0000
PWM_CMPDAT3	PWM_BA+0x38	R/W	PWM Comparator Register 3	0x0000_0000
PWM_CMPDAT4	PWM_BA+0x3C	R/W	PWM Comparator Register 4	0x0000_0000
PWM_CMPDAT5	PWM_BA+0x40	R/W	PWM Comparator Register 5	0x0000_0000
PWM_CMPDAT6	PWM_BA+0x44	R/W	PWM Comparator Register 6	0x0000_0000
PWM_CMPDAT7	PWM_BA+0x48	R/W	PWM Comparator Register 7	0x0000_0000

Bits	Descriptions	
[31:16] n=0,1..7	CMPDn	PWM Comparator Register for Down Counter In Asymmetric Mode CMPn ≥ PERIODn: up counter PWM output is always low. CMPDn ≥ PERIODn: down counter PWM output is always low. Others: PWM output is always high.
[15:0] n=0,1..7	CMPn	PWM Comparator Register CMP determines the PWM duty. Edge-aligned type: PWM frequency = HCLK/((CLKPSCnm+1)*(clock divider))/(PERIODn+1); where nm, could be 01, 23, 45, 67 depending on the selected PWM channel. *Duty ratio = (CMPn+1)/(PERIODn+1).* PERIOD = 0: PWM is always low. When PERIOD!=0, PWM output is as follow: CMPn ≥ PERIODn: PWM output is always high. CMPn < PERIODn: PWM low width = (PERIODn-CMPn) unit; PWM high width = (CMP+1) unit. CMPn = 0: PWM is always low. Center-aligned type: PWM frequency = HCLK/((prescale+1)*(clock divider))/ (2*PERIODn+1); where xy, could be 01, 23, 45, 67 depending on the selected PWM channel. *Duty ratio = (PERIODn - CMPn)/(PERIODn+1).* PERIOD = 0: PWM is always low. When PERIOD!=0, PWM output is as follow: CMPn ≥ PERIODn: PWM output is always low. CMPn < PERIODn: PWM low width = (CMPn + 1) * 2 unit; PWM high width = (PERIODn - CMPn) * 2 unit. CMPn = 0: PWM is always high. (Unit = One PWM clock cycle). **Note**: Any write to CMPn will take effect in the next PWM cycle.

3.10.7.6 PWM Control Register2 (PWM_CTL2)

Register	Offset	R/W	Description	Reset Value
PWM_CTL2	PWM_BA+0x4C	R/W	PWM Control Register	0x0000_0000

Bits	Descriptions	
[31]	CNTTYPE	PWM Counter-aligned Type Select Bit 0 = Edge-aligned type. 1 = Center-aligned type.
[30]	GROUPEN	Group Function Enable Bit 0 = The signals timing of all PWM channels are independent. 1 = Unify the signals timing of PWM0_CH0, PWM0_CH2, PWM0_CH4 and PWM0_CH6 in the same phase which is controlled by PWM0_CH0 and also unify the signals timing of PWM0_CH1, PWM0_CH3, PWM0_CH5 and PWM0_CH7 in the same phase which is controlled by PWM0_CH1.
[29:28]	MODE	PWM Operating Mode Select Bit 00 = Independent mode. 01 = Complementary mode. 10 = Synchronized mode. 11 = Reserved.
[27]	DTCNT67	Dead-time 6 Counter Enable Bit (PWM0_CH6 and PWM0_CH7 Pair for PWMC Group) 0 = Dead-time 6 generator Disabled. 1 = Dead-time 6 generator Enabled. **Note**: When the dead-time generator is enabled, the pair of PWM0_CH6 and PWM0_CH7 becomes a complementary pair for PWMC group.
[26]	DTCNT45	Dead-time 4 Counter Enable Bit (PWM0_CH4 and PWM0_CH5 Pair for PWMC Group) 0 = Dead-time 4 generator Disabled. 1 = Dead-time 4 generator Enabled. **Note**: When the dead-time generator is enabled, the pair of PWM0_CH4 and PWM0_CH5 becomes a complementary pair for PWMC group.
[25]	DTCNT23	Dead-time 2 Counter Enable Bit (PWM0_CH2 and PWM0_CH3 Pair for PWMB Group) 0 = Dead-time 2 generator Disabled. 1 = Dead-time 2 generator Enabled. **Note**: When the dead-time generator is enabled, the pair of PWM0_CH2 and PWM0_CH3 becomes a complementary pair for PWMB group.
[24]	DTCNT01	Dead-time 0 Counter Enable Bit (PWM0_CH0 and PWM0_CH1 Pair for PWMA Group) 0 = Dead-time 0 generator Disabled. 1 = Dead-time 0 generator Enabled. **Note**: When the dead-time generator is enabled, the pair of PWM0_CH0 and PWM0_CH1 becomes a complementary pair for PWMA group.
[23:18]	Reserved	Reserved
[17]	PINTTYPE	PWM Interrupt Type Selection 0 = ZIFn will be set if PWM counter underflows. 1 = ZIFn will be set if PWM counter matches PERIODn register.

		Note: This bit is effective when PWM is in center-aligned type only.
[16:0]	Reserved	Reserved

3.10.7.7 PWM Flag Indication Register (PWM_FLAG)

Register	Offset	R/W	Description	Reset Value
PWM_FLAG	PWM_BA+0x50	R/W	PWM Status Flag Register	0x0000_0000

Bits	Descriptions	
[31:24] n=0,1..7	CMPUFn	PWM Compare Up Flag Flag is set by hardware when PWM0_CHn counter up count reaches CMPn. **Note**: This bit can be cleared by software writing 1.
[23:16] n=0,1..7	PFn	PWM Period Flag Flag is set by hardware when PWM0_CHn counter reaches PERIODn. **Note**: This bit can be cleared by software writing 1.
[15:8] n=0,1..7	CMPDFn	PWM Compare Down Flag Flag is set by hardware when PWMn counter down count reaches CMPn. **Note**: This bit can be cleared by software writing 1.
[7:0] n=0,1..7	ZFn	PWM Zero Point Flag Flag is set by hardware when PWMn counter down count reaches zero point. **Note**: This bit can be cleared by software writing 1.

3.10.7.8 PWM Interrupt Enable Register (PWM_INTEN)

Register	Offset	R/W	Description	Reset Value
PWM_INTEN	PWM_BA+0x54	R/W	PWM Interrupt Enable Register	0x0000_0000

Bits	Descriptions	
[31:24] n=0,1..7	CMPUIENn	PWM Compare Up Interrupt Enable Bit 0 = PWM0_CHn compare up interrupt Disabled. 1 = PWM0_CHn compare up interrupt Enabled.
[23:16] n=0,1..7	PIENn	PWM Period Interrupt Enable Bit 0 = PWM0_CHn period interrupt Disabled. 1 = PWM0_CHn period interrupt Enabled.
[15:8] n=0,1..7	CMPDIENn	PWM Compare Down Interrupt Enable Bit 0 = PWM0_CHn compare down interrupt Disabled. 1 = PWM0_CHn compare down interrupt Enabled.
[7:0] n=0,1..7	ZIENn	PWM Zero Point Interrupt Enable Bit 0 = PWM0_CHn zero point interrupt Disabled. 1 = PWM0_CHn zero point interrupt Enabled.

3.10.7.9 PWM Interrupt Indication Register (PWM_INTSTS)

Register	Offset	R/W	Description	Reset Value
PWM_INTSTS	PWM_BA+0x58	R/W	PWM Interrupt Status Register	0x0000_0000

Bits	Descriptions	
[31:24] n=0,1..7	CMPUIFn	PWM Compare Up Interrupt Flag Flag is set by hardware when PWM0_CHn counter up count reaches CMPn. **Note**: This bit can be cleared by software writing 1.
[23:16] n=0,1..7	PIFn	PWM Period Interrupt Flag Flag is set by hardware when PWM0_CHn counter reaches PERIODn. **Note**: This bit can be cleared by software writing 1.
[15:8] n=0,1..7	CMPDIFn	PWM Compare Down Interrupt Flag Flag is set by hardware when PWMn counter down count reaches CMPn. **Note**: This bit can be cleared by software writing 1.
[7:0] n=0,1..7	ZIFn	PWM Zero Point Interrupt Flag Flag is set by hardware when PWMn counter down count reaches zero point. **Note**: This bit can be cleared by software writing 1.

3.10.7.10 PWM Output Control Register (PWM_POEN)

Register	Offset	R/W	Description	Reset Value
PWM_POEN	PWM_BA+0x5C	R/W	PWM Output Enable Register	0x0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved
[7:0] n=0,1..7	POENn	PWM Output Enable Bits 0 = PWM channel n output to pin Disabled. 1 = PWM channel n output to pin Enabled. **Note**: The corresponding GPIO pin must be switched to PWM function.

3.10.7.11 PWM Dead-time Interval Register (PWM_DTCTL)

Register	Offset	R/W	Description	Reset Value
PWM_DTCTL	PWM_BA+0x64	R/W	PWM Dead-time Control Register	0x0000_0000

Bits	Descriptions	
[31:24]	DTI67	Dead-time Interval Register for Pair Of Channel6 and Channel7 (PWM0_CH6 and PWM0_CH7 Pair) These 8 bits determine dead-time length. The unit time of dead-time length is received from corresponding PWM_CLKDIV bits.
[23:16]	DTI45	Dead-time Interval Register for Pair Of Channel4 and Channel5 (PWM0_CH4 and PWM0_CH5 Pair) These 8 bits determine dead-time length. The unit time of dead-time length is received from corresponding PWM_CLKDIV bits.
[15:8]	DTI23	Dead-time Interval Register for Pair Of Channel2 and Channel3 (PWM0_CH2 and PWM0_CH3 Pair) These 8 bits determine dead-time length. The unit time of dead-time length is received from corresponding PWM_CLKDIV bits.
[7:0]	DTI01	Dead-time Interval Register for Pair Of Channel0 and Channel1 (PWM0_CH0 and PWM0_CH1 Pair) These 8 bits determine dead-time length. The unit time of dead-time length is received from corresponding PWM_CLKDIV bits.

3.10.7.12 PWM Trigger ADC Control Register (PWM_ADCTCTL0)

Register	Offset	R/W	Description	Reset Value
PWM_ADCTCTL0	PWM_BA+0x68	R/W	PWM Trigger Control Register 0	0x0000_0000

Bits	Descriptions	
[31:28]	Reserved	Reserved
[27]	ZPTRGEN3	Channel 3 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel3's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[26]	CDTRGEN3	Channel 3 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel3's counter matching CMP3 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[25]	CPTRGEN3	Channel 3 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel3's Counter Matching PERIOD3 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[24]	CUTRGEN3	Channel 3 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel3's counter matching CMP3 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[23:20]	Reserved	Reserved
[19]	ZPTRGEN2	Channel 2 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel2's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[18]	CDTRGEN2	Channel 2 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel2's counter matching CMP2 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[17]	CPTRGEN2	Channel 2 Center Point Trigger ADC Enable Bit

		Enable PWM Trigger ADC Function While channel2's Counter Matching PERIOD2 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[16]	CUTRGEN2	Channel 2 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel2's counter matching CMP2 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[15:12]	Reserved	Reserved
[11]	ZPTRGEN1	Channel 1 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel1's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[10]	CDTRGEN1	Channel 1 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel1's counter matching CMP1 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[9]	CPTRGEN1	Channel 1 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel0's Counter Matching PERIOD1 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[8]	CUTRGEN1	Channel 1 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel1's counter matching CMP1 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[7:4]	Reserved	Reserved
[3]	ZPTRGEN0	Channel 0 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel0's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled.

		Note: This bit is valid for both center-aligned type and edged-aligned type.
[2]	CDTRGEN0	Channel 0 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel0's counter matching CMP0 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[1]	CPTRGEN0	Channel 0 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel0's Counter Matching PERIOD0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note:** This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[0]	CUTRGEN0	Channel 0 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel0's counter matching CMP0 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note:** This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.

3.10.7.13 PWM Trigger ADC Control Register (PWM_ADCTCTL1)

Register	Offset	R/W	Description	Reset Value
PWM_ADCTCTL1	PWM_BA+0x6C	R/W	PWM Trigger Control Register 0	0x0000_0000

Bits	Descriptions	
[31:28]	Reserved	Reserved
[27]	ZPTRGEN7	Channel 7 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel7's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[26]	CDTRGEN7	Channel 7 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel7's counter matching CMP7 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[25]	CPTRGEN7	Channel 7 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel7's Counter Matching PERIOD7 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[24]	CUTRGEN7	Channel 7 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel7's counter matching CMP7 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[23:20]	Reserved	Reserved
[19]	ZPTRGEN6	Channel 6 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel6's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[18]	CDTRGEN6	Channel 6 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel6's counter matching CMP6 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[17]	CPTRGEN6	Channel 6 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel6's Counter Matching

		PERIOD6 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[16]	CUTRGEN6	Channel 6 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel6's counter matching CMP6 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[15:12]	Reserved	Reserved
[11]	ZPTRGEN5	Channel 5 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel5's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[10]	CDTRGEN5	Channel 5 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel5's counter matching CMP5 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[9]	CPTRGEN5	Channel 5 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel5's Counter Matching PERIOD5 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[8]	CUTRGEN5	Channel 5 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel5's counter matching CMP5 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[7:4]	Reserved	Reserved
[3]	ZPTRGEN4	Channel 4 Zero Point Trigger ADC Enable Bit Enable PWM trigger ADC function while channel4's counter matching 0 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.

[2]	CDTRGEN4	Channel 4 Compare Down Trigger ADC Enable Bit Enable PWM trigger ADC function while channel4's counter matching CMP4 in down-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is valid for both center-aligned type and edged-aligned type.
[1]	CPTRGEN4	Channel 4 Center Point Trigger ADC Enable Bit Enable PWM Trigger ADC Function While channel4's Counter Matching PERIOD4 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.
[0]	CUTRGEN4	Channel 4 Compare Up Trigger ADC Enable Bit Enable PWM trigger ADC function while channel4's counter matching CMP4 in up-count direction 0 = PWM condition trigger ADC function Disabled. 1 = PWM condition trigger ADC function Enabled. **Note**: This bit is only valid for PWM in center-aligned type. When PWM is in edged-aligned type, setting this bit is meaningless and will not take any effect.

3.10.7.14 PWM Trigger Status Register (PWM_ADCTSTS0)

Register	Offset	R/W	Description	Reset Value
PWM_ADCTSTS0	PWM_BA+0x70	R/W	PWM Trigger Status Register 0	0x0000_0000

Bits	Descriptions	
[31:28]	Reserved	Reserved
[27]	ZPTRGF3	Channel 3 Zero Point Trigger ADC Flag When the channel3's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[26]	CDTRGF3	Channel 3 Compare Down Trigger ADC Flag When the channel3's counter is counting down to CMP3, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[25]	CPTRGF3	Channel 3 Center Point Trigger ADC Flag When the channel3's counter is counting to PERIOD3, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[24]	CUTRGF3	Channel 3 Compare Up Trigger ADC Flag When the channel3's counter is counting up to CMP3, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[23:20]	Reserved	Reserved
[19]	ZPTRGF2	Channel 2 Zero Point Trigger ADC Flag When the channel2's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[18]	CDTRGF2	Channel 2 Compare Down Trigger ADC Flag When the channel2's counter is counting down to CMP2, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[17]	CPTRGF2	Channel 2 Center Point Trigger ADC Flag When the channel2's counter is counting to PERIOD2, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[16]	CUTRGF2	Channel 2 Compare Up Trigger ADC Flag When the channel2's counter is counting up to CMP2, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[15:12]	Reserved	Reserved
[11]	ZPTRGF1	Channel 1 Zero Point Trigger ADC Flag When the channel1's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[10]	CDTRGF1	Channel 1 Compare Down Trigger ADC Flag When the channel1's counter is counting down to CMP1, this bit will be set for

		trigger ADC. **Note**: This bit can be cleared by software writing 1.
[9]	CPTRGF1	Channel 1 Center Point Trigger ADC Flag When the channel1's counter is counting to PERIOD1, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[8]	CUTRGF1	Channel 1 Compare Up Trigger ADC Flag When the channel1's counter is counting up to CMP1, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[7:4]	Reserved	Reserved
[3]	ZPTRGF0	Channel 0 Zero Point Trigger ADC Flag When the channel0's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[2]	CDTRGF0	Channel 0 Compare Down Trigger ADC Flag When the channel0's counter is counting down to CMP0, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[1]	CPTRGF0	Channel 0 Center Point Trigger ADC Flag When the channel0's counter is counting to PERIOD0, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[0]	CUTRGF0	Channel 0 Compare Up Trigger ADC Flag When the channel0's counter is counting up to CMP0, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.

3.10.7.15 PWM Trigger Status Register (PWM_ADCTSTS1)

Register	Offset	R/W	Description	Reset Value
PWM_ADCTSTS1	PWM_BA+0x74	R/W	PWM Trigger Status Register 1	0x0000_0000

Bits	Descriptions	
[31:28]	Reserved	Reserved
[27]	ZPTRGF7	Channel 7 Zero Point Trigger ADC Flag When the channel7's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[26]	CDTRGF7	Channel 7 Compare Down Trigger ADC Flag When the channel7's counter is counting down to CMP7, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[25]	CPTRGF7	Channel 7 Center Point Trigger ADC Flag When the channel7's counter is counting to PERIOD7, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[24]	CUTRGF7	Channel 7 Compare Up Trigger ADC Flag When the channel7's counter is counting up to CMP7, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[23:20]	Reserved	Reserved
[19]	ZPTRGF6	Channel 6 Zero Point Trigger ADC Flag When the channel6's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[18]	CDTRGF6	Channel 6 Compare Down Trigger ADC Flag When the channel6's counter is counting down to CMP6, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[17]	CPTRGF6	Channel 6 Center Point Trigger ADC Flag When the channel6's counter is counting to PERIOD6, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[16]	CUTRGF6	Channel 6 Compare Up Trigger ADC Flag When the channel6's counter is counting up to CMP6, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[15:12]	Reserved	Reserved
[11]	ZPTRGF5	Channel 5 Zero Point Trigger ADC Flag When the channel5's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[10]	CDTRGF5	Channel 5 Compare Down Trigger ADC Flag When the channel5's counter is counting down to CMP5, this bit will be set for

		trigger ADC. **Note**: This bit can be cleared by software writing 1.
[9]	CPTRGF5	Channel 5 Center Point Trigger ADC Flag When the channel5's counter is counting to PERIOD5, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[8]	CUTRGF5	Channel 5 Compare Up Trigger ADC Flag When the channel5's counter is counting up to CMP5, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[7:4]	Reserved	Reserved
[3]	ZPTRGF4	Channel 4 Zero Point Trigger ADC Flag When the channel4's counter is counting to zero point, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[2]	CDTRGF4	Channel 4 Compare Down Trigger ADC Flag When the channel4's counter is counting down to CMP4, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[1]	CPTRGF4	Channel 4 Center Point Trigger ADC Flag When the channel4's counter is counting to PERIOD4, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.
[0]	CUTRGF4	Channel 4 Compare Up Trigger ADC Flag When the channel4's counter is counting up to CMP4, this bit will be set for trigger ADC. **Note**: This bit can be cleared by software writing 1.

3.10.7.16 Precise PWM Center-Aligned Type Control Register (PWM_PCACTL)

Register	Offset	R/W	Description	Reset Value
PWM_PCACTL	PWM_BA+0x88	R/W	PWM Precise Center-Aligned Type Control Register	0x0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved
[0]	PCAEN	PWM Precise Center-aligned Type Enable Bit 0 = Precise center-aligned type Disabled. 1 = Precise center-aligned type Enabled.

3.10.8 Operation Steps

3.10.8.1 PWM Counter Start Procedure

The following procedure is recommended for PWM counter start:

1. Configure clock prescale register (*PWM_CLKPSC*).
2. Configure clock select register (*PWM_CLKDIV*) for setting clock source select (CLKDIVn).
3. Configure PWM control register (*PWM_CTL*) for setting auto-reload mode (CNTMODEn = 1), PWM counter aligned type (CNTTYPE) and DISABLE PWM counter (CNTENn = 0).
4. Configure PWM control register (*PWM_CTL*) for setting inverter on/off (PINVn), and Dead-time generator on/off (DTCNTnm). (Optional)
5. Configure *PWM_DTCTL* register to set dead-time interval. (Optional)
6. Configure compare register (*PWM_CMPDATn*) for setting PWM duty (CMPn).
7. Configure PWM period counter register (*PWM_PERIODn*) .
8. Configure PWM interrupt enable register (*PWM_INTEN*) for setting PWM period interrupt type (INTTYPE), PWM zero interrupt enable bit (ZIENn), PWM compare up match interrupt enable bit (CMPUIENn), PWM period interrupt enable bit (PIENn), PWM compare down match interrupt enable bit (CMPDIENn). (Optional)
9. Configure PWM output enable register (*PWM_POEN*) to enable PWM output channel
10. Configure PWM control register (*PWM_CTL*) to enable PWM counter (CNTENn = 1)

3.10.8.2 PWM Counter Stop Procedure

Method 1:

Set 16-bit period counter register (PERIODn) to 0. When interrupt request happened, disable PWM counter (CNTENn in PWM_CTL). (Recommended)

Method 2:

Disable PWM Counter directly (CNTENn in PWM_CTL) (Not recommended)

The reason why this method is not recommended is that disabling CNTENn will immediately stop PWM output signal and lead to change the duty of the PWM output, this may cause damage to the motor control circuit.

3.11 Watchdog Timer (WDT)

3.11.1 Overview

The Watchdog Timer is used to perform a system reset when system runs into an unknown state. This prevents system from hanging for an infinite period of time. Besides, the Watchdog Timer supports the function to wake-up system from Idle/Power-down mode.

3.11.2 Features

- 24-bit free running up counter for WDT time-out interval.
- Selectable time-out interval (2^4 ~ 2^24) WDT_CLK cycles and the time-out interval is 0.5 ms ~ 524.288s, if WDT_CLK = 32 kHz.
- Supports selectable WDT reset delay period, including 1026, 130, 18 or 3 WDT_CLK reset delay period.
- Supports WDT time-out wake-up function only if WDT clock source is selected as RCL or XTL.

3.11.3 Block Diagram

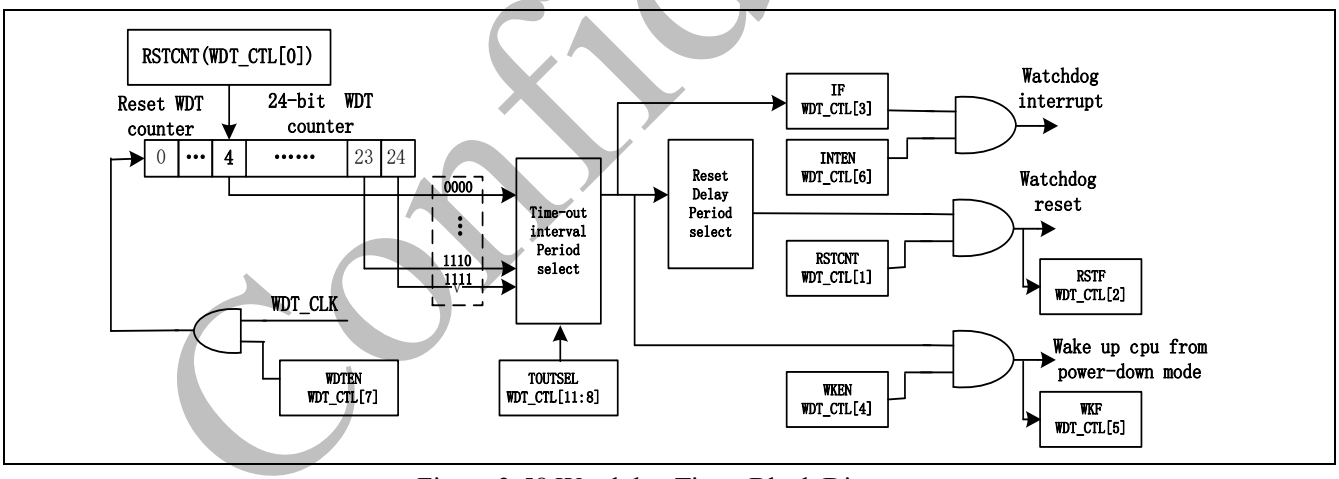


Figure 3-58 Watchdog Timer Block Diagram

3.11.4 Clock Control

The WDT clock control is shown in Figure 3-59.

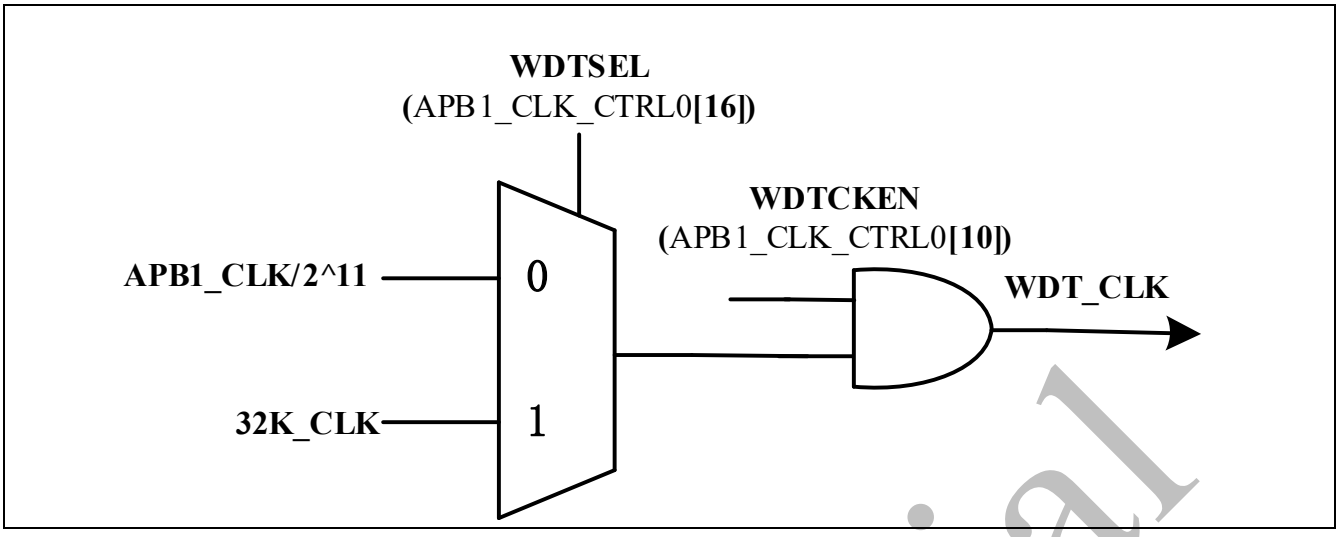


Figure 3-59 Watchdog Timer Clock Control

3.11.5 Basic Configuration

The WDT peripheral clock is enabled in *WDTCKEN* (APB1_CLK_CTRL0[10]) and clock source can be selected in *WDTSEL* (APB1_CLK_CTRL0[16]).

3.11.6 Functional Description

The WDT includes an 24-bit free running up counter with programmable time-out intervals. Table 3-11 shows the WDT time-out interval period selection and Figure 3-60 shows the WDT time-out interval and reset period timing.

3.11.6.1 WDT Time-out Flag

Setting *WDTEN* (WDT_CTL[7]) to 1 will enable the WDT function and the WDT counter to start counting up. There are eight time-out interval period can be selected by setting *TOUTSEL* (WDT_CTL[11:8]). When the WDT up counter reaches the *TOUTSEL* (WDT_CTL[11:8]) setting, the WDT time-out interrupt will occur and then WDT time-out flag TOF (WDT_CTL[16]) will be set to 1 immediately.

3.11.6.2 WDT Time-out Interrupt Flag

Setting WDTEN (WDT_CTL[7]) to 1 will enable the WDT function and the WDT counter to start counting up. There are eight time-out interval period can be selected by setting TOUTSEL (WDT_CTL[11:8]). When the WDT up counter reaches the TOUTSEL (WDT_CTL[11:8]) setting, the WDT time-out interrupt will occur and then WDT time-out interrupt flag IF (WDT_CTL[3]) will be set to 1 immediately when INTEN (WDT_CTL[6]) is set to 1.

3.11.6.3 WDT Reset Delay Period and Reset System

A specified TRSTD reset delay period occurs when the IF (WDT_CTL[3]) is set to 1. User should set *RSTCNT* (WDT_CTL[0]) to reset the 24-bit WDT up counter value to avoid generating the WDT time-out reset signal before the TRSTD reset delay period expires. Moreover, user should set *RSTDSEL* (WDT_ALTCTL [1:0]) to select reset delay period to clear WDT counter. If the WDT up counter value has not been cleared after the specified TRSTD delay period expires, the WDT control will set RSTF (WDT_CTL[2]) to 1 if RSTEN (WDT_CTL[1]) bit is enabled, and then chip enters reset state immediately. Refer to Figure 3-60 Watchdog Timer Time-out Interval and Reset Period Timing. The TRST reset period will keep the last 63 WDT clocks and then chip restart executing program from reset vector (0x0000_0000). The RSTF (WDT_CTL[2]) will keep 1 after WDT time-out resets the chip. User can check RSTF (WDT_CTL[2]) via software to recognize if the system has been reset by WDT time-out reset or not.

Table 3-11 Watchdog Timer Time-out Interval Period Selection

TOUTSEL	Time-Out Interval Period TTIS	Reset Delay Period TRSTD
0000	2^4 * TWDT	(3/18/130/1026) * TWDT
0001	2^6 * TWDT	(3/18/130/1026) * TWDT
0010	2^8 * TWDT	(3/18/130/1026) * TWDT
0011	2^{10} * TWDT	(3/18/130/1026) * TWDT
0100	2^{12} * TWDT	(3/18/130/1026) * TWDT
0101	2^{14} * TWDT	(3/18/130/1026) * TWDT
0110	2^{15} * TWDT	(3/18/130/1026) * TWDT
0111	2^{16} * TWDT	(3/18/130/1026) * TWDT
1000	2^{17} * TWDT	(3/18/130/1026) * TWDT
1001	2^{18} * TWDT	(3/18/130/1026) * TWDT
1010	2^{19} * TWDT	(3/18/130/1026) * TWDT
1011	2^{20} * TWDT	(3/18/130/1026) * TWDT
1100	2^{21} * TWDT	(3/18/130/1026) * TWDT
1101	2^{22} * TWDT	(3/18/130/1026) * TWDT
1110	2^{23} * TWDT	(3/18/130/1026) * TWDT
1111	2^{24} * TWDT	(3/18/130/1026) * TWDT

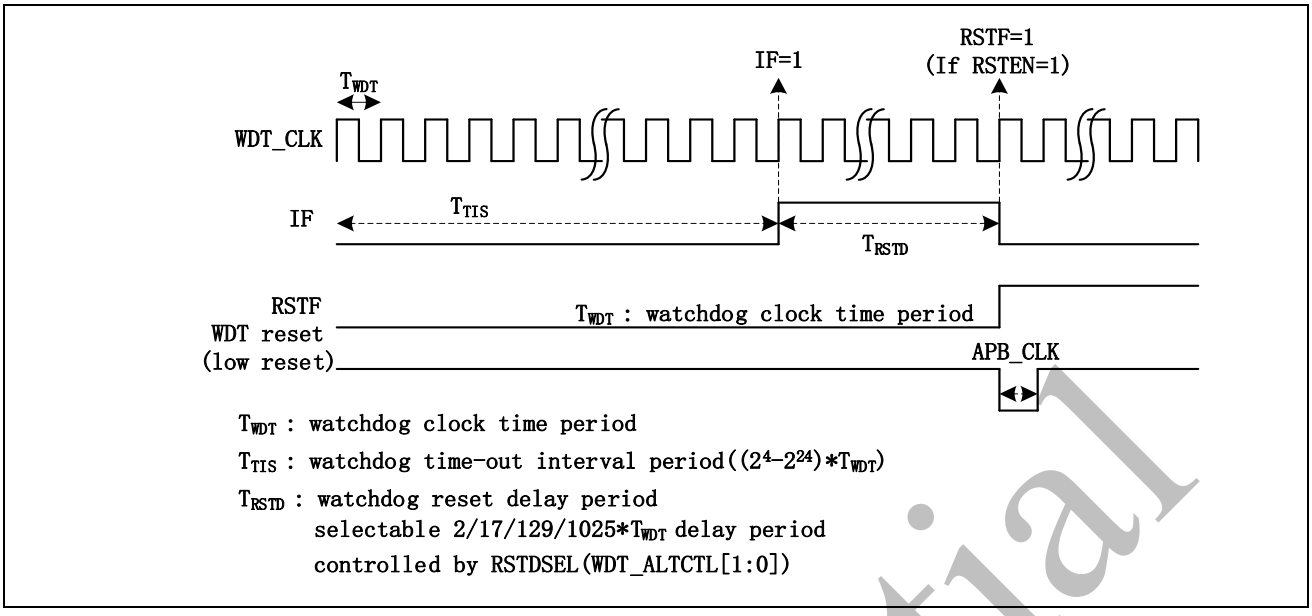


Figure 3-60 Watchdog Timer Time-out Interval and Reset Period Timing

3.11.6.4 WDT Wake-up

If WDT clock source is selected to RCL or XTL, system can be woken up from Power-down mode while WDT time-out interrupt signal is generated and WKEN (WDT_CTL[4]) enabled. In the meanwhile, the *WKF* (WDT_CTL[5]) will be set to 1 automatically. User can check *WKF* (WDT_CTL[5]) status via software to recognize if the system has been woken up by WDT time-out interrupt or not.

3.11.7 WDT Control Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
WDT Base Address: WDT_BA = 0x4000_4000				
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0F00
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000

3.11.8 WDT Register Description

3.11.8.1 WDT Control Register (WDT_CTL)

Register	Offset	R/W	Description	Reset Value
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0F00

Bits	Descriptions	
[31]	ICEDEBUG	ICE Debug Mode Acknowledge Disable Bit (Write Protect) 0 = ICE debug mode acknowledgment affects WDT counting. WDT up counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgment Disabled. WDT up counter will keep going no matter CPU is held by ICE or not. **Note**: This bit is write protected. Refer to the *SYS_REGLCTL* register.
[30:17]	Reserved	Reserved.
[16]	TOF	WDT Time-out Flag This bit will be set to 1 while WDT up counter value reaches the selected WDT time-out interval 0 = WDT time-out interrupt did not occur. 1 = WDT time-out interrupt occurred. **Note**: This bit is cleared by writing 1 to it.
[15:13]	Reserved	Reserved.
[12]	RST_REGION_SEL	Reset Region Select(Write Protect) 0: just 1.2V region without anactrl & rcc module 1: all chip without work_mode(rom mode) module
[11:8]	TOUTSEL	WDT Time-out Interval Selection (Write Protect) These four bits select the time-out interval period for the WDT. 0000 = 2^4 * WDT_CLK. 0001 = 2^6 * WDT_CLK. 0010 = 2^8 * WDT_CLK. 0011 = 2^10 * WDT_CLK. 0100 = 2^12 * WDT_CLK. 0101 = 2^14 * WDT_CLK. 0110 = 2^15 * WDT_CLK. 0111 = 2^16 * WDT_CLK. 1000 = 2^17 * WDT_CLK. 1001 = 2^18 * WDT_CLK. 1010 = 2^19 * WDT_CLK. 1011 = 2^20 * WDT_CLK. 1100 = 2^21 * WDT_CLK. 1101 = 2^22 * WDT_CLK. 1110 = 2^23 * WDT_CLK. 1111 = 2^24 * WDT_CLK. **Note**: This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	WDTEN	WDT Enable Bit (Write Protect) 0 = WDT Disabled (This action will reset the internal up counter value). 1 = WDT Enabled.

		Note: This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	INTEN	WDT Time-out Interrupt Enable Bit (Write Protect) If this bit is enabled, the WDT time-out interrupt signal is generated and inform to CPU. 0 = WDT time-out interrupt Disabled. 1 = WDT time-out interrupt Enabled. **Note**: This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	WKF	WDT Time-out Wake-up Flag (Write Protect) This bit indicates the interrupt wake-up flag status of WDT 0 = WDT does not cause chip wake-up. 1 = Chip wake-up from deepsleep mode if WDT time-out interrupt signal generated. **Note1**: This bit is write protected. Refer to the SYS_REGLCTL register. **Note2**: This bit is cleared by writing 1 to it.
[4]	WKEN	WDT Time-out Wake-up Function Control (Write Protect) If this bit is set to 1, while WDT time-out interrupt flag IF (WDT_CTL[3]) is generated to 1 and interrupt enable bit INTEN (WDT_CTL[6]) is enabled, the WDT time-out interrupt signal will generate a wake-up trigger event to chip. 0 = Wake-up trigger event Disabled if WDT time-out interrupt signal generated. 1 = Wake-up trigger event Enabled if WDT time-out interrupt signal generated. **Note**: This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	IF	WDT Time-out Interrupt Flag This bit will be set to 1 while WDT up counter value reaches the selected WDT time-out interval 0 = WDT time-out interrupt did not occur. 1 = WDT time-out interrupt occurred. **Note**: This bit is cleared by writing 1 to it.
[2]	RSTF	WDT Time-out Reset Flag This bit indicates the system has been reset by WDT time-out reset or not. 0 = WDT time-out reset did not occur. 1 = WDT time-out reset occurred. **Note**: This bit is cleared by writing 1 to it.
[1]	RSTEN	WDT Time-out Reset Enable Bit (Write Protect) Setting this bit will enable the WDT time-out reset function If the WDT up counter value has not been cleared after the specific WDT reset delay period expires. 0 = WDT time-out reset function Disabled. 1 = WDT time-out reset function Enabled. **Note**: This bit is write-protected. Refer to the SYS_REGLCTL register.
[0]	RSTCNT	Reset WDT Up Counter (Write Protect) 0 = No effect. 1 = Reset the internal 24-bit WDT up counter value. **Note1**: This bit is write protected. Refer to the SYS_REGLCTL register. **Note2**: This bit will be automatically cleared by hardware.

3.11.8.2 WDT Alternative Control Register (WDT_ALTCTL)

Register	Offset	R/W	Description	Reset Value
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000

Bits	Descriptions	
[31:2]	Reserved	Reserved.
[1:0]	RSTDSEL	WDT Reset Delay Selection (Write Protect) When WDT time-out happened, user has a time named WDT Reset Delay Period to clear WDT counter by setting RSTCNT (WDT_CTL[0]) to prevent WDT time-out reset happened. User can select a suitable setting of RSTDSEL for different WDT Reset Delay Period. 00 = WDT Reset Delay Period is 1026* WDT_CLK. 01 = WDT Reset Delay Period is 130 * WDT_CLK. 10 = WDT Reset Delay Period is 18 * WDT_CLK. 11 = WDT Reset Delay Period is 3 * WDT_CLK. **Note1**: This bit is write protected. Refer to the SYS_REGLCTL register. **Note2**: This register will be reset to 0 if WDT time-out reset happened.

3.12 Window Watchdog Timer (WWDT)

3.12.1 Overview

The Window Watchdog Timer (WWDT) is used to perform a system reset within a specified window period to prevent software run to uncontrollable status by any unpredictable condition.

3.12.2 Features

- 6-bit down counter value (CNTDAT) and 6-bit compare value (CMPDAT) to make the WWDT time-out window period flexible
- Supports 4-bit value (PSCSEL) to programmable maximum 11-bit prescale counter period of WWDT counter

3.12.3 Block Diagram

The WWDT block diagram is shown in Figure 3-61.

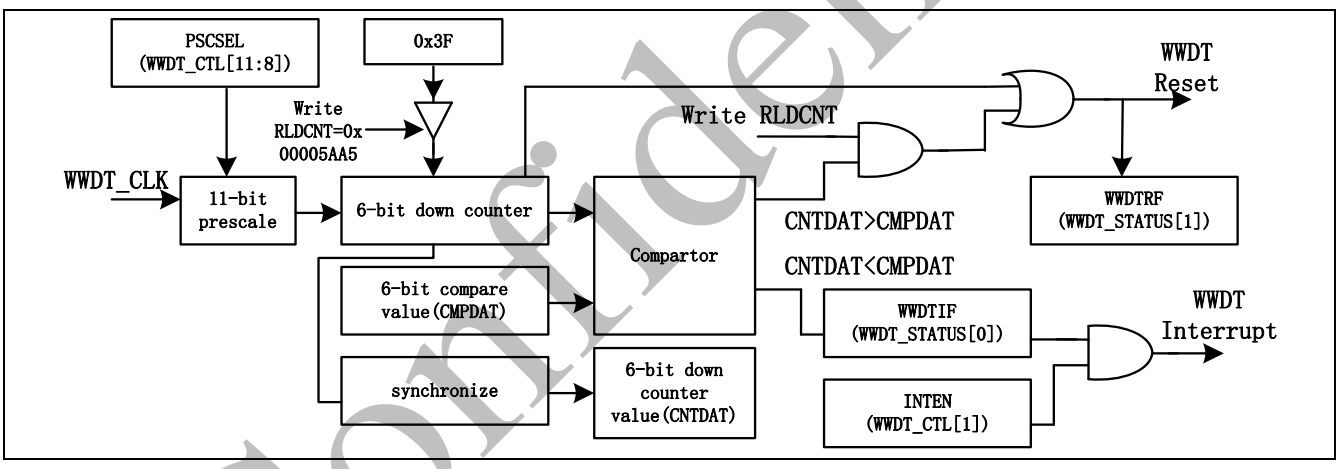


Figure 3-61 WWDT Block Diagram

3.12.4 Clock Control

The WWDT peripheral clock is enabled in WDTCKEN (CLK_APBCLK[0]) and clock source can be selected in WWDTSEL[1:0] (CLK_CLKSEL2[17:16]).

The WWDT clock control is shown in Figure 3-62.

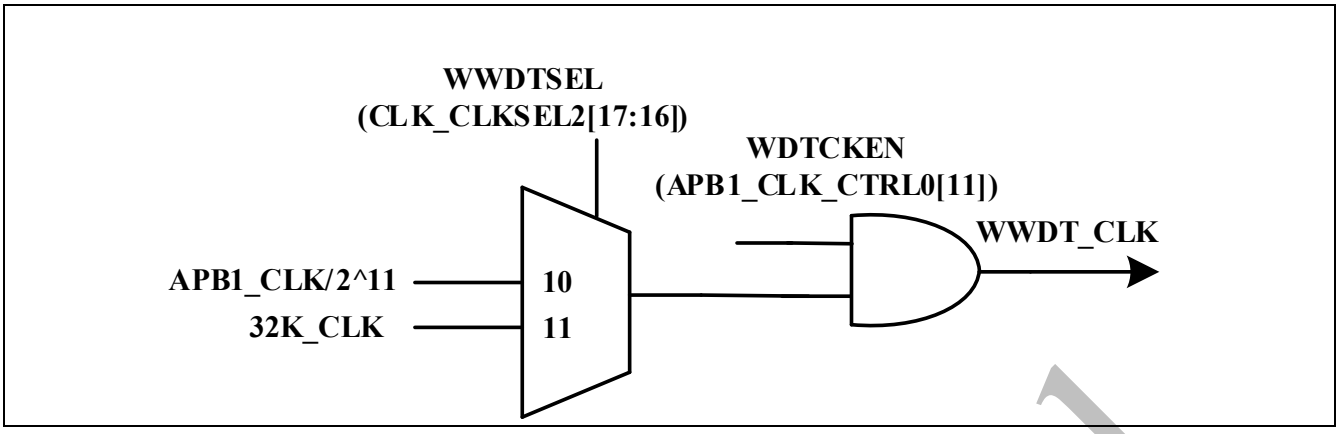


Figure 3-62 WWDT Clock Control

3.12.5 Functional Description

The WWDT includes a 6-bit down counter with programmable prescale value to define different WWDT time-out intervals. The clock source of 6-bit WWDT is based on system clock divide 2048 (HCLK/2048) or 32 kHz internal low speed RC oscillator (LIRC) with a programmable 11-bit prescale counter value which controlled by *PSCSEL* (WWDT_CTL[11:8]). Also, the correlate of PSCSEL (WWDT_CTL[11:8]) and prescale value are listed in Table 3-12.

Table 3-12 WWDT Prescaler Value Selection

PSCSEL	Prescaler Value	Max. Time-Out Period	Max. Time-Out Interval (WWDT_CLK=32 KHz)
0000	1	1 * 64 * T_{WWDT}	2ms
0001	2	2 * 64 * T_{WWDT}	4ms
0010	4	4 * 64 * T_{WWDT}	8ms
0011	8	8 * 64 * T_{WWDT}	16ms
0100	16	16 * 64 * T_{WWDT}	32ms
0101	32	32 * 64 * T_{WWDT}	64ms
0110	64	64 * 64 * T_{WWDT}	128ms
0111	128	128 * 64 * T_{WWDT}	256ms
1000	192	192 * 64 * T_{WWDT}	384ms
1001	256	256 * 64 * T_{WWDT}	512ms
1010	384	384 * 64 * T_{WWDT}	768ms
1011	512	512 * 64 * T_{WWDT}	1.024s
1100	768	768 * 64 * T_{WWDT}	1.536s
1101	1024	1024 * 64 * T_{WWDT}	2.048s
1110	1536	1536 * 64 * T_{WWDT}	3.072s
1111	2048	2048 * 64 * T_{WWDT}	4.096s

3.12.5.1 WWDT Counting

When the WWDTEN (WWDT_CTL[0]) is set, WWDT down counter will start counting from 0x3F to 0. To prevent program runs to disable WWDT counter counting unexpected, the WWDT_CTL register can only be written once after chip is powered on or reset. User cannot disable WWDT counter counting (WWDTEN), change counter prescale period (PSCSEL) or change window compare value (CMPDAT) while WWDTEN (WWDT_CTL[0]) has been enabled by user unless chip is reset.

3.12.5.2 WWDT Compare Match Flag

During down counting by the WWDT counter, the WWDTF (WWDT_STATUS[2]) is set to 1 while the WWDT counter value (CNTDAT) is equal to window compare value (CMPDAT) and WWDTF can be cleared by user.

3.12.5.3 WWDT Compare Match Interrupt Flag

During down counting by the WWDT counter, the WWDTIF (WWDT_STATUS[0]) is set to 1 while the WWDT counter value (CNTDAT) is equal to window compare value (CMPDAT) and INTEN(WWDT_CTL[1]) is set to 1. WWDTIF can be cleared by user.

3.12.5.4 WWDT Reset System

When WWDTIF (WWDT_STATUS[0]) is generated, user must reload WWDT counter value to 0x3F by writing 0x00005AA5 to WWDT_RLDCNT register, and also to prevent WWDT counter value reached to 0 and generate WWDT reset system signal to inform system reset. If current CNTDAT (WWDT_CNT[5:0]) is larger than CMPDAT (WWDT_CTL[21:16]) and user writes 0x00005AA5 to the WWDT_RLDCNT register, the WWDT reset system signal will be generated immediately to cause chip reset also.

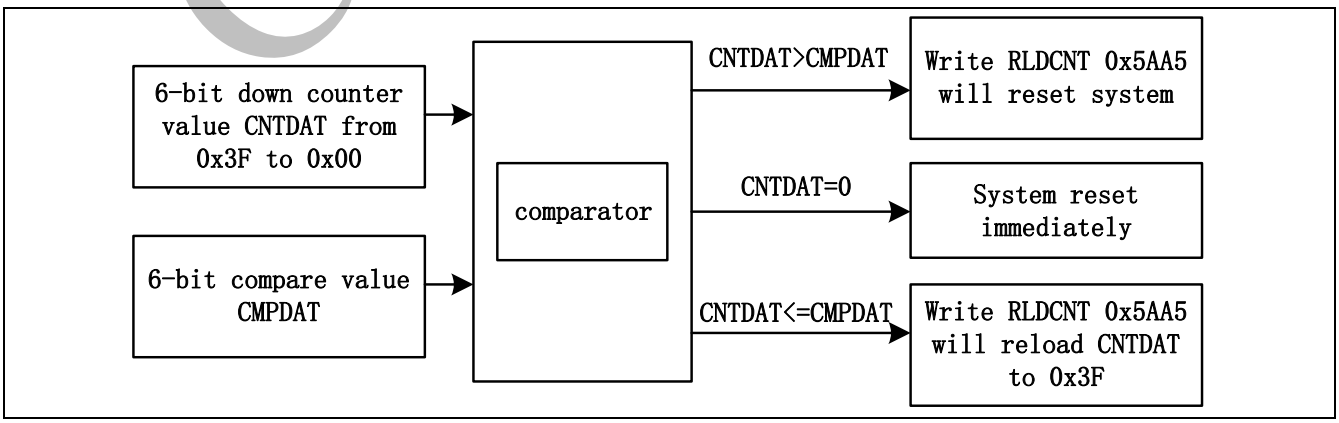


Figure 3-63 WWDT Reset and Reload Behavior

3.12.5.5 WWDT Window Setting Limitation

When user writes 0x00005AA5 to WWDT_RLDCNT register to reload WWDT counter value to 0x3F, it needs 3 WWDT clocks to sync the reload command to actually perform reload action. Notice that if user set PSCSEL (WWDT_CTL[11:8]) to 0000, the counter prescale value should be as 1, and the CMPDAT (WWDT_CTL[21:16]) must be larger than 2. Otherwise, writing WWDT_RLDCNT register to reload WWDT counter value to 0x3F is unavailable, WWDTIF(WWDT_STATUS[0]) is generated, and WWDT reset system event always happened.

Table 3-13 CMPDAT Setting Limitation

PSCSEL	Prescale	Value
0000	1	0x03 ~ 0x3F
0001	2	0x02 ~ 0x3F
others	others	0x00 ~ 0x3F

3.12.6 WWDT Control Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
WWDT Base Address: WWDT_BA = 0x4000_4100				
WWDT_RLDCNT	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000
WWDT_CNT	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

3.12.7 WWDT Register Description

3.12.7.1 WWDT Reload Counter Register (WWDT_RLDCNT)

Register	Offset	R/W	Description	Reset Value
WWDT_RLDCNT	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000

Bits	Descriptions	
[31:0]	RLDCNT	WWDT Reload Counter Register Writing 0x00005AA5 to this register will reload the WWDT counter value to 0x3F. **Note**: User can only write WWDT_RLDCNT register to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT (WWDT_CTL[21:16]). If user writes WWDT_RLDCNT when current WWDT counter value is larger than CMPDAT, WWDT reset signal will generate immediately.

3.12.7.2 WWDT Control Register (WWDT_CTL)

Register	Offset	R/W	Description	Reset Value
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800

Bits	Descriptions	
[31]	ICEDEBUG	ICE Debug Mode Acknowledge Disable Bit 0 = ICE debug mode acknowledgment effects WWDT counting. WWDT down counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgment Disabled. WWDT down counter will keep going no matter CPU is held by ICE or not.
[30:22]	Reserved	Reserved
[21:16]	CMPDAT	WWDT Window Compare Bits Set this register to adjust the valid reload window. **Note**: User can only write WWDT_RLDCNT register to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT. If user writes WWDT_RLDCNT register when current WWDT counter value larger than CMPDAT, WWDT reset signal will generate immediately.
[15:12]	Reserved	Reserved
[11:8]	PSCSEL	WWDT Counter Prescale Period Select Bits 0000 = Pre-scale is 1; Max time-out period is 1 * 64 * WWDT_CLK. 0001 = Pre-scale is 2; Max time-out period is 2 * 64 * WWDT_CLK. 0010 = Pre-scale is 4; Max time-out period is 4 * 64 * WWDT_CLK. 0011 = Pre-scale is 8; Max time-out period is 8 * 64 * WWDT_CLK. 0100 = Pre-scale is 16; Max time-out period is 16 * 64 * WWDT_CLK. 0101 = Pre-scale is 32; Max time-out period is 32 * 64 * WWDT_CLK. 0110 = Pre-scale is 64; Max time-out period is 64 * 64 * WWDT_CLK. 0111 = Pre-scale is 128; Max time-out period is 128 * 64 * WWDT_CLK. 1000 = Pre-scale is 192; Max time-out period is 192 * 64 * WWDT_CLK. 1001 = Pre-scale is 256; Max time-out period is 256 * 64 * WWDT_CLK. 1010 = Pre-scale is 384; Max time-out period is 384 * 64 * WWDT_CLK. 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * WWDT_CLK. 1100 = Pre-scale is 768; Max time-out period is 768 * 64 * WWDT_CLK. 1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * WWDT_CLK. 1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * WWDT_CLK. 1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * WWDT_CLK.
[7:2]	Reserved	Reserved
[1]	INTEN	WWDT Interrupt Enable Bit If this bit is enabled, the WWDT counter compare match interrupt signal is generated and inform to CPU. 0 = WWDT counter compare match interrupt Disabled. 1 = WWDT counter compare match interrupt Enabled.
[0]	WWDTEN	WWDT Enable Bit Set this bit to enable WWDT counter counting. 0 = WWDT counter is stopped. 1 = WWDT counter is starting counting.

3.12.7.3 WWDT Status Register (WWDT_STATUS)

Register	Offset	R/W	Description	Reset Value
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved
[2]	WWDTF	WWDT Compare Match Flag This bit indicates the flag status of WWDT while WWDT counter value matches CMPDAT (WWDT_CTL[21:16]). 0 = No effect. 1 = WWDT counter value matches CMPDAT. **Note**: This bit is cleared by writing 1 to it.
[1]	WWDTRF	WWDT Timer-out Reset Flag This bit indicates the system has been reset by WWDT time-out reset or not. 0 = WWDT time-out reset did not occur. 1 = WWDT time-out reset occurred. **Note**: This bit is cleared by writing 1 to it.
[0]	WWDTIF	WWDT Compare Match Interrupt Flag This bit indicates the interrupt flag status of WWDT while WWDT counter value matches CMPDAT (WWDT_CTL[21:16]). 0 = No effect. 1 = WWDT counter value matches CMPDAT. **Note**: This bit is cleared by writing 1 to it.

3.12.7.4 WWDT Counter Value Register (WWDT_CNT)

Register	Offset	R/W	Description	Reset Value
WWDT_CNT	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

Bits	Descriptions	
[31:6]	Reserved	Reserved
[5:0]	CNTDAT	WWDT Counter Value CNTDAT will be updated continuously to monitor 6-bit WWDT down counter value.

3.13 I2C Serial Interface Controller (I2C)

3.13.1 Overview

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. A master is a device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

The I2C module can operate in standard mode (with data rates 0 to 100 Kb/s), fast mode (with data rates less than or equal to 400 Kb/s), fast mode plus (with data rates less than or equal to 1000 Kb/s), high-speed mode (with data rates less than or equal to 1.6Mb/s).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

3.13.2 Features

- Support 2 I2C device
- Three speeds:
  - Standard mode (0 to 100 Kb/s)
  - Fast mode (≤ 400 Kb/s) or fast mode plus (≤ 1000 Kb/s)
  - High-speed mode (≤ 1.6 Mb/s)
- Two operation mode:
  - Slave mode
  - Master mode
- Clock synchronization
- 7- or 10-bit addressing
- 7- or 10-bit combined format transfers
- Bulk transmit mode
- Ignore CBUS addresses (an older ancestor of I2C that used to share the I2C bus)
- Transmit and receive buffers
- Interrupt or polled-mode operation
- Support DMA

- Programmable SDA hold time (tHD;I2C_DATA)
- Multiple masters arbitration

3.13.3 Block Diagram

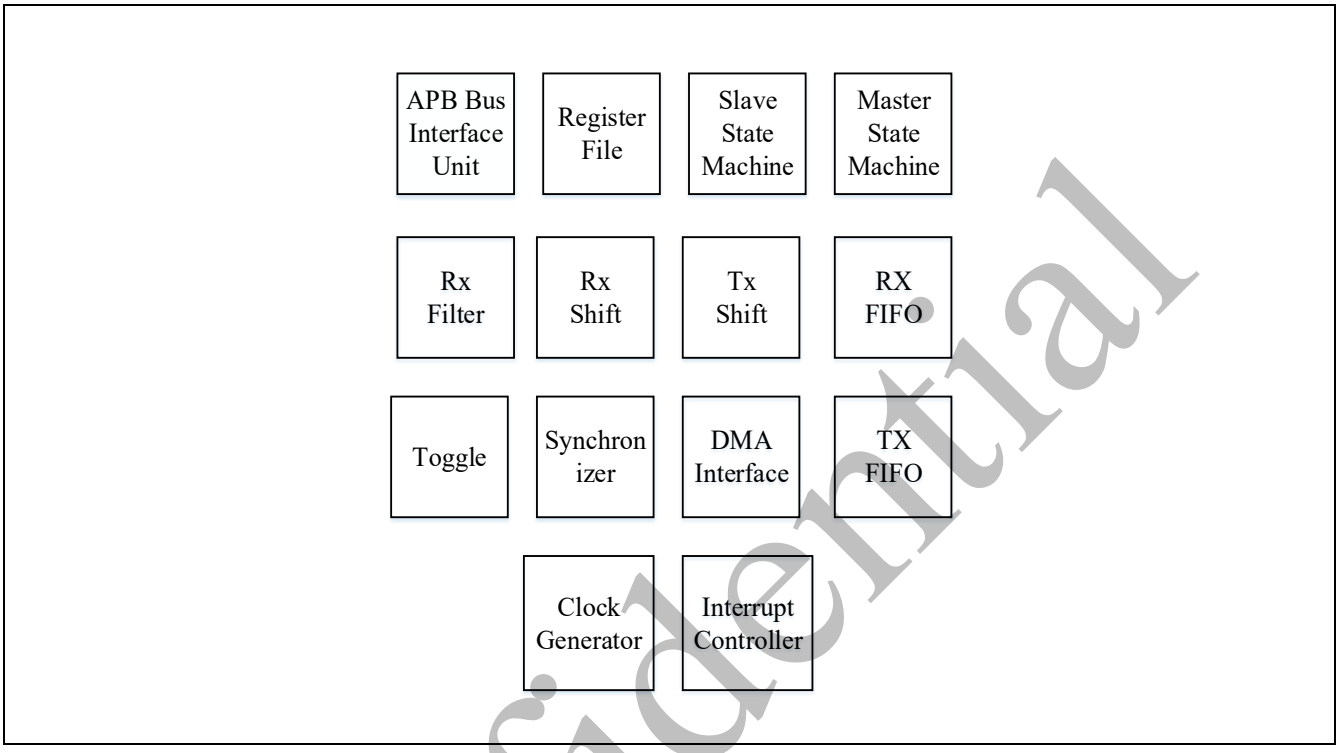


Figure 3-64 I2C Controller Block Diagram

3.13.4 Functional Description

This chapter describes the functional behavior of I2C in more detail.

3.13.4.1 I2C Behavior

On I2C bus, data is transferred between a Master and a Slave. The master is responsible for generating the clock and controlling the transfer of data. The slave is responsible for either transmitting or receiving data to/from the master. Data bits transfer on the SCL and SDA lines are synchronous on a byte-by-byte basis. There is one SCL clock pulse for each data bit with the MSB being transmitted first, and an acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Each slave has a unique address that is determined by the system designer. When a master wants to communicate with a slave, the master transmits a START/RESTART condition that

is then followed by the slave's address and a control bit (R/W) to determine if the master wants to transmit data or receive data from the slave. The slave then sends an acknowledge (ACK) pulse after the address.

Moreover, the I2C protocol also allows multiple masters to reside on the I2C bus and uses an arbitration procedure to determine bus ownership.

The interface can operate in one of the four following modes:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

If the master (master-transmitter) is writing to the slave (slave-receiver), the receiver gets one byte of data. This transaction continues until the master terminates the transmission with a STOP condition. If the master is reading from a slave (master-receiver), the slave transmits (slave-transmitter) a byte of data to the master, and the master then acknowledges the transaction with the ACK pulse. This behavior is illustrated in Figure 3-65.This transaction continues until the master terminates the transmission by not acknowledging (NAK) the transaction after the last byte is received, and then the master issues a STOP condition or addresses another slave after issuing a RESTART condition. This behavior is illustrated in Figure 3-66.

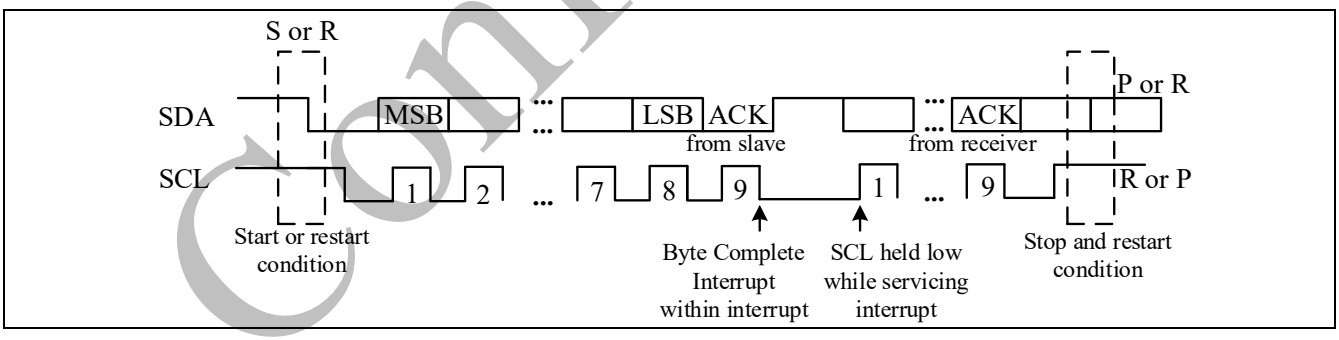


Figure 3-65 Data transfer on the I2C Bus for Master Transmitter

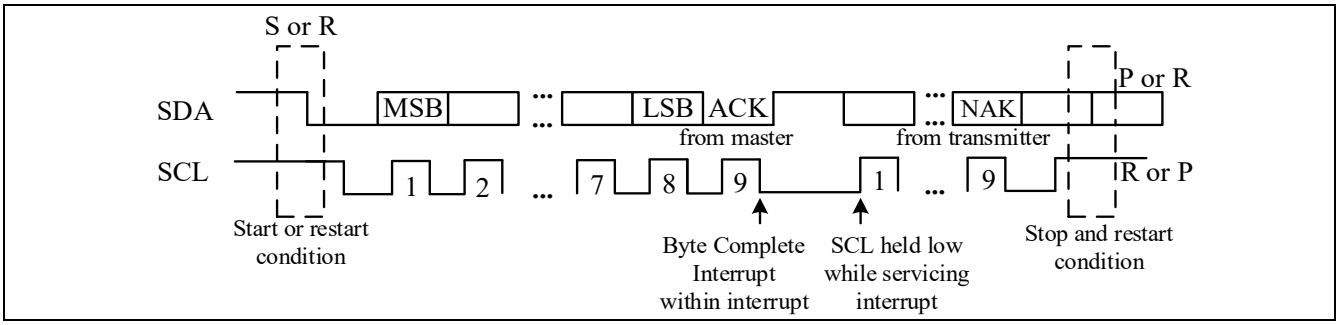


Figure 3-66 Data transfer on the I2C Bus for Master Receiver

The I2C is a synchronous serial interface. The SDA line is a bidirectional signal and changes only while the SCL line is low, except for STOP, START, and RESTART conditions. The output drivers are open-drain or open-collector to perform wire-AND functions on the bus. Data is transmitted in byte packages.

When operating as an I2C master, putting data into the transmit FIFO causes the I2C to generate a START condition on the I2C bus. Writing a 1 to DATA_CMD[9] causes the I2C to generate a STOP condition on the I2C bus; a STOP condition is not issued if this bit is not set, even if the transmit FIFO is empty.

When operating as a slave, the I2C does not generate START and STOP conditions, as per the protocol. However, if a read request is made to the I2C, it holds the SCL line low until read data has been supplied to it. This stalls the I2C bus until read data is provided to the slave I2C module, or the I2C slave is disabled by writing a 0 to bit 0 of the I2C_EN register.

The I2C supports mixed read and write combined format transactions in both 7-bit and 10-bit addressing modes.

The I2C does not support mixed address and mixed address format—that is, a 7-bit address transaction followed by a 10-bit address transaction or vice versa—combined format transactions.

To initiate combined format transfers, RESTART_EN should be set to 1. With this value set and operating as a master, when the I2C completes an I2C transfer, it checks the transmit FIFO and executes the next transfer. If the direction of this transfer differs from the previous transfer, the combined format is used to issue the transfer. If the transmit FIFO is empty when the current I2C transfer completes, DATA_CMD[9] is checked:

- If set to 1, a STOP bit is issued.
- If set to 0, the SCL is held low until the next command is written to the transmit FIFO.

3.13.4.2 I2C Protocols

Normally, a standard communication consists of four parts:

1) START or Repeated START signal generation
2) Slave address and R/W bit transfer
3) Data transfer
4) STOP signal generation

START and STOP Conditions

When the bus is idle, both the SCL and SDA signals are pulled high through external pull-up

resistors on the bus. When the master wants to start a transmission on the bus, the master issues a START condition. This is defined to be a high-to-low transition of the SDA signal while SCL is 1. When the master wants to terminate the transmission, the master issues a STOP condition. This is defined to be a low-to-high transition of the SDA line while SCL is 1. Figure 3-67 shows the timing of the START and STOP conditions. When data is being transmitted on the bus, the SDA line must be stable when SCL is 1.

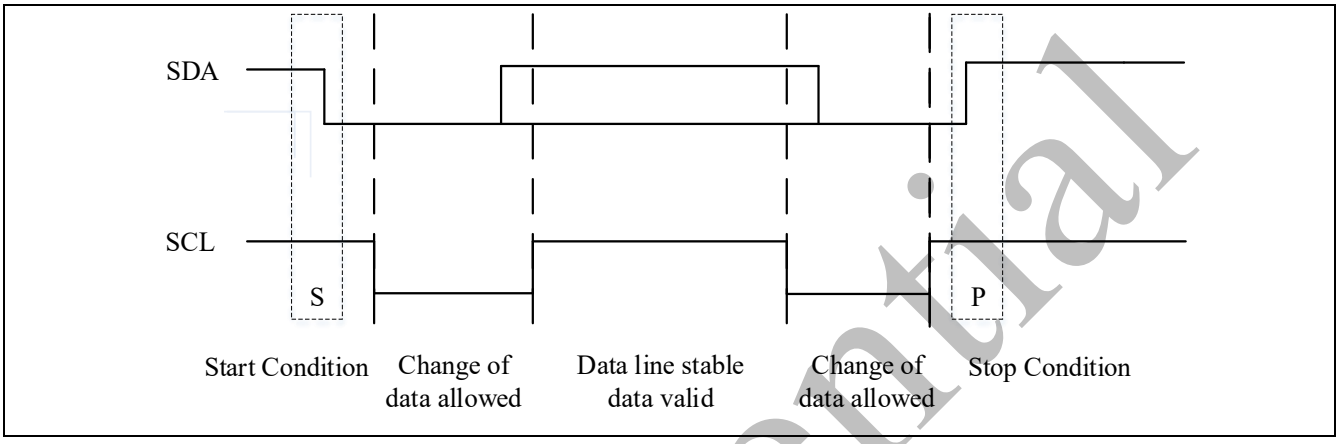


Figure 3-67 START and STOP Condition

Addressing Slave Protocol

There are two address formats: the 7-bit address format and the 10-bit address format.

7-bit Address Format:

During the 7-bit address format, the first seven bits (bits 7:1) of the first byte set the slave address and the LSB bit (bit 0) is the R/W bit as shown in Figure 3-68. When bit 0 (R/W) is set to 0, the master writes to the slave. When bit 0 (R/W) is set to 1, the master reads from the slave.

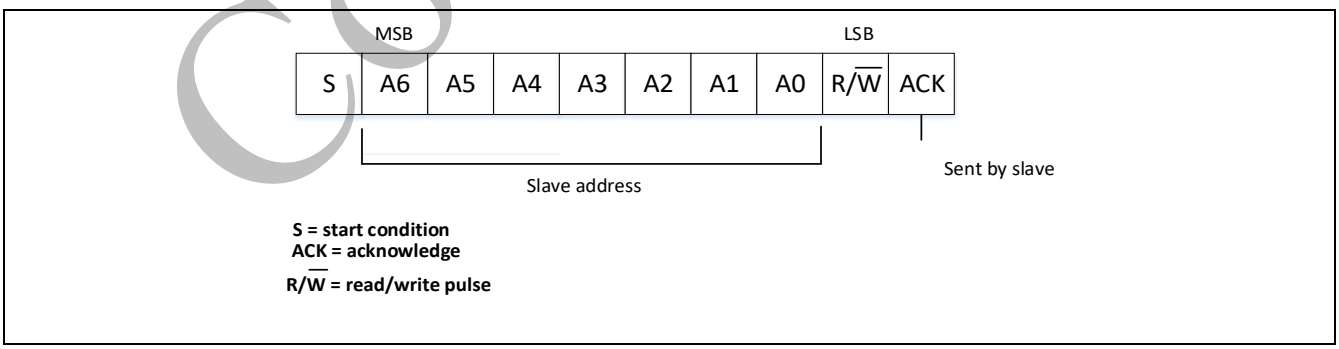


Figure 3-68 7-bit Address Format

10-bit Address Format:

During 10-bit addressing, two bytes are transferred to set the 10-bit address. The transfer of the first byte contains the following bit definition. The first five bits (bits 7:3) notify the slaves that this is a 10-bit transfer followed by the next two bits (bits 2:1), which set the slaves

address bits 9:8, and the LSB bit (bit 0) is the R/W bit. The second byte transferred sets bits 7:0 of the slave address. Figure 3-69 shows the 10-bit address format.

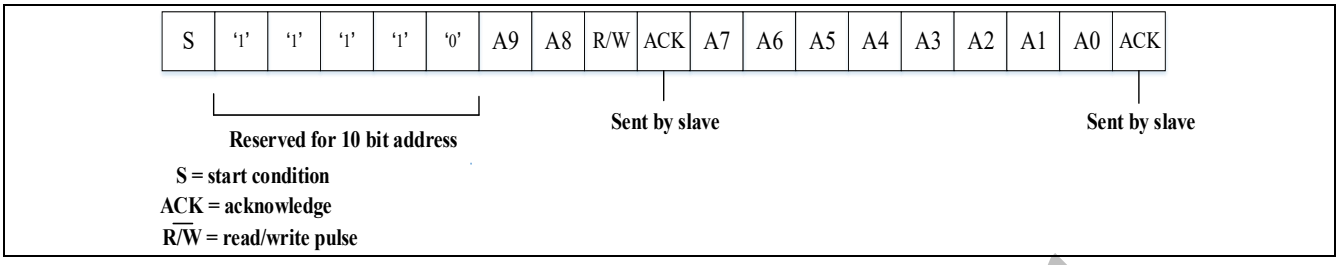


Figure 3-69 10-bit Address Format

Table 3-14 I2C Definition of Bits in First Byte

Slave Address	R/W Bit	Description
0000 000	0	General Call Address. I2C places the data in the receive buffer and issues a General Call interrupt.
0000 000	1	START Byte.
0000 001	X	CBUS address. Our I2C ignores these accesses.
0000 010	X	Reserved
0000 011	X	Reserved
0000 1XX	X	High-speed master code
1111 1XX	X	Reserved
1111 0×X	X	10-bit slave addressing
0001 000	X	Reserved
0001 100	X	Reserved
1100 001	X	Reserved

This I2C does not restrict you from using these reserved addresses. However, if you use these reserved addresses, you may run into incompatibilities with other I2C components.

Transmitting and Receiving Protocol

The master can initiate data transmission and reception to/from the bus, acting as either a master-transmitter or master-receiver. A slave responds to requests from the master to either transmit data or receive data to/from the bus, acting as either a slave-transmitter or slave-receiver, respectively

Master-Transmitter and Slave-Receiver

All data is transmitted in byte format, with no limit on the number of bytes transferred per data transfer. After the master sends the address and R/W bit or the master transmits a byte of data to the slave, the slave-receiver must respond with the acknowledge signal (ACK). When a slave-receiver does not respond with an ACK pulse, the master aborts the transfer by issuing a STOP condition. The slave must leave the SDA line high so that the master can abort the transfer.

If the master-transmitter is transmitting data as shown in Figure 3-70, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

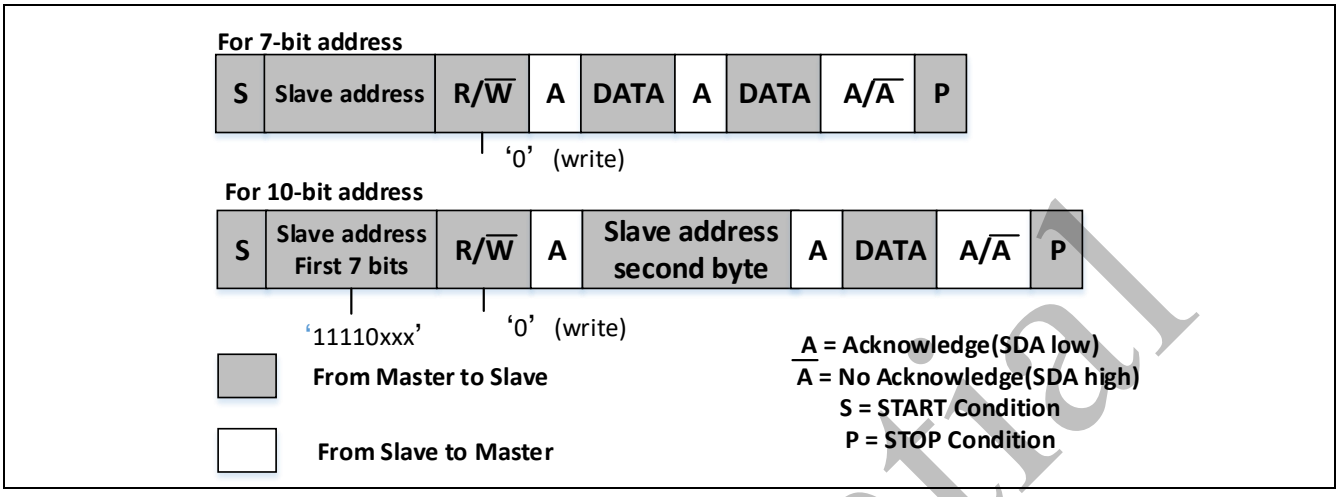


Figure 3-70 Master-Transmitter Protocol

Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 3-71, then the master responds to the slave-transmitter with an acknowledge pulse after a byte of data has been received, except for the last byte. This is the way the master-receiver notifies the slave-transmitter that this is the last byte. The slave-transmitter relinquishes the SDA line after detecting the No Acknowledge (NACK) so that the master can issue a STOP condition.

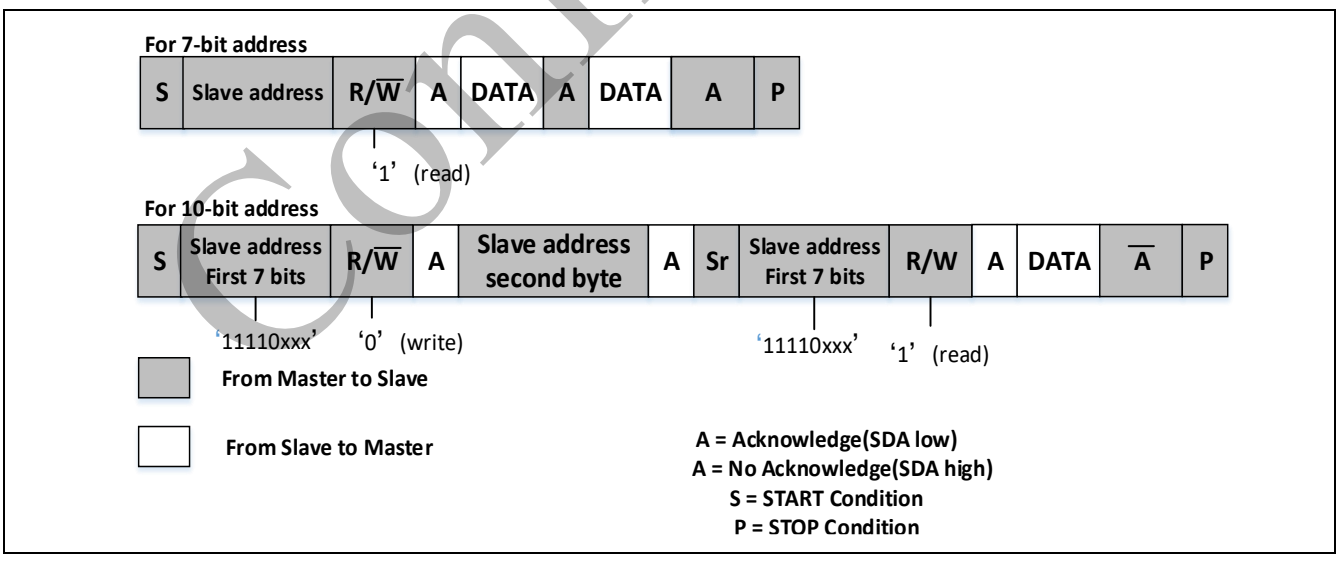


Figure 3-71 Master-Receiver Protocol

When a master does not want to relinquish the bus with a STOP condition, the master can issue a RESTART condition. This is identical to a START condition except it occurs after the ACK pulse. Operating in master mode, the I2C can then communicate with the same slave

using a transfer of a different direction.

START BYTE Transfer Protocol:

The START BYTE transfer protocol is set up for systems that do not have an on-board dedicated I2C hardware module. When the I2C is addressed as a slave, it always samples the I2C bus at the highest speed supported so that it never requires a START BYTE transfer. However, when I2C is a master, it supports the generation of START BYTE transfers at the beginning of every transfer in case a slave device requires it.

This protocol consists of seven zeros being transmitted followed by a 1, as illustrated in Figure 3-72. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the micro-controller detects a 0, it switches from the under sampling rate to the correct rate of the master.

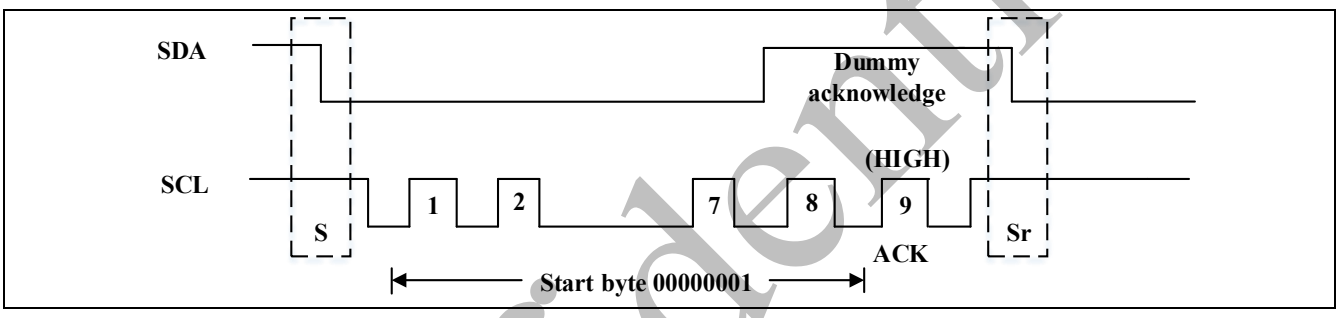


Figure 3-72 START BYTE Transfer

The START BYTE procedure is as follows:

1 Master generates a START condition.
2 Master transmits the START byte (0000 0001).
3 Master transmits the ACK clock pulse. (Present only to conform with the byte handling format used on the bus)
4 No slave sets the ACK signal to 0.
5 Master generates a RESTART (R) condition.

A hardware receiver does not respond to the START BYTE because it is a reserved address and resets after the RESTART condition is generated.

3.13.4.3 Tx FIFO Management and START, STOP and RESTART Generation

When I2C operates as a master and IC_EMPTYFIFO_HOLD_MASTER_EN = 1, the component does not generate a STOP if the Tx FIFO becomes empty; in this situation, the component holds the SCL line low, stalling the bus until a new entry is available in the Tx FIFO. A STOP condition is generated only when the user specifically requests it by setting bit 9 (Stop bit) of the command written to DATA_CMD register.

IC_DATA_CMD	RESTART	STOP	CMD	DATA
	10	9	8	7 ... 0

DATA –Read/Write field; data retrieved from slave is read from this field; data to be sent to slave is written to this field
CMD –Write-only field; this bit determines whether transfer to be carried out is Read (CMD=1) or Write (CMD=0)
Stop –Write-only field; this bit determines whether STOP is generated after data byte is sent or received
Restart–Write-only field; this bit determines whether RESTART (or STOP followed by START in case of restart capa bility is not enabled) is generated before data byte is sent of received

Figure 3-73 DATA_CMD Register if IC_EMPTYFIFO_HOLD_MASTER_EN = 1

3.13.4.4 Multiple Master Arbitration

The I2C bus protocol allows multiple masters to reside on the same bus. If there are two masters on the same I2C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating a START condition at the same time. Once a master (for example, a micro-controller) has control of the bus, no other master can take control until the first master sends a STOP condition and places the bus in an idle state.

Arbitration takes place on the SDA line, while the SCL line is 1. The master, which transmits a 1 while the other master transmits 0, loses arbitration and turns off its data output stage. The master that lost arbitration can continue to generate clocks until the end of the byte transfer. If both masters are addressing the same slave device, the arbitration could go into the data phase.

Upon detecting that it has lost arbitration to another master, the I2C will stop generating SCL. Figure 3-74 illustrates the timing of when two masters are arbitrating on the bus.

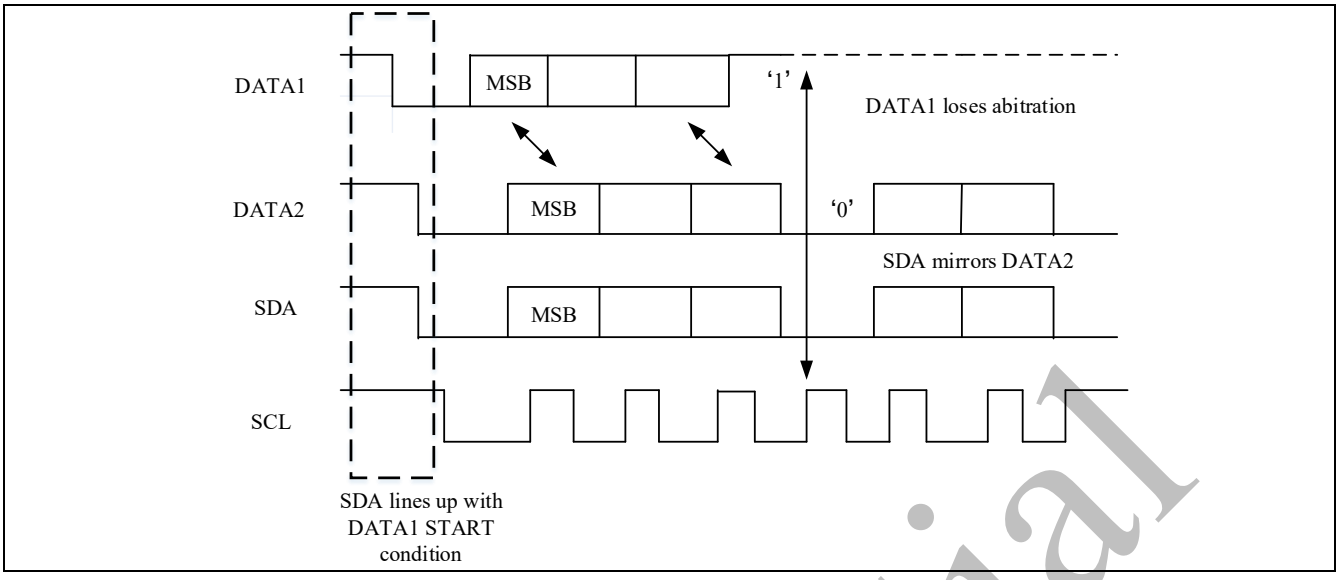


Figure 3-74 Multiple Master Arbitration

For high-speed mode, the arbitration cannot go into the data phase because each master is programmed with a unique high-speed master code. This 8-bit code is defined by the system designer and is set by writing to the High Speed Master Mode Code Address Register, HS_MADDR. Because the codes are unique, only one master can win arbitration, which occurs by the end of the transmission of the high-speed master code.

Control of the bus is determined by address or master code and data sent by competing masters, so there is no central master nor any order of priority on the bus.

Arbitration is not allowed between the following conditions:

- A RESTART condition and a data bit
- A STOP condition and a data bit
- A RESTART condition and a STOP condition
- Slaves are not involved in the arbitration process.

3.13.4.5 Clock Synchronization

When two or more masters try to transfer information on the bus at the same time, they must arbitrate and synchronize the SCL clock. All masters generate their own clock to transfer messages. Data is valid only during the high period of SCL clock. Clock synchronization is performed using the wired-AND connection to the SCL signal. When the master transitions the SCL clock to 0, the master starts counting the low time of the SCL clock and transitions the SCL clock signal to 1 at the beginning of the next clock period. However, if another master is holding the SCL line to 0, then the master goes into a HIGH wait state until the SCL clock line transitions to 1.

All masters then count off their high time, and the master with the shortest high time transitions the SCL line to 0. The masters then counts out their low time and the one with the longest low time forces the other master into a HIGH wait state. Therefore, a synchronized SCL clock is generated, which is illustrated in Figure 3-75. Optionally, slaves may hold the SCL line low to slow down the timing on the I2C bus.

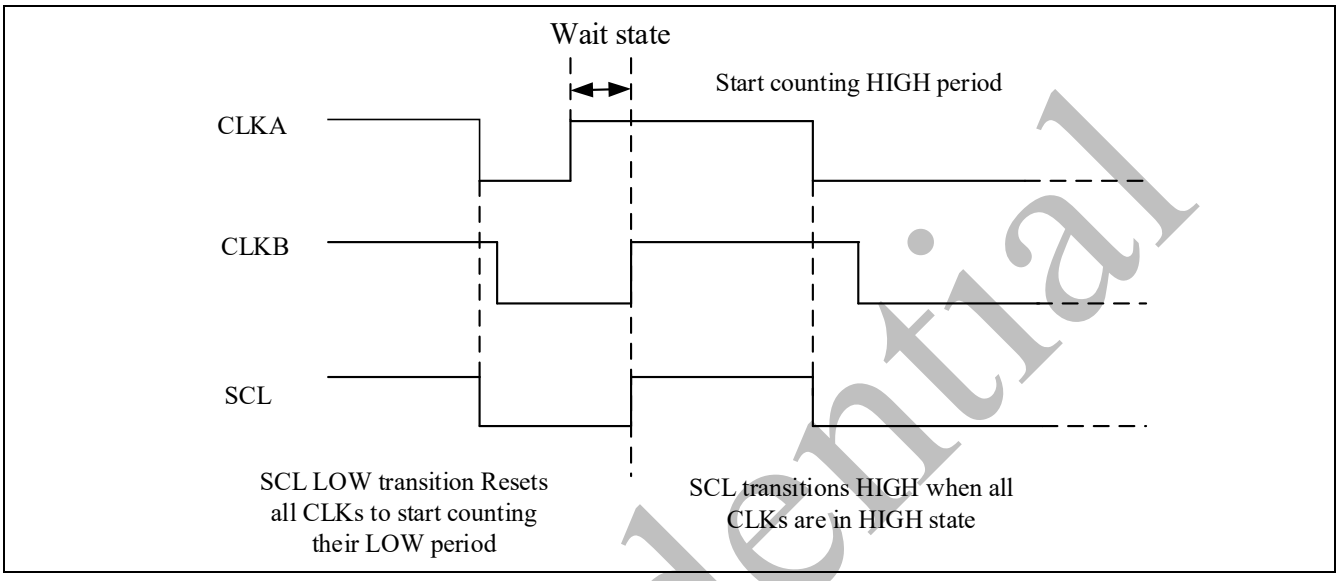


Figure 3-75 Multi-Master Clock Synchronization

3.13.4.6 Operation Modes

This section provides information on operation modes.

Slave Mode Operation

Initial Configuration

To use the I2C as a slave, perform the following steps:

1. Disable the I2C by writing a '0' to bit 0 of the I2C_EN register.
2. Write to the SLAVE_ADDR register (bits 9:0) to set the slave address. This is the address to which the I2C responds.
3. Write to the I2C_CNTRL register to specify which type of addressing is supported (7- or 10-bit by setting bit 3). Enable the I2C in slave-only mode by writing a '0' into bit 6 (SLAVE_DIS) and a '0' to bit 0 (MASTER_EN).
4. Enable the I2C by writing a '1' in bit 0 of the I2C_EN register.

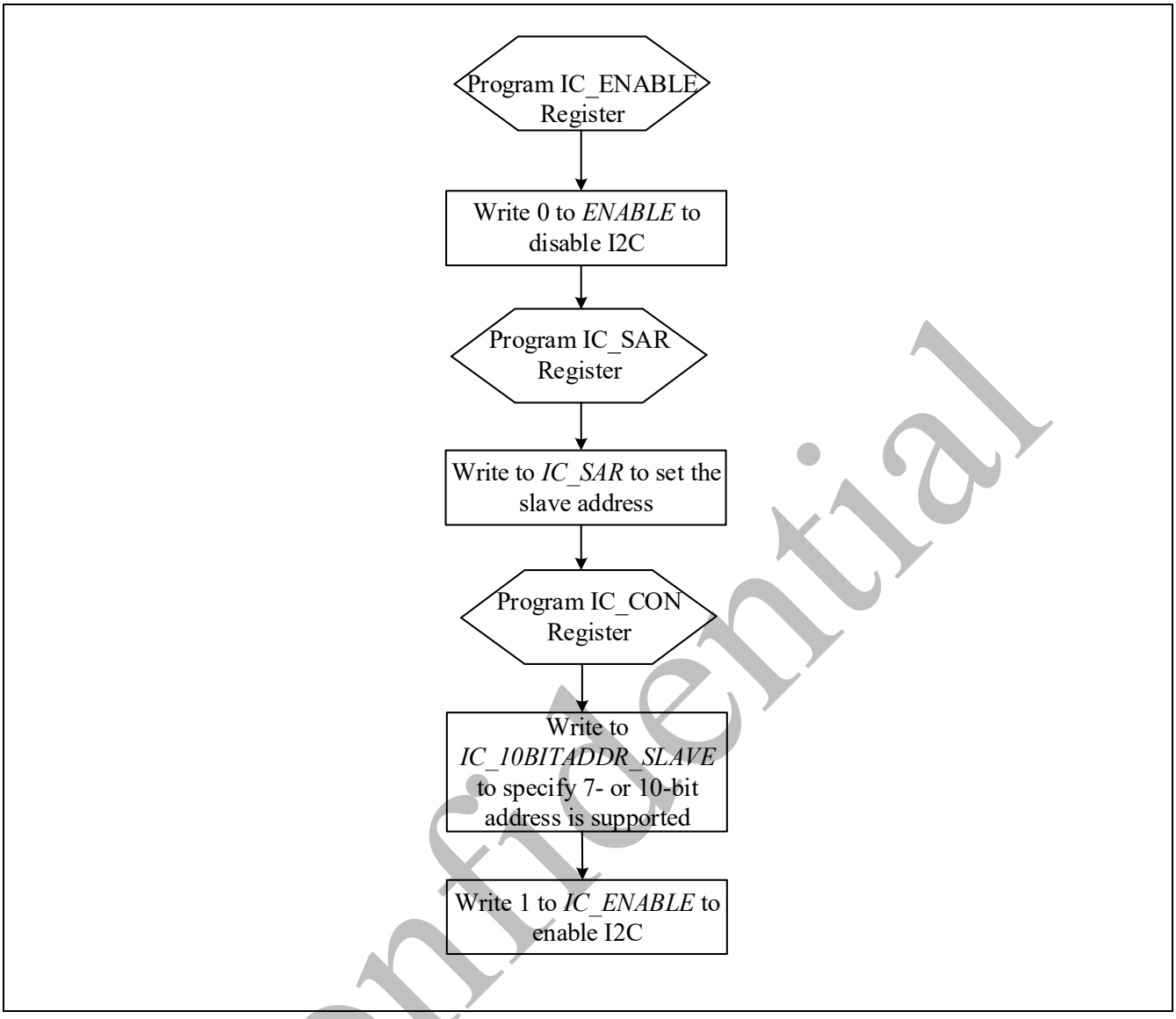


Figure 3-76 Initial Configuration

Slave-Transmitter Operation for a Single Byte

When another I2C master device on the bus addresses the I2C and requests data, the I2C acts as a slave-transmitter and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the slave address in the SLAVE_ADDR register of the I2C.
2. The I2C acknowledges the sent address and recognizes the direction of the transfer to indicate that it is acting as a slave-transmitter.
3. The I2C asserts the RD_REQ interrupt (bit 5 of the RAW_IRQ_STATUS register) and holds the SCL line low. It is in a wait state until software responds. If the RD_REQ interrupt has been masked, due to IRQ_MASK[5] register (M_RD_REQ bit field) being set to 0, then it is recommended that a hardware and/or software timing routine be used

to instruct the CPU to perform periodic reads of the RAW_IRQ_STATUS register.

a. Reads that indicate RAW_IRQ_STATUS[5] (R_RD_REQ bit field) being set to 1 must be treated as the equivalent of the RD_REQ interrupt being asserted.
b. Software must then act to satisfy the I2C transfer.
c. The timing interval used should be in the order of 10 times the fastest SCL clock period the I2C can handle. For example, for 400 kb/s, the timing interval is 25us.

4. If there is any data remaining in the Tx FIFO before receiving the read request, then the I2C asserts a TX_ABRT interrupt (bit 6 of the RAW_IRQ_STATUS register) to flush the old data from the TX FIFO.

   If the TX_ABRT interrupt has been masked, due to of IRQ_MASK[6] register (M_TX_ABRT bit field) being set to 0, then it is recommended that re-using the timing routine (described in the previous step), or a similar one, be used to read the RAW_IRQ_STATUS register.

   a. Reads that indicate bit 6 (R_TX_ABRT) being set to 1 must be treated as the equivalent of the TX_ABRT interrupt being asserted.
   b. There is no further action required from software.
   c. The timing interval used should be similar to that described in the previous step for the RAW_IRQ_STATUS[5] register.

5. Software writes to the DATA_CMD register with the data to be written (by writing a '0' in bit 8).
6. Software must clear the RD_REQ and TX_ABRT interrupts (bits 5 and 6, respectively) of the RAW_IRQ_STATUS register before proceeding. If the RD_REQ and/or TX_ABRT interrupts have been masked, then clearing of the RAW_IRQ_STATUS register will have already been performed when either the R_RD_REQ or R_TX_ABRT bit has been read as 1.
7. The I2C releases the SCL and transmits the byte.
8. The master may hold the I2C bus by issuing a RESTART condition or release the bus by issuing a STOP condition.

Slave-Receiver Operation for a Single Byte

When another I2C master device on the bus addresses the I2C and is sending data, the I2C acts as a slave-receiver and the following steps occur:

1. The other I2C master device initiates an I2C transfer with an address that matches the I2C's slave address in the SLAVE_ADDR register.

2. The I2C acknowledges the sent address and recognizes the direction of the transfer to indicate that the I2C is acting as a slave-receiver.
3. I2C receives the transmitted byte and places it in the receive buffer.
4. I2C asserts the RX_FULL interrupt (RAW_IRQ_STATUS[2] register). If the RX_FULL interrupt has been masked, due to setting IRQ_MASK[2] register to 0 or setting FIFO_TX_TSD to a value larger than 0, then it is recommended that a timing routine be implemented for periodic reads of the I2C_STATUS register. Reads of the I2C_STATUS register, with bit 3 (RX_FIFO_NE) set at 1, must then be treated by software as the equivalent of the RX_FULL interrupt being asserted.
5. Software may read the byte from the DATA_CMD register (bits 7:0).
6. The other master device may hold the I2C bus by issuing a RESTART condition, or release the bus by issuing a STOP condition.

Slave-Transfer Operation For Bulk Transfers

In the standard I2C protocol, all transactions are single byte transactions and the programmer responds to a remote master read request by writing one byte into the slave's TX FIFO. When a slave (slave-transmitter) is issued with a read request (RD_REQ) from the remote master (master-receiver), at a minimum there should be at least one entry placed into the slave-transmitter's TX FIFO. I2C is designed to handle more data in the TX FIFO so that subsequent read requests can take that data without raising an interrupt to get more data. Ultimately, this eliminates the possibility of significant latencies being incurred between raising the interrupt for data each time had there been a restriction of having only one entry placed in the TX FIFO. This mode only occurs when I2C is acting as a slave-transmitter. If the remote master acknowledges the data sent by the slave-transmitter and there is no data in the slave's TX FIFO, the I2C holds the I2C SCL line low while it raises the read request interrupt (RD_REQ) and waits for data to be written into the TX FIFO before it can be sent to the remote master.

If the RD_REQ interrupt is masked, due to bit 5 (M_RD_REQ) of the IRQ_STATUS register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the RAW_IRQ_STATUS register. Reads of RAW_IRQ_STATUS that return bit 5 (R_RD_REQ) set to 1 must be treated as the equivalent of the RD_REQ interrupt referred to in this section. T

The RD_REQ interrupt is raised upon a read request, and like interrupts, must be cleared when exiting the interrupt service handling routine (ISR). The ISR allows you to either write 1 byte or more than 1 byte into the Tx FIFO. During the transmission of these bytes to the

master, if the master acknowledges the last byte. then the slave must raise the RD_REQ again because the master is requesting for more data.

If the programmer knows in advance that the remote master is requesting a packet of n bytes, then when another master addresses I2C and requests data, the Tx FIFO could be written with n number bytes and the remote master receives it as a continuous stream of data. For example, the I2C slave continues to send data to the remote master as long as the remote master is acknowledging the data sent and there is data available in the Tx FIFO. There is no need to hold the SCL line low or to issue RD_REQ again.

If the remote master is to receive n bytes from the I2C but the programmer wrote a number of bytes larger than n to the Tx FIFO, then when the slave finishes sending the requested n bytes, it clears the Tx FIFO and ignores any excess bytes.

The I2C generates a transmit abort (TX_ABRT) event to indicate the clearing of the Tx FIFO in this example. At the time an ACK/NACK is expected, if a NACK is received, then the remote master has all the data it wants. At this time, a flag is raised within the slave's state machine to clear the leftover data in the Tx FIFO. This flag is transferred to the processor bus clock domain where the FIFO exists and the contents of the Tx FIFO is cleared at that time.

Master Mode Operation

Initial Configuration

The initial configuration procedure for Master Mode Operation depends on the configuration parameter I2C_DYNAMIC_TAR_UPDATE. When set to "Yes" (1), the target address and address format can be changed dynamically without having to disable I2C. This parameter only applies to when I2C is acting as a master because the slave requires the component to be disabled before any changes can be made to the address.

The procedures are very similar and are only different with regard to where the ADDR_MASTER bit is set (either bit 4 of I2C_CNTRL register or bit 12 of TAR_ADDR register).

To use the I2C as a master when the I2C_DYNAMIC_TAR_UPDATE configuration parameter is set to "Yes" (1), perform the following steps:

1. Disable the I2C by writing 0 to bit 0 of the I2C_EN register.
2. Write to the I2C_CNTRL register to set the maximum speed mode supported for slave operation (bits2:1) and to specify whether the I2C starts its transfers in 7/10 bit addressing mode when the device is a slave (bit 3).

3. Write to the TAR_ADDR register the address of the I2C device to be addressed. It also indicates whether a General Call or a START BYTE command is going to be performed by I2C. The desired speed of the I2C master-initiated transfers, either 7-bit or 10-bit addressing, is controlled by the ADDR_MASTER bit field (bit 12).
4. Only applicable for high-speed mode transfers. Write to the HS_MADDR register the desired master code for the I2C. The master code is programmer-defined.
5. Enable the I2C by writing a 1 to bit 0 of the I2C_EN register.
6. Now write the transfer direction and data to be sent to the DATA_CMD register. If the DATA_CMD register is written before the I2C is enabled, the data and commands are lost as the buffers are kept cleared when I2C is not enabled.

Dynamic TAR_ADDR or ADDR_MASTER Update

The I2C supports dynamic updating of the TAR_ADDR (bits 9:0) and ADDR_MASTER (bit 12) bit fields of the TAR_ADDR register. In order to perform a dynamic update of the TAR_ADDR register, the I2C_DYNAMIC_TAR_UPDATE configuration parameter must be set to Yes (1). You can dynamically write to the TAR_ADDR register provided the software ensures that there are no other commands in the Tx FIFO that use the existing TAR address. If the software does not ensure this, then TAR_ADDR should be re-programmed only if the following conditions are met:

- I2C is not enabled (I2C_EN[0]=0);

OR

- I2C is enabled (I2C_EN[0]=1); AND
- I2C is NOT engaged in any Master (tx, rx) operation (I2C_STATUS[5]=0);

AND

- I2C is enabled to operate in Master mode (I2C_CNTRL[0]=1);

AND

- There are NO entries in the Tx FIFO (I2C_STATUS[2]=1)

Master Transmit and Master Receive

The I2C supports switching back and forth between reading and writing dynamically. To transmit data, write the data to be written to the lower byte of the I2C Rx/Tx Data Buffer and Command Register (DATA_CMD). The RNW bit [8] should be written to 0 for I2C write operations. Subsequently, a read command may be issued by writing “don’t cares” to the lower byte of the DATA_CMD register, and a 1 should be written to the RNW bit. The I2C master continues to initiate transfers as long as there are commands present in the transmit FIFO. If

the transmit FIFO becomes empty—depending on the value of IC_EMPTYFIFO_HOLD_MASTER_EN, the master either inserts a STOP condition after completing the current transfers, or it checks to see if DATA_CMD[9] is set to 1.

- If set to 1, it issues a STOP condition after completing the current transfer.
- If set to 0, it holds SCL low until next command is written to the transmit FIFO.

Disabling I2C

The register I2C_EN_STATUS is added to allow software to unambiguously determine when the hardware has completely shut down in response to bit 0 of the I2C_EN register being set from 1 to 0.

1. Define a timer interval (ti2c_poll) equal to the 10 times the signaling period for the highest I2C transfer speed used in the system and supported by I2C. For example, if the highest I2C transfer mode is 400 kb/s, then this ti2c_poll is 25us.
2. Define a maximum time-out parameter, MAX_T_POLL_COUNT, such that if any repeated polling operation exceeds this maximum value, an error is reported.
3. Execute a blocking thread/process/function that prevents any further I2C master transactions to be started by software, but allows any pending transfers to be completed.
4. The variable POLL_COUNT is initialized to zero.
5. Set bit 0 of the I2C_EN register to 0.
6. Read the I2C_EN_STATUS register and test the EN_STATUS bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT ≥ MAX_T_POLL_COUNT, exit with the relevant error code.
7. If I2C_EN_STATUS[0] is 1, then sleep for ti2c_poll and proceed to the previous step. Otherwise, exit with a relevant success code.

Aborting I2C Transfers

The ABORT control bit of the I2C_EN register allows the software to relinquish the I2C bus before completing the issued transfer commands from the Tx FIFO. In response to an ABORT request, the controller issues the STOP condition over the I2C bus, followed by Tx FIFO flush. Aborting the transfer is allowed only in master mode of operation.

1. Stop filling the Tx FIFO (DATA_CMD) with new commands.
2. When operating in DMAC mode, disable the transmit DMAC by setting TDMACE to 0.
3. Set bit 1 of the I2C_EN register (ABORT) to 1.
4. Wait for the M_TX_ABRT interrupt.

5. Read the TX_ASR register to identify the source as MASTER_TRANS_ABORT.

3.13.4.7 Spike Suppression

The I2C contains programmable spike suppression logic that match requirements imposed by the I2C Bus Specification for SS/FS and HS modes.

This logic is based on counters that monitor the input signals (SCL and SDA), checking if they remain stable for a predetermined amount of i2c_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of i2c_clk cycles can be programmed by the user and should be calculated taking into account the frequency of i2c_clk and the relevant spike length specification.

Each counter is started whenever its input signal changes its value. Depending on the behavior of the input signal, one of the following scenarios occurs:

- The input signal remains unchanged until the counter reaches its count limit value. When this happens, the internal version of the signal is updated with the input value, and the counter is reset and stopped. The counter is not restarted until a new change on the input signal is detected.
- The input signal changes again before the counter reaches its count limit value. When this happens, the counter is reset and stopped, but the internal version of the signal is not updated. The counter remains stopped until a new change on the input signal is detected.

The timing diagram in Figure 3-77 illustrates the behavior described above.

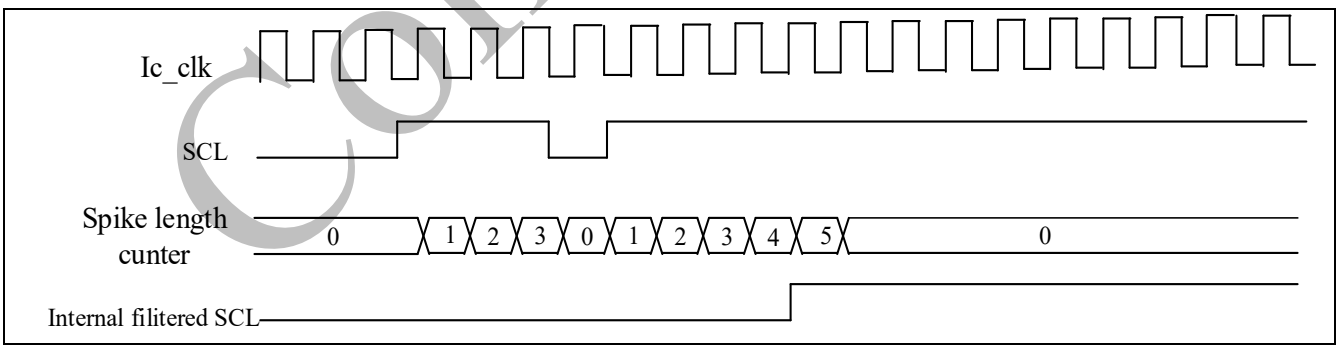


Figure 3-77 Spike Suppression Example

The count limit value used in this example is 5 and was calculated for a 10ns i2c_clk period and for SS/FS operation (50 ns spike suppression).

The I2C Bus Specification calls for different maximum spike lengths according to the operating mode—50 ns for SS and FS; 10 ns for HS, so two registers are required to store the values needed for each case:

- Register SF_SSLR holds the maximum spike length for SS and FS/FM+ modes
- Register HS_SSLR holds the maximum spike value for HS mode.

These registers are 8 bits wide and accessible through the APB interface for read and write purposes; however, they can be written to only when the I2C is disabled. The minimum value that can be programmed into these registers is 1; attempting to program a value smaller than 1 results in the value 1 being written.

3.13.4.8 Fast Mode Plus Operation

In fast mode plus, the I2C allows the fast mode operation to be extended to support speeds up to 1000 Kb/s. To enable the I2C for fast mode plus operation, perform the following steps before initiating any data transfer:

1. Configure the Maximum Speed mode of I2C Master or Slave to Fast Mode or High Speed mode.
2. I2c_clk frequency should greater than or equal to 32 MHz
3. Program the I2C_CNTRL register [2:1] = 2'b10 for fast mode or fast mode plus.
4. Program FSCL_LCOUNT and FSCL_HCOUNT registers to meet the fast mode plus SCL.
5. Program the SF_SSLR register to suppress the maximum spike of 50ns.
6. Program the SDA_SR register to meet the minimum data setup time (tSU; I2C_DATA)

3.13.4.9 IC_CLK Frequency Configuration

When the I2C is configured as a Standard (SS), Fast (FS)/Fast-Mode Plus (FM+), or High Speed (HS) master, the *CNT registers must be set before any I2C bus transaction can take place in order to ensure proper I/O timing.

When the I2C operates as an I2C master, in both transmit and receive transfers:

- SSCL_LCOUNT and FSCL_LCOUNT register values must be larger than SF_SSLR+7.
- SSCL_HCOUNT and FSCL_HCOUNT register values must be larger than SF_SSLR+5.
- If the component is programmed to support HS, HSCL_LCOUNT register value must be larger than HS_SSLR + 7.
- If the component is programmed to support HS, HSCL_HCOUNT register value must be larger than HS_SSLR + 5.

Details regarding the I2C high and low counts are as follows:

- The minimum value of IC_*_SPKLEN + 7 for the *_LCNT registers is due to the time required for the I2C to drive SDA after a negative edge of SCL.

- The minimum value of IC_*_SPKLEN + 5 for the *_HCNT registers is due to the time required for the I2C to sample SDA during the high period of SCL.
- The I2C adds one cycle to the programmed *_LCNT value in order to generate the low period of the SCL clock; this is due to the counting logic for SCL low counting to (*_LCNT + 1).
- The I2C adds IC_*_SPKLEN + 7 cycles to the programmed *_HCNT value in order to generate the high period of the SCL clock; this is due to the following factors:
  - The counting logic for SCL high counts to (*_HCNT+1).
  - The digital filtering applied to the SCL line incurs a delay of SPKLEN + 2 ic_clk cycles, where SPKLEN is:
    - SF_SSLR if the component is operating in SS or FS
    - HS_SSLR if the component is operating in HS.
  - This filtering includes meta-stability removal and the programmable spike suppression on SDA and SCL edges.
  - Whenever SCL is driven 1 to 0 by the I2C—that is, completing the SCL high time—an internal logic latency of three i2c_clk cycles is incurred. Consequently, the minimum SCL low time of which the I2C is capable is nine (9) i2c_clk periods (7 + 1 + 1), while the minimum SCL high time is thirteen (13) i2c_clk periods (6 + 1 + 3 + 3).

3.13.4.10 SDA Hold Time

The I2C protocol specification requires 300ns of hold time on the SDA signal (tHD;I2C_DATA) in standard mode and fast mode, and a hold time long enough to bridge the undefined part between logic 1 and logic 0 of the falling edge of SCL in high speed mode and fast mode plus.

The I2C contains a software programmable register (SDA_HTLR) to enable dynamic adjustment of the SDA hold-time. The SDA_HTLR register can be programmed only when the I2C is disabled (I2C_EN[0] = 0).

SDA Hold Timings in Receiver

When I2C acts as a receiver, according to the I2C protocol, the device should internally hold the SDA line to bridge undefined gap between logic 1 and logic 0 of SCL.

SDA_RX_HTLR can be used to alter the internal hold time which I2C applies to the incoming SDA line. Each value in the SDA_RX_HTLR register represents a unit of one i2c_clk period. The minimum value of SDA_RX_HTLR is 0. This hold time is applicable only when SCL is

HIGH. The receiver does not extend the SDA after SCL goes LOW internally.

Figure 3-78 shows the I2C as receiver with SDA_RX_HTLR programmed to greater than or equal to 3.

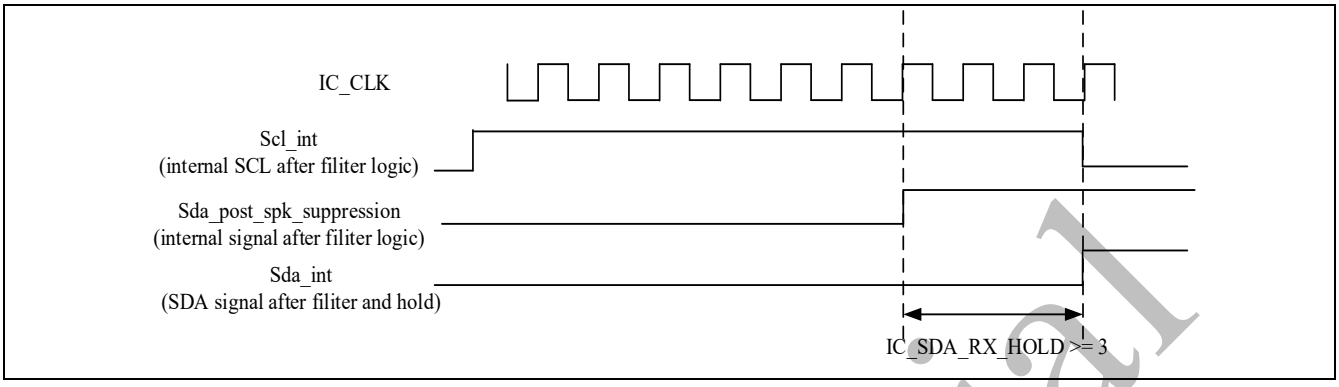


Figure 3-78 I2C receiver with SDA_RX_HTLR

If SDA_RX_HTLR is greater than 3, I2C does not hold SDA beyond 3 ic_clk cycles, because SCL goes LOW internally.

The maximum values of SDA_RX_HTLR that can be programmed in the register for the respective speed modes are derived from the equations show in Table 3-15.

Table 3-15 Maximum Values for SDA_RX_HTLR

Speed Mode	Maximum SDA_RX_HTLR Value
Standard Mode	SSCL_HCOUNT – SF_SSLR – 3
Fast Mode or Fast Mode Plus	FSCL_HCOUNT – SF_SSLR – 3
High Speed	Min {FSCL_HCOUNT – SF_SSLR – 3 , HSCL_LCOUNT – HS_SSLR – 3}

SDA Hold Timings in Transmitter

The SDA_TX_HTLR register can be used to alter the timing of the generated SDA signal by the I2C. Each value in the SDA_TX_HTLR register represents a unit of one ic_clk period. When the I2C is operating in Master Mode, the minimum tHD:I2C_DATA timing is one i2c_clk period. Therefore even when SDA_TX_HTLR has a value of zero, the I2C will drive SDA one i2c_clk cycle after driving SCL to logic 0. For all other values of SDA_TX_HTLR, the following is true:

- Drive on SDA occurs SDA_TX_HTLR i2c_clk cycles after driving SCL to logic 0
  - When the I2C is operating in Slave Mode, the minimum tHD:I2C_DATA timing is SPKLEN + 7 i2c_clk periods, where SPKLEN is:
- SF_SSLR if the component is operating in standard mode, fast mode, or fast mode plus
- HS_SSLR if the component is operating in high speed mode

- This delay allows for synchronization and spike suppression on the SCL sample. Therefore, even when SDA_TX_HTLR has a value less than SPKLEN + 7, the I2C drives SDA SPKLEN + 7 i2c_clk cycles after SCL has transitioned to logic 0. For all other values of SDA_TX_HTLR, the following is true:

• Drive on SDA occurs SDA_TX_HTLR i2c_clk cycles after SCL has transitioned to logic 0

- Figure 3-79 shows the tHD:I2C_DATA timing generated by the I2C operating in Master Mode when SDA_TX_HTLR = 3.

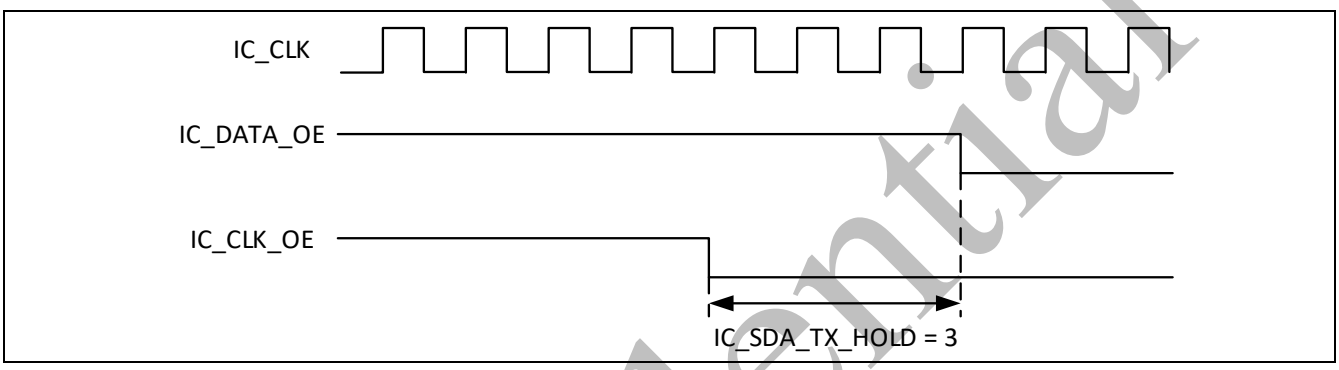


Figure 3-79 I2C Master Implementing tHD;I2C_DATA with SDA_HTLR = 3

3.13.4.11 DMA Controller Interface

The I2C has an optional built-in DMA capability that can be selected at configuration time; it has a handshaking interface to a DMA Controller to request and control transfers. The APB bus is used to perform the data transfer to or from the DMA. The specific programing flowchart is shown in Figure 3-80.

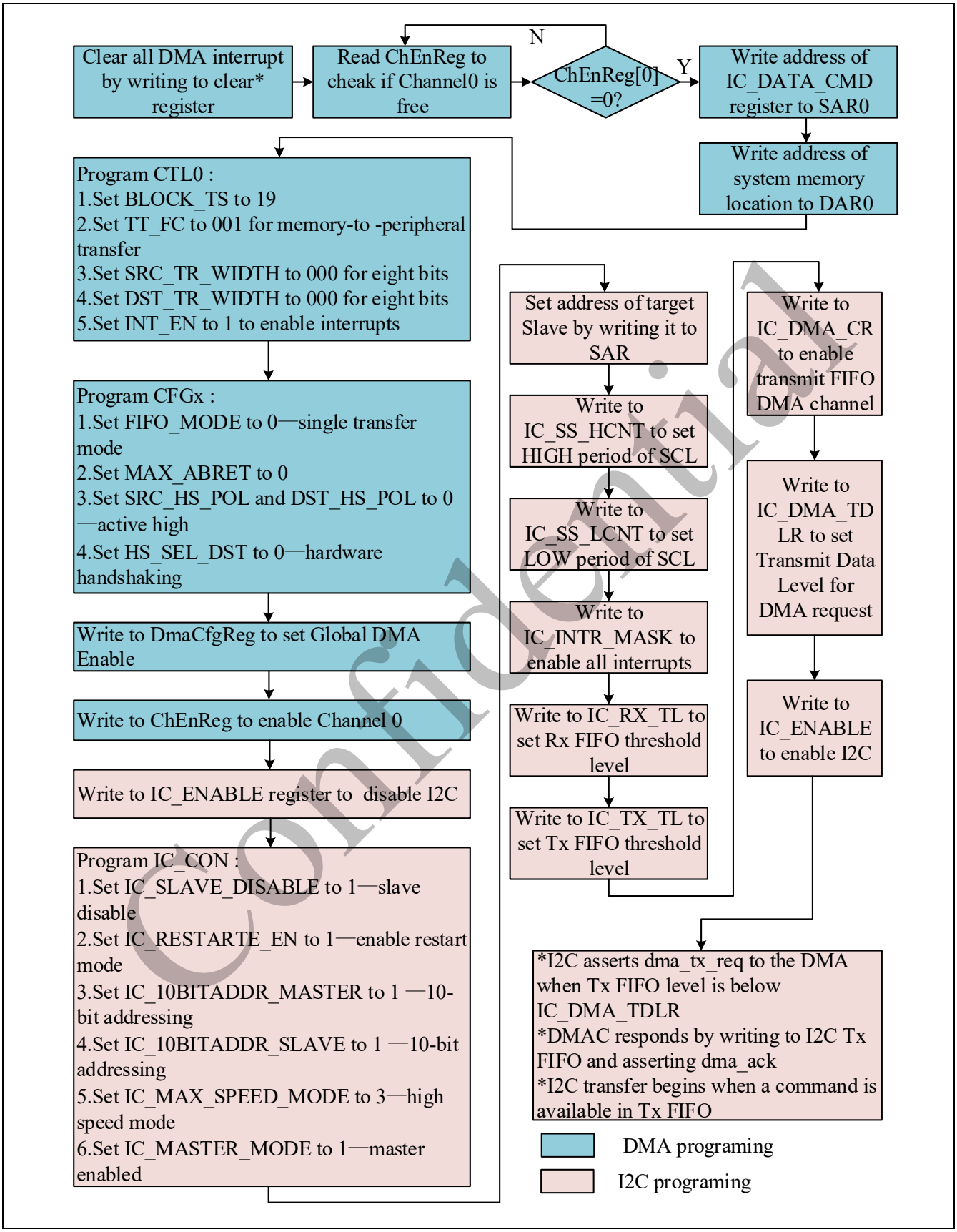


Figure 3-80 Flowchart for DMA and I2C Programing

3.13.5 I2C Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
I2C Base Address: I2C_BA = 0×4000_0000				
I2C_CNTRL	I2C_BA+ 0×00	R/W	I2C Control Register	0×0000_007F
TAR_ADDR	I2C_BA+ 0×04	R/W	I2C Target Address Register	0×0000_1055
SLAVE_ADDR	I2C_BA+0×08	R/W	I2C Slave Address Register	0×0000_0055
HS_MADDR	I2C_BA+0×0C	R/W	I2C High Speed Master Mode Code Address Register	0×0000_0001
DATA_CMD	I2C_BA+0×10	R/W	I2C Rx/Tx Data Buffer and Command Register	0×0000_0000
SSCL_HCOUNT	I2C_BA+0×14	R/W	Standard Speed I2C Clock SCL High Count Register	0×0000_0190
SSCL_LCOUNT	I2C _BA+0×18	R/W	Standard Speed I2C Clock SCL Low Count Register	0×0000_01D6
FSCL_HCOUNT	I2C_BA+0×1C	R/W	Fast Speed I2C Clock SCL High Count Register	0×0000_003C
FSCL_LCOUNT	I2C_BA+0×20	R/W	Fast Speed I2C Clock SCL Low Count Register	0×0000_0082
HSCL_HCOUNT	I2C_BA+0×24	R/W	High Speed I2C Clock SCL High Count Register	0×0000_0006
HSCL_LCOUNT	I2C_BA+0×28	R/W	High Speed I2C Clock SCL Low Count Register	0×0000_0010
IRQ_STATUS	I2C_BA+0×2C	R	I2C Interrupt Status Register	0×0000_0000
IRQ_MASK	I2C_BA+0×30	R/W	I2C Interrupt Mask Register	0×0000_08FF
RAW_IRQ_STATUS	I2C_BA+0×34	R	I2C Raw Interrupt Status Register	0×0000_0000
FIFO_RX_TSD	I2C_BA+0×38	R/W	I2C Receive FIFO Threshold Register	0×0000_0000
FIFO_TX_TSD	I2C_BA+0×3C	R/W	I2C Transmit FIFO Threshold Register	0×0000_0000
CLR_IRQ	I2C_BA+ 0×40	R	Clear Combined and Individual Interrupt Register	0×0000_0000
CLR_RX_UNDER	I2C_BA+0×44	R	Clear RX_UNDER Interrupt	0×0000_0000
CLR_RX_OVER	I2C_BA+0×48	R	Clear RX_OVER Interrupt	0×0000_0000
CLR_TX_OVER	I2C_BA+0×4C	R	Clear TX_OVER Interrupt	0×0000_0000
CLR_RD_REQ	I2C_BA+0×50	R	Clear RD_REQ Interrupt	0×0000_0000
CLR_TX_ABRT	I2C_BA+0×54	R	Clear TX_ABRT Interrupt	0×0000_0000
CLR_RX_DONE	I2C_BA+0×58	R	Clear RX_DONE Interrupt	0×0000_0000
CLR_ACTIVITY	I2C_BA+0×5C	R	Clear ACTIVITY Interrupt	0×0000_0000
CLR_STOP_CHECK	I2C_BA+0×60	R	Clear STOP_CHECK Interrupt	0×0000_0000
CLR_START_CHECK	I2C_BA+0×64	R	Clear START_CHECK Interrupt	0×0000_0000
CLR_GENERAL_CALL	I2C_BA+0×68	R	Clear GENERAL_CALL Interrupt	0×0000_0000

I2C_EN	I2C_BA+0×6C	R/W	I2C Enable Register	0×0000_0000
I2C_STATUS	I2C_BA+0×70	R	I2C Status Register	0×0000_0006
TX_FIFO_LR	I2C_BA+0×74	R	Transmit FIFO Level Register	0×0000_0000
RX_FIFO_LR	I2C_BA+0×78	R	Receive FIFO Level Register	0×0000_0000
SDA_HTLR	I2C_BA+0×7C	R/W	SDA Hold Time Length Register	0×0000_0001
TX_ASR	I2C_BA+0×80	R	I2C Transmit Abort Status Register	0×0000_0000
SLAVE_NACK	I2C_BA+0×84	R/W	Generate SLV_DATA_NACK Register	0×0000_0000
DMA_CNTRL	I2C_BA+0×88	R/W	DMA Control Register for transmit and receive	0×0000_0000
DMA_TX_DLR	I2C_BA+0×8C	R/W	DMA Transmit Data Level	0×0000_0000
DMA_RX_DLR	I2C_BA+0×90	R/W	DMA Receive Data Level	0×0000_0000
SDA_SR	I2C_BA+0×94	R/W	I2C SDA Setup Register	0×0000_0064
ACK_GENERAL_CALL	I2C_BA+0×98	R/W	I2C ACK General Call Register	0×0000_0001
I2C_EN_STATUS	I2C_BA+0×9C	R	I2C Enable Status Register	0×0000_0000
SF_SSLR	I2C_BA+0×A0	R/W	I2C SS and FS Spike Suppression Limit Register	0×0000_0005
HS_SSLR	I2C_BA+0×A4	R/W	I2C HS Spike Suppression Limit Register	0×0000_0001
CLR_RESTART_CHECK	I2C_BA+0×A8	R	Clear RESTART_CHECK Interrupt Register	0×0000_0000
SCL_SLT	I2C_BA+0×AC	R/W	I2C SCL Stuck at Low Timeout	0×FFFF_FFFF
SCL_SLDIR	I2C_BA+0×B4	R	Clear SCL Stuck at Low Detect Interrupt Register	0×0000_0000

3.13.6 Operation of Interrupt Registers

Table 3-16 lists the operation of the I2C interrupt registers and how they are set and cleared. Some bits are set by hardware and cleared by software, whereas other bits are set and cleared by hardware.

Table 3-16 Operation of the Interrupt Register

Interrupt Bit Fields	Set by Hardware/ Cleared by Software	Set and Cleared by Hardware
MASTER_HOLD	×	√
RESTART_CHECK	√	×
GENERAL_CALL	√	×
START_CHECK	√	×
STOP_CHECK	√	×
ACTIVITY	√	×
RX_DONE	√	×
TX_ABRT	√	×
RD_REQ	√	×
TX_EMPTY	×	√
TX_OVER	√	×
RX_FULL	×	√
RX_OVER	√	×
RX_UNDER	√	×

3.13.7 I2C Register Description

This section describes the registers listed in Table 6-1. Registers are on the HCLK domain, but status bits reflect actions that occur in the i2c_clk domain. Therefore, there is delay when the HCLK register reflects the activity that occurred on the i2c_clk side.

Some registers may be written only when the I2C is disabled, programmed by the I2C_EN register. Software should not disable the I2C while it is active. If the I2C is in the process of transmitting when it is disabled, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. The slave continues receiving until the remote master aborts the transfer, in which case the I2C could be disabled. Registers that cannot be written to when the I2C is enabled are indicated in their descriptions.

3.13.7.1 I2C_CNTRL

This register can be written only when the I2C is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect.

Register	Offset	R/W	Description	Reset Value
I2C_CNTRL	I2C_BA+0×00	R/W	I2C Control Register	0×0000_007F

Bits	Description	
[31:8]	Reserved	Reserved.
[7]	STOP_CHECK_IFSOLVED	In slave mode: 1'b1 – issues the STOP_CHECK interrupt only when it is addressed. 1'b0 – issues the STOP_CHECK irrespective of whether it's addressed or not. Dependencies: This register bit value is applicable in the slave mode only (MASTER_EN = 1'b0) Note: During a general call address, this slave does not issue the STOP_CHECK interrupt if STOP_CHECK_IFSOLVED = 1'b1, even if the slave responds to the general call address by generating ACK. The STOP_CHECK interrupt is generated only when the transmitted address matches the slave address (SAR).
[6]	SLAVE_DIS	This bit controls whether I2C has its slave disabled, which means once the preset_n signal is applied, then this bit takes on the value of the configuration parameter SLAVE_DIS. You have the choice of having the slave enabled or disabled after reset is applied, which means software does not have to configure the slave. By default, the slave is always enabled (in reset state as well). If you need to disable it after reset, set this bit to 1. If this bit is set (slave is disabled), I2C functions only as a master and does not perform any action that requires a slave. 0 = slave is enabled 1 = slave is disabled Note: Software should ensure that if this bit is written with'0,' then bit 0 should

		also be written with a '0'.
[5]	RESTART_EN	Determines whether RESTART conditions may be sent when acting as a master. Some older slaves do not support handling RESTART conditions; however, RESTART conditions are used in several I2C operations. 0 = disable 1 = enable When the RESTART is disabled, the I2C master is incapable of performing the following functions: Sending a START BYTE Performing any high-speed mode operation Performing direction changes in combined format mode Performing a read operation with a 10-bit address By replacing RESTART condition followed by a STOP and a subsequent START condition, split operations are broken down into multiple I2C transfers. If the above operations are performed, it will result in setting bit 6 (TX_ABRT) of the RAW_IRQ_STATUS register.
[4]	ADDR_MASTER_R	The function of this bit is handled by bit 12 of TAR_ADDR register, and becomes a read-only copy called ADDR_MASTER_R. 0 = 7-bit addressing 1 = 10-bit addressing
[3]	ADDR_SLAVE	When acting as a slave, this bit controls whether the I2C responds to 7- or 10-bit addresses. 0 = 7-bit addressing. The I2C ignores transactions that involve 10-bit addressing; for 7-bit addressing, only the lower 7 bits of the SLAVE_ADDR register are compared. 1 = 10-bit addressing. The I2C responds to only10-bit addressing transfers that match the full 10 bits of the SLAVE_ADDR register.
[2:1]	SPEED_SEL	These bits control at which speed the I2C operates. Its setting is relevant only if one is operating the I2C in master mode. Hardware protects against illegal values being programmed by software. These bits must be programmed appropriately for slave mode also, as it is used to capture correct value of spike filter as per the speed mode. This register should be programmed only with a value in the range of 1 to 3; otherwise, hardware updates this register with the value of 3. 1 = standard mode (0 to 100 Kb/s) 2 = fast mode (≤ 400 Kb/s) or fast mode plus (≤ 1000 Kb/s) 3 = high speed mode (≤ 3.4 Mb/s)
[0]	MASTER_EN	This bit controls whether the I2C master is enabled. 0 = master disabled 1 = master enabled Note: Software should ensure that if this bit is written with'1', then bit 6 should also be written with a '1'.

3.13.7.2 TAR_ADDR

Register	Offset	R/W	Description	Reset Value
TAR_ADDR	I2C_BA+0×04	R/W	I2C Target Address Register	0×0000_1055

Bits	Description	
[31:13]	Reserved	Reserved.
[12]	ADDR_MASTER	This bit controls whether the I2C starts its transfers in 7- or 10-bit addressing mode when acting as a master. 0 = 7-bit addressing 1 = 10-bit addressing
[11]	IFEXECUTE	This bit indicates whether software performs a Device-ID, General Call or START BYTE command. 0 = ignore bit 10 CALL_START and use TAR_ADDR normally 1 = perform special I2C command as specified in CALL_START bit
[10]	CALL_START	If bit 11 (IFEXECUTE) is set to 1, then this bit indicates whether a General Call or START BYTE command is to be performed by the I2C. 0 = General Call Address – after issuing a General Call, only writes may be performed. Attempting to issue a read command results in setting bit 6 (TX_ABRT) of the RAW_IRQ_STATUS register. The I2C remains in General Call mode until the IFEXECUTE bit value (bit 11) is cleared. 1 = START BYTE
[9:0]	TAR_ADDR	This is the target address for any master transaction. When transmitting a General Call, these bits are ignored. To generate a START BYTE, the CPU needs to write only once into these bits. If the TAR_ADDR and SLAVE_ADDR are the same, loop-back exists but the FIFOs are shared between master and slave, so full loop-back is not feasible. Only one direction loop-back mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only slave.

3.13.7.3 SLAVE_ADDR

Register	Offset	R/W	Description	Reset Value
SLAVE_ADDR	I2C_BA+0×08	R/W	I2C Slave Address Register	0×0000_0055

Bits	Descriptions	
[31:10]	Reserved	Reserved.
[9:0]	SLAVE_ADDR	The SLAVE_ADDR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only SLAVE_ADDR [6:0] is used. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN [0] being set to 0. Writes at other times have no effect. Note: The default values cannot be any of the reserved address locations: that is, 0×00 to 0×07, or 0×78 to 0×7F. The correct operation of the device is not guaranteed if you program the SLAVE_ADDR or TAR_ADDR to a reserved value.

3.13.7.4 HS_MADDR

Register	Offset	R/W	Description	Reset Value
HS_MADDR	I2C_BA+0×0c	R/W	I2C High Speed Master Mode Code Address Register	0×0000_0001

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2:0]	HS_MADDR	This bit field holds the value of the I2C HS mode master code. HS-mode master codes are reserved 8-bit codes (00001xxx) that are not used for slave addressing or other purposes. Each master has its unique master code; up to eight high speed mode masters can be present on the same I2C bus system. Valid values are from 0 to 7. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect.

3.13.7.5 DATA_CMD

Register	Offset	R/W	Description	Reset Value
DATA_CMD	I2C_BA+0×10	R/W	I2C Rx/Tx Data Buffer and Command Register	0×0000_0000

Bits	Descriptions	
[31:11]	Reserved	Reserved.
[10]	RESTART	This bit controls whether a RESTART is issued before the byte is sent or received. (only writable) 1 = If RESTART_EN is 1, a RESTART is issued before the data is sent/received (according to the value of RNW), regardless of whether or not the transfer direction is changing from the previous command. 0 = If RESTART_EN is 1, a RESTART is issued only if the transfer direction is changing from the previous command.
[9]	STOP	This bit controls whether a STOP is issued after the byte is sent or received. (only writable) 1 = STOP is issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master immediately tries to start a new transfer by issuing a START and arbitrating for the bus. 0 = STOP is not issued after this byte, regardless of whether or not the Tx FIFO is empty. If the Tx FIFO is not empty, the master continues the current transfer by sending/receiving data bytes according to the value of the RNW bit. If the Tx FIFO is empty, the master holds the SCL line low and stalls the bus until a new command is available in the Tx FIFO.
[8]	RNW	This bit controls whether a read or a write is performed. (only writable) This bit does not control the direction when the I2C acts as a slave. It controls only the direction when it acts as a master. 1 = Read 0 = Write When a command is entered in the TX FIFO, this bit distinguishes the write and read commands. In slave-receiver mode, this bit is a “don’t care” because writes to this register are not required. In slave-transmitter mode, a “0” indicates that the data in DATA_CMD is to be transmitted. When programming this bit, you should remember the following: attempting to perform a read operation after a General Call command has been sent results in a TX_ABRT interrupt (bit 6 of the RAW_IRQ_STATUS register), unless bit 11 (IFEXECUTE) in the TAR_ADDR register has been cleared. If a “1” is written to this bit after receiving a RD_REQ interrupt, then a TX_ABRT interrupt occurs.
[7:0]	I2C_DATA	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 (I2C_DATA) are ignored by the I2C. However, when you read this register, these bits return the value of data received on the I2C interface.

3.13.7.6 SSCL_HCOUNT

Register	Offset	R/W	Description	Reset Value
SSCL_HCOUNT	I2C_BA+0×14	R/W	Standard Speed I2C Clock SCL High Count Register	0×0000_0190

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[15:0]	SSCL_HCOUNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for standard speed. For more information, refer to *IC_CLK Frequency Configuration*. This register can be written only when the I2C interface is disabled which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set. **Note**: This register must not be programmed to a value higher than 65525,because I2C uses a 16-bit counter to flag an I2C bus idle condition when this counter reaches a value of SSCL_HCOUNT + 10.

3.13.7.7 SSCL_LCOUNT

Register	Offset	R/W	Description	Reset Value
SSCL_LCOUNT	I2C_BA+0×18	R/W	Standard Speed I2C Clock SCL Low Count Register	0×0000_01D6

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[15:0]	SSCL_LCOUNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for standard speed. For more information, refer to *IC_CLK Frequency Configuration*. This register can be written only when the I2C interface is disabled which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted, results in 8 being set. .

3.13.7.8 FSCL_HCOUNT

Register	Offset	R/W	Description	Reset Value
FSCL_HCOUNT	I2C_BA+0×1C	R/W	Fast Mode or Fast Mode Plus I2C Clock SCL High Count Register	0×0000_003C

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[15:0]	FSCL_HCOUNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high-period count for fast mode or fast mode plus. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information,refer to *IC_CLK Frequency Configuration*. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.

3.13.7.9 FSCL_LCOUNT

Register	Offset	R/W	Description	Reset Value
FSCL_LCOUNT	I2C_BA+0×20	R/W	Fast Mode or Fast Mode Plus I2C Clock SCL Low Count Register	0×0000_0082

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[15:0]	FSCL_LCOUNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for fast mode or fast mode plus. It is used in high-speed mode to send the Master Code and START BYTE or General CALL. For more information, refer to *IC_CLK Frequency Configuration*. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set.

3.13.7.10 HSCL_HCOUNT

Register	Offset	R/W	Description	Reset Value
HSCL_HCOUNT	I2C_BA+0×24	R/W	High Speed I2C Clock SCL High Count Register	0×0000_0006

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[15:0]	HSCL_HCOUNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock high period count for high speed. For more information, refer to *IC_CLK Frequency Configuration*. The SCL High time depends on the loading of the bus. For 100pF loading,the SCL High time is 60ns; for 400pF loading, the SCL High time is 120ns. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 6; hardware prevents values less than this being written, and if attempted results in 6 being set.

3.13.7.11 HSCL_LCOUNT

Register	Offset	R/W	Description	Reset Value
HSCL_LCOUNT	I2C_BA+0×28	R/W	High Speed I2C Clock SCL Low Count Register	0×0000_0010

Bits	Descriptions	
[31:16]	Reserved	Reserved.
[15:0]	HSCL_LCOUNT	This register must be set before any I2C bus transaction can take place to ensure proper I/O timing. This register sets the SCL clock low period count for high speed. For more information, refer to *IC_CLK Frequency Configuration*. The SCL low time depends on the loading of the bus. For 100pF loading, the SCL low time is 160ns; for 400pF loading, the SCL low time is 320ns. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 8; hardware prevents values less than this being written, and if attempted results in 8 being set. If the value is less than 8 then the count value gets changed to 8.

3.13.7.12 IRQ_STATUS

Register	Offset	R/W	Description	Reset Value
IRQ_STATUS	I2C_BA+0×2C	R	I2C Interrupt Status Register	0×0000_0000

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[13]	R_MASTER_HOLD	See *RAW_IRQ_STATUS* for a detailed description of this bit.
[12]	Reserved	Reserved
[11]	R_GENERAL_CALL	See *RAW_IRQ_STATUS* for a detailed description of this bit.
[10]	R_START_CHECK	
[9]	R_STOP_CHECK	
[8]	R_ACTIVITY	
[7]	R_RX_DONE	
[6]	R_TX_ABRT	
[5]	R_RD_REQ	
[4]	R_TX_EMPTY	
[3]	R_TX_OVER	
[2]	R_RX_FULL	
[1]	R_RX_OVER	
[0]	R_RX_UNDER	

3.13.7.13 IRQ_MASK

Register	Offset	R/W	Description	Reset Value
IRQ_MASK	I2C_BA+0×30	R/W	I2C Interrupt Mask Register	0×0000_08FF

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[13]	M_MASTER_HOLD	This bit masks the R_MASTER_HOLD interrupt bit in the *IRQ_STATUS* register.(Read-Only)
[12]	Reserved	Reserved
[11]	M_GENERAL_CALL	These bits mask their corresponding interrupt status bits in the *IRQ_STATUS* register.
[10]	M_START_CHECK	
[9]	M_STOP_CHECK	
[8]	M_ACTIVITY	
[7]	M_RX_DONE	
[6]	M_TX_ABRT	
[5]	M_RD_REQ	
[4]	M_TX_EMPTY	
[3]	M_TX_OVER	
[2]	M_RX_FULL	
[1]	M_RX_OVER	
[0]	M_RX_UNDER	

3.13.7.14 RAW_IRQ_STATUS

Register	Offset	R/W	Description	Reset Value
RAW_IRQ_STATUS	I2C_BA+0×34	R	I2C raw Interrupt Status Register	0×0000_0000

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[13]	MASTER_HOLD	Indicates whether a master is holding the bus and the Tx FIFO is empty.
[12]	Reserved	Reserved
[11]	GENERAL_CALL	Set only when a General Call address is received and it is acknowledged. It stays set until it is cleared either by disabling I2C or when the CPU reads bit 0 of the CLR_GENERAL_CALL register. I2C stores the received data in the Rx buffer.
[10]	START_CHECK	Indicates whether a START or RESTART condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode.
[9]	STOP_CHECK	Indicates whether a STOP condition has occurred on the I2C interface regardless of whether I2C is operating in slave or master mode. In Slave Mode: If I2C_CNTRL[7]=1'b1 (STOP_CHECK_IFSOLVED), the STOP_CHECK interrupt is generated only if the slave is addressed. Note: During a general call address, this slave does not issue a STOP_CHECK interrupt if STOP_CHECK_IFSOLVED=1'b1, even if the slave responds to the general call address by generating ACK. The STOP_CHECK interrupt is generated only when the transmitted address matches the slave address (SAR). If I2C_CNTRL[7]=1'b0 (STOP_CHECK_IFSOLVED), the STOP_CHECK interrupt is issued irrespective of whether it is being addressed. In Master Mode: The STOP_CHECK interrupt is issued irrespective of whether the master is active.
[8]	ACTIVITY	This bit captures I2C activity and stays set until it is cleared. There are four ways to clear it: Disabling the I2C Reading the *CLR_ACTIVITY* register Reading the *CLR_IRQ* register System reset Once this bit is set, it stays set unless one of the four methods is used to clear it. Even if the I2C module is idle, this bit remains set until cleared, indicating that there was activity on the bus.
[7]	RX_DONE	When the I2C is acting as a slave-transmitter, this bit is set to 1 if the master does not acknowledge a transmitted byte. This occurs on the last byte of the transmission, indicating that the transmission is done.
[6]	TX_ABRT	This bit indicates if I2C, as an I2C transmitter, is unable to complete the intended actions on the contents of the transmit FIFO. This situation can occur both as an I2C master or an I2C slave, and is referred to as a “transmit abort”.When this bit is set to 1, the TX_ASR register indicates the reason why the transmit abort takes places. Note: The I2C flushes/resets/empties only the TX_FIFO whenever there is a transmit abort caused by any of the events tracked by the TX_ASR register. The Tx FIFO remains in this flushed state until the register CLR_TX_ABRT is read.

		Once this read is performed, the Tx FIFO is then ready to accept more data bytes from the APB interface. RX FIFO is also flushed.
[5]	RD_REQ	This bit is set to 1 when I2C is acting as a slave and another I2C master is attempting to read data from I2C. The I2C holds the I2C bus in a wait state (SCL=0) until this interrupt is serviced, which means that the slave has been addressed by a remote master that is asking for data to be transferred. The processor must respond to this interrupt and then write the requested data to the DATA_CMD register. This bit is set to 0 just after the processor reads the CLR_RD_REQ register.
[4]	TX_EMPTY	The behavior of the TX_EMPTY interrupt status differs based on the TX_EMPTY_CTRL selection in the I2C_CNTRL register. When TX_EMPTY_CTRL = 0: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the FIFO_TX_TSD register. When TX_EMPTY_CTRL = 1: This bit is set to 1 when the transmit buffer is at or below the threshold value set in the FIFO_TX_TSD register and the transmission of the address/data from the internal shift register for the most recently popped command is completed. It is automatically cleared by hardware when the buffer level goes above the threshold. When I2C_EN[0] is set to 0, the TX FIFO is flushed and held in reset. There the TX FIFO looks like it has no data within it, so this bit is set to 1, provided there is activity in the master or slave state machines. When there is no longer any activity, then with EN_STATUS=0, this bit is set to 0.
[3]	TX_OVER	Set during transmit if the transmit buffer is filled to IC_TX_BUFFER_DEPTH and the processor attempts to issue another I2C command by writing to the DATA_CMD register. When the module is disabled, this bit keeps its level until the master or slave state machines go into idle, and when EN_STATUS goes to 0, this interrupt is cleared.
[2]	RX_FULL	Set when the receive buffer reaches or goes above the RX_TSD threshold in the FIFO_RX_TSD register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (I2C_EN[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once I2C_EN[0] is set to 0, regardless of the activity that continues.
[1]	RX_OVER	Set if the receive buffer is completely filled to IC_RBR_DEPTH and an additional byte is received from an external I2C device. The I2C acknowledges this, but any data bytes received after the FIFO is full are lost. If the module is disabled (I2C_EN[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when EN_STATUS goes to 0, this interrupt is cleared.
[0]	RX_UNDER	Set if the processor attempts to read the receive buffer when it is empty by reading from the DATA_CMD register. If the module is disabled(I2C_EN[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when EN_STATUS goes to 0, this interrupt is cleared.

3.13.7.15 FIFO_RX_TSD

Register	Offset	R/W	Description	Reset Value
FIFO_RX_TSD	I2C_BA+0×38	R/W	I2C Receive FIFO Threshold Register	0×0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	RX_TSD	Receive FIFO Threshold Level Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in *RAW_IRQ_STATUS* register). The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.

3.13.7.16 FIFO_TX_TSD

Register	Offset	R/W	Description	Reset Value
FIFO_TX_TSD	I2C_BA+0×3C	R/W	I2C Transmit FIFO Threshold Register	0×0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	TX_TSD	Transmit FIFO Threshold Level Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in *RAW_IRQ_STATUS* register). The valid range is 0-255, with the additional restriction that it may not be set to value larger than the depth of the buffer. If an attempt is made to do that, the actual value set will be the maximum depth of the buffer. A value of 0 sets the threshold for 0 entries, and a value of 255 sets the threshold for 255 entries.

3.13.7.17 CLR_IRQ

Register	Offset	R/W	Description	Reset Value
CLR_IRQ	I2C_BA+0×40	R	Clear Combined and Individual Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	RD_CLR_IRQ	Read this register to clear the combined interrupt, all individual interrupts, and the *TX_ASR* register. This bit does not clear hardware clear interrupts but software clear interrupts. Refer to Bit 9 of the TX_ASR register for an exception to clearing TX_ASR.

3.13.7.18 CLR_RX_UNDER

Register	Offset	R/W	Description	Reset Value
CLR_RX_UNDER	I2C_BA+0×44	R	Clear RX_UNDER Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_RX_UNDER	Read this register to clear the RX_UNDER interrupt (bit 0) of the *RAW_IRQ_STATUS* register.

3.13.7.19 CLR_RX_OVER

Register	Offset	R/W	Description	Reset Value
CLR_RX_OVER	I2C_BA+0×48	R	Clear RX_OVER Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_RX_OVER	Read this register to clear the RX_OVER interrupt (bit 1) of the *RAW_IRQ_STATUS* register.

3.13.7.20 CLR_TX_OVER

Register	Offset	R/W	Description	Reset Value
CLR_TX_OVER	I2C_BA+0×4C	R	Clear TX_OVER Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_TX_OVER	Read this register to clear the TX_OVER interrupt (bit 3) of the *RAW_IRQ_STATUS* register.

3.13.7.21 CLR_RD_REQ

Register	Offset	R/W	Description	Reset Value
CLR_RD_REQ	I2C_BA+0×50	R	Clear RD_REQ Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_RD_REQ	Read this register to clear the RD_REQ interrupt (bit 5) of the register.

3.13.7.22 CLR_TX_ABRT

Register	Offset	R/W	Description	Reset Value
CLR_TX_ABRT	I2C_BA+0×54	R	Clear TX_ABRT Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_TX_ABRT	Read this register to clear the TX_ABRT interrupt (bit 6) of the *RAW_IRQ_STATUS* register, and the *TX_ASR* register. This also releases the Tx FIFO from the flushed/reset state, allowing more writes to the Tx FIFO. Refer to Bit 9 of the *TX_ASR* register for an exception to clearing TX_ASR.

3.13.7.23 CLR_RX_DONE

Register	Offset	R/W	Description	Reset Value
CLR_RX_DONE	I2C_BA+0×58	R	Clear RX_DONE Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_RX_DONE	Read this register to clear the RX_DONE interrupt (bit 7) of the *RAW_IRQ_STATUS* register.

3.13.7.24 CLR_ACTIVITY

Register	Offset	R/W	Description	Reset Value
CLR_ACTIVITY	I2C_BA+0×5C	R	Clear ACTIVITY Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_ACTIVITY	Reading this register clears the ACTIVITY interrupt if the I2C is not active anymore. If the I2C module is still active on the bus, the ACTIVITY interrupt bit continues to be set. It is automatically cleared by hardware if the module is disabled and if there is no further activity on the bus. The value read from this register to get status of the ACTIVITY interrupt (bit 8) of the *RAW_IRQ_STATUS* register.

3.13.7.25 CLR_STOP_CHECK

Register	Offset	R/W	Description	Reset Value
CLR_STOP_CHECK	I2C_BA+0×60	R	Clear STOP_CHECK Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_STOP_CHECK	Read this register to clear the STOP_CHECK interrupt (bit 9) of the *RAW_IRQ_STATUS* register.

3.13.7.26 CLR_START_CHECK

Register	Offset	R/W	Description	Reset Value
CLR_START_CHECK	I2C_BA+0×64	R	Clear START_CHECK Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_START_CHECK	Read this register to clear the START_CHECK interrupt (bit 10) of the *RAW_IRQ_STATUS* register.

3.13.7.27 CLR_GENERAL_CALL

Register	Offset	R/W	Description	Reset Value
CLR_GENERAL_CALL	I2C_BA+0×68	R	Clear GENERAL_CALL Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_GENERAL_CALL	Read this register to clear the GENERAL_CALL interrupt (bit 11) of *RAW_IRQ_STATUS* register.

3.13.7.28 I2C_EN

Register	Offset	R/W	Description	Reset Value
I2C_EN	I2C_BA+0×6C	R/W	I2C Enable Register	0×0000_0000

Bits	Descriptions	
[31:2]	Reserved	Reserved.
[1]	ABORT	When set, the controller initiates the transfer abort. 0 = ABORT not initiated or ABORT done 1 = ABORT operation in progress The software can abort the I2C transfer in master mode by setting this bit. The software can set this bit only when EN is already set; otherwise, the controller ignores any write to ABORT bit. The software cannot clear the ABORT bit once set. In response to an ABORT, the controller issues a STOP and flushes the Tx FIFO after completing the current transfer, then sets the TX_ABORT interrupt after the abort operation. The ABORT bit is cleared automatically after the abort operation.
[0]	EN	Controls whether the I2C is enabled. 0 = Disables I2C (TX and RX FIFOs are held in an erased state) 1 = Enables I2C Software can disable I2C while it is active. However, it is important that care be taken to ensure that I2C is disabled properly.. When I2C is disabled, the following occurs: The TX FIFO and RX FIFO get flushed. Status bits in the IRQ_STATUS register are still active until I2C goes into IDLE state. If the module is transmitting, it stops as well as deletes the contents of the transmit buffer after the current transfer is complete. If the module is receiving, the I2C stops the current transfer at the end of the current byte and does not acknowledge the transfer.

3.13.7.29 I2C_STATUS

Register	Offset	R/W	Description	Reset Value
I2C_STATUS	I2C_BA+0×70	R	I2C Status Register	0×0000_0006

Bits	Descriptions	
[31:7]	Reserved	Reserved.
[6]	SLAVE_ACT	Slave FSM Activity Status. When the Slave Finite State Machine (FSM) is not in the IDLE state, this bit is set. 0 = Slave FSM is in IDLE state so the Slave part of I2C is not Active 1 = Slave FSM is not in IDLE state so the Slave part ofI2C is Active
[5]	MASTER_ACT	Master FSM Activity Status. When the Master Finite State Machine (FSM) is not in the IDLE state, this bit is set. 0 = Master FSM is in IDLE state so the Master part of I2C is not Active 1 = Master FSM is not in IDLE state so the Master part ofI2C is Active Note: I2C_STATUS[0]—that is, ACTIVITY bit—is the OR of SLAVE_ACT and MASTER_ACT bits.
[4]	RX_FIFO_CF	Receive FIFO Completely Full. When the receive FIFO is completely full, this bit is set. When the receive FIFO contains one or more empty location, this bit is cleared. 0 = Receive FIFO is not full 1 = Receive FIFO is full
[3]	RX_FIFO_NE	Receive FIFO Not Empty. This bit is set when the receive FIFO contains one or more entries; it is cleared when the receive FIFO is empty. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty
[2]	TX_FIFO_CE	Transmit FIFO Completely Empty. When the transmit FIFO is completely empty, this bit is set. When it contains one or more valid entries, this bit is cleared. This bit field does not request an interrupt. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty
[1]	TX_FIFO_NF	Transmit FIFO Not Full. Set when the transmit FIFO contains one or more empty locations, and is cleared when the FIFO is full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full
[0]	ACT_STATUS	I2C Activity Status.

3.13.7.30 TX_FIFO_LR

This register contains the number of valid data entries in the transmit FIFO buffer. It is cleared whenever:

- The I2C is disabled
- There is a transmit abort—that is, TX_ABRT bit is set in the RAW_IRQ_STATUS register
- The slave bulk transmit mode is aborted

The register increments whenever data is placed into the transmit FIFO and decrements when data is taken from the transmit FIFO.

Register	Offset	R/W	Description	Reset Value
TX_FIFO_LR	I2C_BA+0×74	R	I2C Transmit FIFO Level Register	0×0000_0000

Bits	Descriptions	
[31:4]	Reserved	Reserved.
[3:0]	TX_FIFO_LR	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

3.13.7.31 RX_FIFO_LR

This register contains the number of valid data entries in the receive FIFO buffer. It is cleared whenever:

- The I2C is disabled
- Whenever there is a transmit abort caused by any of the events tracked in TX_ASR

The register increments whenever data is placed into the receive FIFO and decrements when data is taken from the receive FIFO.

Register	Offset	R/W	Description	Reset Value
RX_FIFO_LR	I2C_BA+0×78	R	I2C Receive FIFO Level Register	0×0000_0000

Bits	Descriptions	
[31:4]	Reserved	Reserved.
[3:0]	RX_FIFO_LR	Receive FIFO Level. Contains the number of valid data entries in the receive FIFO.

3.13.7.32 SDA_HTLR

Writes to this register succeed only when I2C_EN[0]=0.

The values in this register are in units of i2c_clk period. The value programmed in SDA_TX_HTLR must be greater than the minimum hold time in each mode —one cycle in master mode, seven cycles in slave mode —for the value to be implemented.

The programmed SDA hold time during transmit (SDA_TX_HTLR) cannot exceed at any time the duration of the low part of scl. Therefore the programmed value cannot be larger than N_SCL_LOW-2, where N_SCL_LOW is the duration of the low part of the scl period measured in i2c_clk cycles.

Register	Offset	R/W	Description	Reset Value
SDA_HTLR	I2C_BA+0×7C	R/W	I2C SDA Hold Time Length Register	0×0001_0001

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[23:16]	SDA_RX_HTLR	Sets the required SDA hold time in units of i2c_clk period, when I2C acts as a receiver. They are used to extend the SDA transition (if any) whenever SCL is HIGH in the receiver in either master or slave mode.
[15:0]	SDA_TX_HTLR	Sets the required SDA hold time in units of i2c_clk period, when I2C acts as a transmitter. They are used to control the hold time of SDA during transmit in both slave and master mode (after SCL goes from HIGH to LOW).

3.13.7.33 TX_ASR

This register has 32 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the CLR_TX_ABRT register or the CLR_IRQ register is read. To clear Bit 9, the source of the RESTART_DIS_SEND_STA must be fixed first; RESTART must be enabled (I2C_CNTRL[5]=1), the IFEXECUTE bit must be cleared (TAR_ADDR[11]), or the CALL_START bit must be cleared (TAR_ADDR[10]).

Once the source of the RESTART_DIS_SEND_STA is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the RESTART_DIS_SEND_STA is not fixed before attempting to clear this bit, Bit 9 clears for one cycle and is then re-asserted.

Register	Offset	R/W	Description	Reset Value
TX_ASR	I2C_BA+0×80	R	I2C Transmit Abort Source Register	0×0000_0000

Bits	Descriptions	
[31]	TX_UPDATE_CNT	This field indicates the number of Tx FIFO data commands that are flushed due to TX_ABRT interrupt. It is cleared whenever I2C is disabled. Role of I2C: Master-Transmitter or Slave-Transmitter
[30:17]	Reserved	Reserved.
[16]	MASTER_TRANS_ABORT	This is a master-mode-only bit. Master has detected the transfer abort (I2C_EN[1]). Role of I2C: Master-Transmitter
[15]	SLAVE_TRANS_ABORT	1 = When the processor side responds to a slave mode request for data to be transmitted to a remote master and user writes a 1 in RNW (bit 8) of *DATA_CMD* register. Role of I2C:Slave-Transmitter
[14]	SLAVE_LOST	1 = Slave lost the bus while transmitting data to a remote master. *TX_ASR* [12] is set at the same time. Note: Even though the slave never “owns” the bus, something could go wrong on the bus.This is a fail safe check. For instance, during a data transmission at the low-to-high transition of SCL, if what is on the data bus is not what is supposed to be transmitted, then I2C no longer own the bus. Role of I2C:Slave-Transmitter
[13]	SLAVE_UTXF	1 = Slave has received a read command and some data exists in the TX FIFO so the slave issues a TX_ABRT interrupt to flush old data in TX FIFO. Role of I2C:Slave-Transmitter
[12]	MASTER_LOST	1 = Master has lost arbitration, or if *TX_ASR*[14] is also set, then the slave transmitter has lost arbitration. Role of I2C:Slave-Transmitter Master-Transmitter
[11]	MASTER_OPER_DIS	1 = User tries to initiate a Master operation with the Master mode disabled. Role of I2C:Master-Receiver

		Master-Transmitter
[10]	RESTART_DIS_RD_10B	1 = The restart is disabled (RESTART_EN bit (I2C_CNTRL[5]) = 0) and the master sends a read command in10-bit addressing mode. Role of I2C:Master-Receiver
[9]	RESTART_DIS_SEND_STA	To clear Bit 9, the source of the RESTART_DIS_SEND_STA must be fixed first; restart must be enabled (I2C_CNTRL[5]=1), the IFEXECUTE bit must be cleared (*TAR_ADDR*[11]), or the CALL_START bit must be cleared(*TAR_ADDR*[10]). Once the source of the RESTART_DIS_SEND_STA is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the RESTART_DIS_SEND_STA is not fixed before attempting to clear this bit, bit 9 clears for one cycle and then gets re-asserted. 1 = The restart is disabled (RESTART_EN bit (I2C_CNTRL[5]) = 0) and the user is trying to send a START Byte. Role of I2C: Master
[8]	RESTART_DIS_HS	1 = The restart is disabled (RESTART_EN bit (I2C_CNTRL[5]) = 0) and the user is trying to use the master to transfer data in High Speed mode. Role of I2C:Master-Transmitter Master-Receiver
[7]	START_ACK	1 = Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Role of I2C:Master
[6]	HS_ACK	1 = Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Role of I2C: Master
[5]	RD_GEN_CALL	1 = I2C in master mode sent a General Call but the user programmed the byte following the General Call to be a read from the bus (DATA_CMD[9] is set to 1). Role of I2C: Master-Transmitter
[4]	GEN_CALL_NACK	1 = I2C in master mode sent a General Call and no slave on the bus acknowledged the General Call. Role of I2C: Master-Transmitter
[3]	TX_DATA_NACK	1 = This is a master-mode only bit. Master has received an acknowledgment for the address, but when it sent data byte(s) following the address, it did not receive an acknowledge from the remote slave(s). Role of I2C: Master-Transmitter
[2]	MASTER_10ADDR2_NACK	1 = Master is in 10-bit address mode and the second address byte of the 10-bit address was not acknowledged by any slave. Role of I2C: Master-Transmitter Role of I2C: Master-Receiver
[1]	MASTER_10ADDR1_NACK	1 = Master is in 10-bit address mode and the first 10-bit address byte was not acknowledged by any slave. Role of I2C: Master-Transmitter Role of I2C: Master-Receiver
[0]	MASTER_7ADDR_NACK	1 = Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Role of I2C: Master-Transmitter Role of I2C: Master-Receiver

3.13.7.34 SLAVE_NACK

Register	Offset	R/W	Description	Reset Value
SLAVE_NACK	I2C_BA+0×84	R/W	Generate Slave Data NACK Register	0×0000_0000

Bits	Description		
[31:1]	Reserved	N/A	Reserved.
[0]	GEN_NACK	R/W	Generate NACK. This NACK generation only occurs when I2C is a slave receiver. If this register is set to a value of 1, it can only generate a NACK after a data byte is received; hence, the data transfer is aborted and the data received is not pushed to the receive buffer. When the register is set to a value of 0, it generates NACK/ACK, depending on normal criteria. 1 = generate NACK after data byte received 0 = generate NACK/ACK normally

3.13.7.35 DMA_CNTRL

Register	Offset	R/W	Description	Reset Value
DMA_CNTRL	I2C_BA+0×88	R/W	DMA Control Register	0×0000_0000

Bits	Descriptions	
[31:2]	Reserved	Reserved.
[1]	DMA_TX_EN	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
[0]	DMA_RX_EN	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel. 0 = Receive DMA disabled 1 = Receive DMA enabled

3.13.7.36 DMA_TX_DLR

Register	Offset	R/W	Description	Reset Value
DMA_TX_DLR	I2C_BA+0×8C	R/W	DMA Transmit Data Level Register	0×0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2:0]	DMA_TX_DLR	Transmit Data Level. This bit field controls the level at which a DMA request is made by the transmit logic. It is equal to the watermark level; that is, the dma_tx_req signal is generated when the number of valid data entries in the transmit FIFO is equal to or below this field value, and TDMACE = 1.

3.13.7.37 DMA_RX_DLR

Register	Offset	R/W	Description	Reset Value
DMA_RX_DLR	I2C_BA+0×90	R/W	DMA Receive Data Level Register	0×0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2:0]	DMA_RX_DLR	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMA_RX_DLR+1; that is, dma_rx_req is generated when the number of valid data entries in the receive FIFO is equal to or more than this field value + 1, and DMA_RX_EN = 1. For instance, when DMA_RX_DLR is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.

3.13.7.38 SDA_SR

Register	Offset	R/W	Description	Reset Value
SDA_SR	I2C_BA+0×94	R/W	I2C SDA Setup Register	0×0000_0064

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	SDA_SR	SDA Setup. It is recommended that if the required delay is 1000ns, then for an i2c_clk frequency of 10 MHz, SDA_SR should be programmed to a value of 11. Note: Writes to this register succeed only when I2C_EN [0] = 0.

3.13.7.39 ACK_GENERAL_CALL

Register	Offset	R/W	Description	Reset Value
ACK_GENERAL_CALL	I2C_BA+0×98	R/W	I2C ACK General Call Register	0×0000_0001

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	ACK_GENERAL_CALL	ACK General Call. When set to 1, I2C responds with a ACK (by asserting i2c_data_oe) when it receives a General Call. When set to 0, the I2C does not generate General Call interrupts. Note: This register is applicable only when the DW_apb_i2c is in the slave mode.

3.13.7.40 I2C_EN_STATUS

The register is used to report the I2C hardware status when I2C_EN[0] is set from 1 to 0; that is, when I2C is disabled.

- If I2C_EN[0] has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.
- If I2C_EN[0] has been set to 0, bits 2:1 is only be valid as soon as bit 0 is read as '0'.

Register	Offset	R/W	Description	Reset Value
I2C_EN_STATUS	I2C_BA+0×9C	R	I2C Enable Status Register	0×0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2]	SLAVE_RXDL	Slave Received Data Lost. This bit indicates if a Slave-Receiver operation has been aborted with at least one data byte received from an I2C transfer due to setting I2C_EN[0] from 1 to 0. When read as 1, I2C is deemed to have been actively engaged in an aborted I2C transfer (with matching address) and the data phase of the I2C transfer has been entered, even though a data byte has been responded with a NACK. **Note**: If the remote I2C master terminates the transfer with a STOP condition before the I2C has a chance to NACK a transfer, and I2C_EN[0] has been set to 0, then this bit is also set to 1. When read as 0, I2C is deemed to have been disabled without being actively involved in the data phase of a Slave-Receiver transfer. **Note**: The CPU can safely read this bit when EN_STATUS (bit 0) is read as 0.
[1]	SLAVE_DWB	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due to setting bit 0 of the I2C_EN register from 1 to 0. This bit is set when the CPU writes a 0 to bit 0 of I2C_EN while: • I2C is receiving the address byte of the Slave-Transmitter operation from a remote master; OR • Address and data bytes of the Slave-Receiver operation from a remote master When read as 1, I2C is deemed to have forced a NACK during any part of an I2C transfer, irrespective of whether the I2Caddress matches the slave address set in I2C (SLAVE_ADDR register) OR if the transfer is completed before bit 0 of I2C_EN is set to 0, but has not taken effect. **Note**: If the remote I2C master terminates the transfer with a STOP condition before the I2C has a chance to NACK a transfer, and bit 0 of I2C_EN has been set to 0, then this bit will also be set to 1. When read as 0, I2C is deemed to have been disabled when there is master activity, or when the I2C bus is idle. **Note**: The CPU can safely read this bit when *EN_STATUS* (bit 0) is read as 0.
[0]	EN_STATUS	I2c_en Status. This bit always reflects the value driven on the output port I2c_en. • When read as 1, I2C is deemed to be in an enabled state. • When read as 0, I2C is deemed completely inactive. **Note:** The CPU can safely read this bit anytime. When this bit is read as 0, the

		CPU can safely read SLAVE_RXDL (bit 2) and SLAVE_DWB (bit 1).

3.13.7.41 SF_SSLR

Register	Offset	R/W	Description	Reset Value
SF_SSLR	I2C_BA+0×A0	R/W	I2C SS and FS Spike Suppression Limit Register	0×0000_0005

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	SF_SSLR	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in i2c_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic; for more information, refer to *Spike Suppression*. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set.

3.13.7.42 HS_SSLR

Register	Offset	R/W	Description	Reset Value
HS_SSLR	I2C_BA+0×A4	R/W	I2C HS Spike Suppression Limit Register	0×0000_0001

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	HS_SSLR	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in i2c_clk cycles, of the longest spike in the SCL or SDA lines that are filtered out by the spike suppression logic; for more information, refer to *Spike Suppression*. This register can be written only when the I2C interface is disabled, which corresponds to I2C_EN[0] being set to 0. Writes at other times have no effect. The minimum valid value is 1; hardware prevents values less than this being written, and if attempted, results in 1 being set.

3.13.7.43 CLR_RESTART_CHECK

Register	Offset	R/W	Description	Reset Value
CLR_RESTART_CHECK	I2C_BA+0×A8	R	Clear RESTART_CHECK Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	RESTART	Read this register to clear the RESTART_CHECK interrupt (bit 12) of the *RAW_IRQ_STATUS* register.

3.13.7.44 SCL_SLT

Register	Offset	R/W	Description	Reset Value
SCL_SLT	I2C_BA+0×AC	R/W	I2C SCL Stuck at Low Timeout	0×FFFF_FFFF

Bits	Descriptions	
[31:0]	SCL_SLT	I2C generates the interrupt to indicate SCL stuck at low if it detects the SCL stuck at low for the SCL_SLT in units of i2c_clk period.

3.13.7.45 SCL_SLDIR

Register	Offset	R/W	Description	Reset Value
SCL_SLDIR	I2C_BA+0×B4	R	Clear SCL Stuck at Low Detect Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved
[0]	SDA_SALT	Read this register to clear the SCL_STUCK_DET interrupt (bit 14) of the *RAW_IRQ_STATUS* register.

3.14 Analog-to-Digital Converter (ADC)

3.14.1 Overview

ADC contains one 12-bit successive approximation analog-to-digital converters (SAR A/D converter) with seven input channels. The A/D converters can be started by software, external pin (STADC: P0.5) or PWM trigger.

3.14.2 Features

- Two selectable analog input voltage range
  - 0~ AVDD analog input voltag.
    It has large measuring range and low precision.
  - 0~VBG analog input voltag.
    It has small measuring range and high precision.Generated by the internal BandGap output.
- 12-bit resolution and 9-bit accuracy is guaranteed
- Up to seven single-end analog input channels, one band-gap input channel, one band-gap/2 input channel, two temperature input channels and one voltage/4 input channel.
- Maximum ADC clock frequency is 32 MHz, and 4 ADC clocks per sample
- Three operating modes
  - Single mode: A/D conversion is performed one time on a specified channel
  - Shunt mode: Two of three ADC channels from 0 to 2 will automatically convert analog data in the sequence of channel [0,1] or channel[1,2] or channel[0,2] defined by MODESEL (ADC_SEQCTL[3:2])
  - PWM sequence mode: PWM continuously triggers a certain ADC channel, which is selected by SEQ_CH_SEL[2:0]
- An A/D conversion can be started by:
  - Software write 1 to SWTRG bit
  - External pin STADC(P0.5)
  - PWM trigger with optional start delay period
- Each Conversion result is held in data register with valid and overrun indicators
- Conversion results can be compared with specified value and user can select whether to generate an interrupt when conversion result matches the compare register setting
- Support DMA transfer

- Support FIFO mode, FIFO depth is 16
- Support the function of collecting bias voltage
- Support left shift (4bit) function

3.14.3 Block Diagram

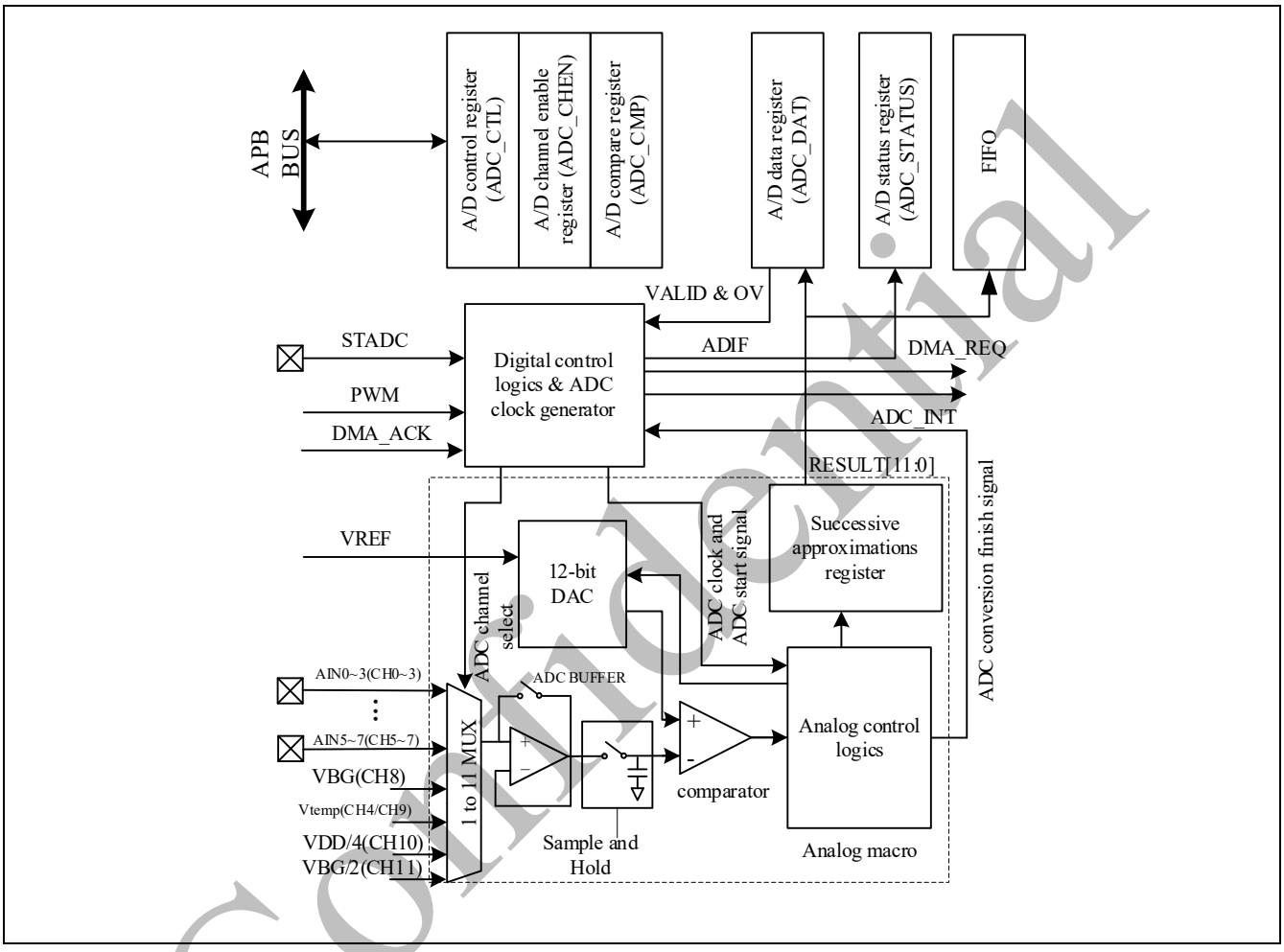


Figure 3-81 ADC Block Diagram

3.14.4 Basic Configuration

The ADC pin functions are configured in SYS_Px_MFP(x=0,1,2) register. It is recommended to disable the digital input path of the analog input pins to avoid the leakage current. User can disable the digital input path by configuring Px_DINOFF(x=0,1,2) register.

The ADC peripheral clock can be enabled in Adccken(APB1_CLK_CTRL0[9]).

3.14.5 Functional Description

The A/D converter operates by successive approximation with 12-bit resolution. When changing the analog input channel is enabled, in order to prevent incorrect operation, software

must clear SWTRG bit to 0 in the ADC_CTL register. The A/D converter discards the current conversion immediately and enters idle state while SWTRG bit is cleared. The ADC has a total of 12 channels. ADC channels 0-3/5-7 are used to test the external input voltage. Channel 8 tests the internal reference voltage VBG-1.2V. Channels 4/9 are the temperature sensor channels. The linear temperature sensor can measure the temperature on the chip, and the measurement range is -40~85℃.

Channel 10 tests the VDD/4 voltage. VDD is the chip supply voltage.

3.14.5.1 ADC Peripheral Clock Generator

The ADC engine is always from PCLK. The ADC clock peripheral frequency is divided by an 3-bit prescaler with the following formula:

ADC peripheral clock frequency = (ADC peripheral clock source frequency) / (ADC_DIV+1);

where the 3-bit ADC_DIV is located in register ADC_CTL2[10:8].

Temperature Sensor

The ADC module contains an internal temperature sensor that can be used to measure the CPU and ambient temperature (TA). The temperature sensor is internally connected to the ADC_CH9 input channel, which converts the voltage output by the sensor into a digital value. The temperature range supported by the internal temperature sensor is: -40~125 °C.

Battery detection

The ADC module includes a power detection channel, which is internally connected to the ADC_CH10 input channel, and uses a constant voltage Bandgap as a reference voltage to collect the adc code value of the 1/4 voltage dividing point within the power supply voltage VDD. Calculate the current supply voltage through the formula.

Calculation formula: VDD = ((Code * Vbg)/ 4096) * 4

Code: ADC sampling value of vdd 1/4 voltage dividing point

Vbg: Internal constant voltage, the reference value is around 1.2V, need to be calibrated

3.14.5.2 ADC Operation

A/D conversion is performed only once on the specified single channel. The operation is as follows:

1 A/D conversion will be started when the SWTRG bit of ADC_CTL is set to 1 by software or external trigger input.

2 When A/D conversion is finished, the result is stored in the A/D data register or FIFO.

3 The ADIF bit of ADC_STATUS register will be set to 1. If the ADCIEN bit of ADC_CTL

register is set to 1, the ADC interrupt will be asserted.

4 The SWTRG bit remains 1 during A/D conversion. When A/D conversion ends, the SWTRG bit is automatically cleared to 0 and the A/D converter enters idle state.

5 Figure 3-82 shows an example timing diagram for Single mode.

Note: If software enables more than one channel, the channel with the smallest number will be selected and the other enabled channels will be ignored.

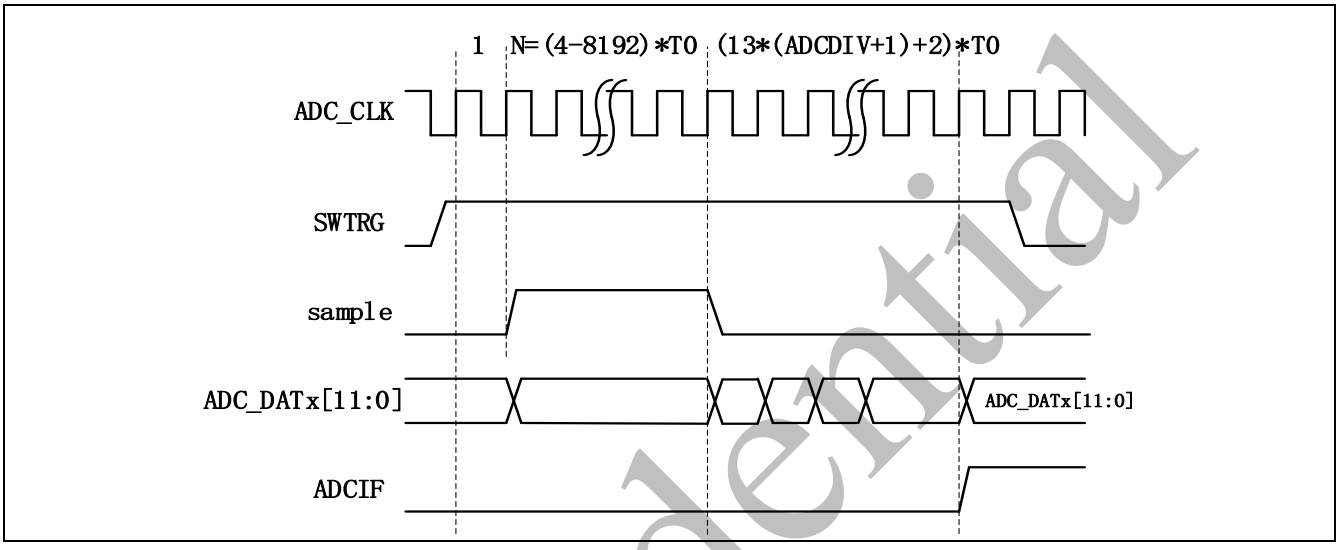


Figure 3-82 Single Mode Conversion Timing Diagram

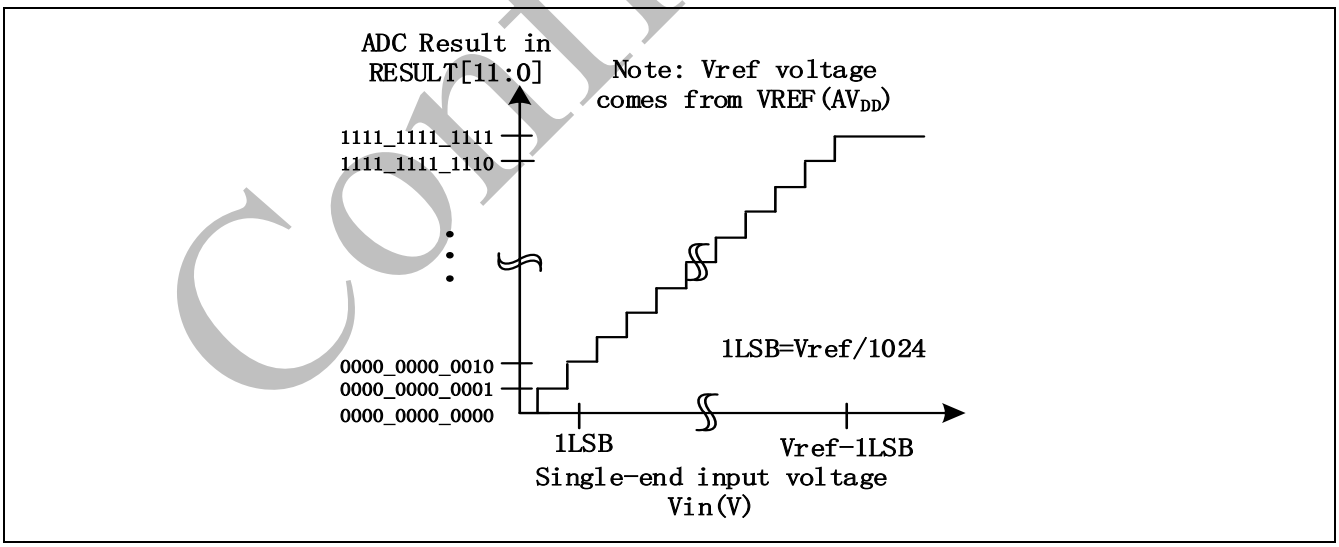


Figure 3-83 Conversion Result Mapping Diagram of ADC Single-end Input

3.14.5.3 External Trigger Input Sampling and A/D Conversion Time

A/D conversion can be triggered by external pin request. When the HWTRGEN (ADC_CTL[8]) bit is set to 1 to enable ADC external trigger function, setting the HWTRGSEL (ADC_CTL[5:4]) bits to 00b is to select external trigger input from the STADC

pin. Software can set HWTRGCOND to select trigger condition between falling or rising edge. An 3-bit internal sampling counter is used to deglitch. If edge trigger condition is selected, the high and low state must be kept at least 4 PCLK. Pulse that is shorter than this specification will be ignored.

3.14.5.4 PWM Trigger

A/D conversion can also be triggered by PWM request. When the HWTRGEN is set to high to enable ADC external hardware trigger function, setting the HWTRGSEL (ADC_CTL[5:4]) bits to 11b is to select external hardware trigger input source from PWM trigger. When PWM trigger is enabled, setting DELAY (ADC_TRGDLY[7:0]) bits can insert a delay time between PWM trigger condition and ADC start conversion.

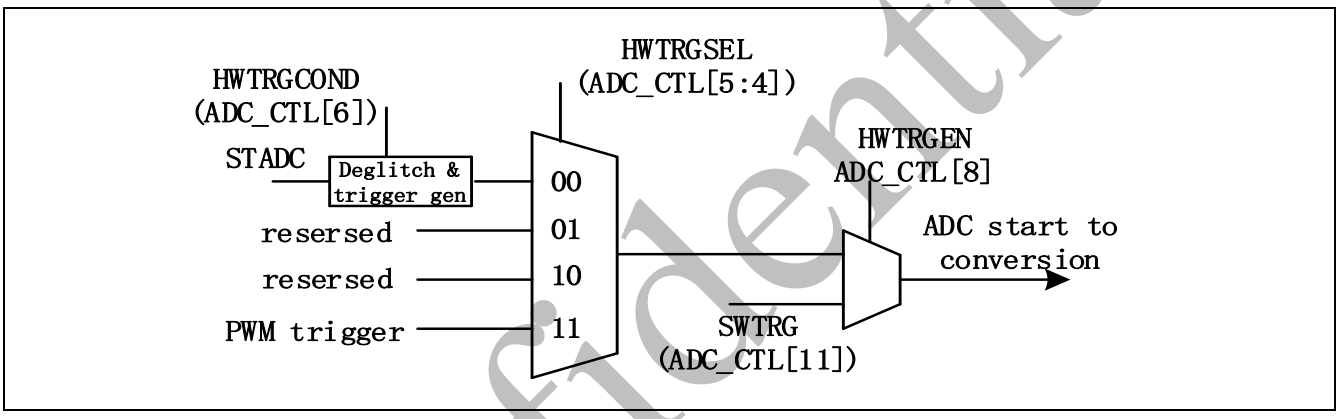


Figure 3-84 ADC Start Conversion Conditions

3.14.5.5 Conversion Result Monitor by Compare Mode Function

The ADC controller provides two compare registers, ADC_CMP0 and ADC_CMP1, to monitor maximum two specified channels. Software can select which channel to be monitored by setting CMPCH (ADC_CMPx[6:3]). CMPCOND bit is used to determine the compare condition. If CMPCOND bit is cleared to 0, the internal match counter will increase one when the conversion result is less than the value specified in CMPDAT (ADC_CMPx[27:16]) ; if CMPCOND bit is set to 1, the internal match counter will increase one when the conversion result is greater than or equal to the value specified in CMPDAT[11:0]. When the conversion of the channel specified by CMPCH is completed, the comparing action will be triggered one time automatically. When the compare result meets the setting, compare match counter will increase 1, otherwise, the compare match counter will be clear to 0. When the match counter reaches the setting of (CMPMCNT+1) then ADCMPIF bit will be set to 1, if ADCMPIE bit is set then an ADC_INT interrupt request is generated. Software can use it to monitor the

external analog input pin voltage transition in scan mode without imposing a load on software. Figure 3-85 show detailed logic diagram.

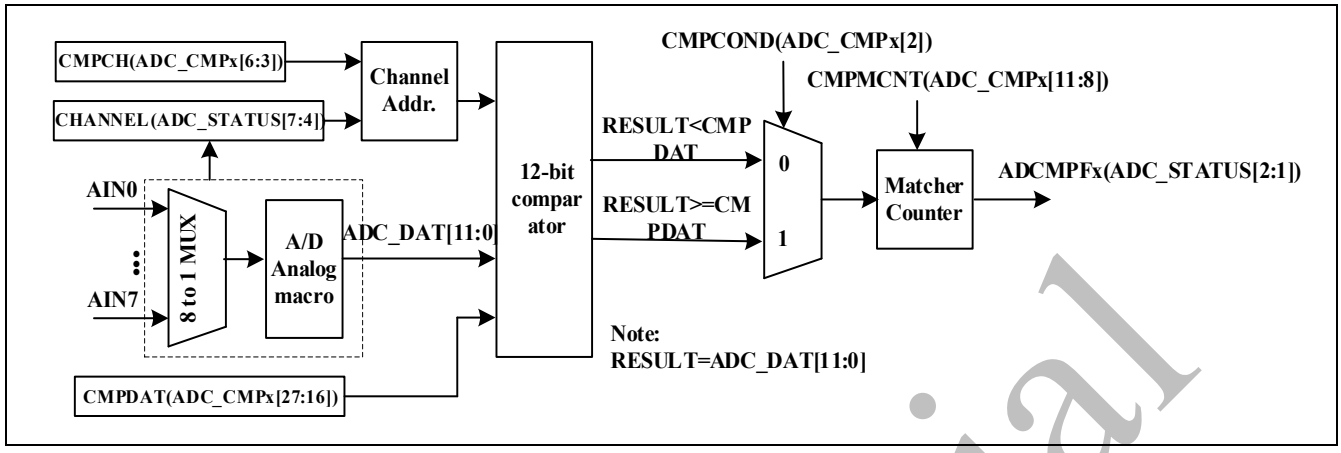


Figure 3-85 A/D Conversion Result Monitor Logics Diagram

3.14.5.6 Interrupt Mode

There are three interrupt sources of ADC interrupt. When an ADC operation mode finishes its conversion, the A/D conversion end flag, ADIF, will be set to 1. The ADCMPIF0 and ADCMPIF1 are the compare flags of compare function. When the conversion result meets the settings of ADC_CMP0/1, the corresponding flag will be set to 1. When one of the flags, ADIF, ADCMPIF0 and ADCMPIF1, is set to 1 and the corresponding interrupt enable bit, ADCIEN of ADC_CTL and ADCMPIE of ADC_CMP0/1, is set to 1, the ADC interrupt will be asserted. Software can clear these flags to revoke the interrupt request.

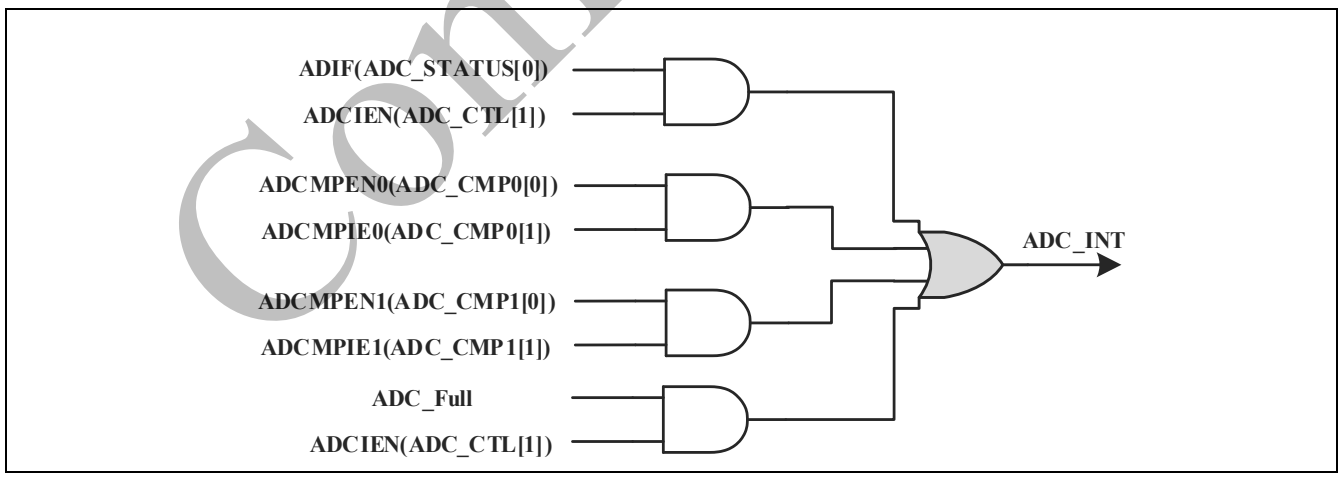


Figure 3-86 A/D Controller Interrupt

3.14.5.7 Polling Mode

There are three flag sources of ADC conversion. When an ADC operation mode finishes its conversion, the A/D conversion end flag, ADCF, will be set to 1. The ADCMPF0 and ADCMPF1 are the compare flags of compare function. When the conversion result meets the

settings of ADC_CMP0/1, the corresponding flag will be set to 1.

3.14.5.8 Shunt Mode

Support sequential mode for 2 channels to reduce half interrupt frequency. When the SEQEN (ADC_SEQCTL[0]) is set to high to enable ADC function. When ADC sequential mode is enabled, two of three ADC channels from 0 to 2 will automatically convert analog data in the sequence of channel [0, 1] or channel[1,2] or channel[0,2] defined by MODESEL (ADC_SEQCTL[3:2]).

3.14.5.9 PWM Sequential Mode

PWM continuously triggers a certain ADC channel, which is selected by SEQ_CH_SEL[2:0]. TRGxCTL[1:0] is used to select PWM trigger type, and TRGxCTL[3:2] is used to select PWM channel(PWM channel0/2/4/6).When the SEQEN (ADC_SEQCTL[0]) is set to high to enable ADC function. The software should follow the steps below to configure ADC and PWM。

1) HWTRGSEL=11
2) HWTRGEN=1
3) MODESEL=11
4) SEQ_ONE_CH_EN=1
5) TRGxCTL[3:2] choose PWM trigger channel
6) TRGxCTL[1:0] chooses PWM trigger mode
7) SEQEN=1
8) Start PWM

3.14.5.10 DMA Operation

When fifo_en=1 and DMA_EN=1, and the data in FIFO reaches half full (8 data), dma_req will issue a request, and the peripheral DMA will move 8 data from the ADC's FIFO. When the data in the FIFO is not half full, dma_req cannot be triggered. If the data in the FIFO does not reach 8 during the last data conversion, the CPU needs to manually read the data in the FIFO.

3.14.5.11 Left Shift

When the ADC is configured with the left shift function, the ADC converted data will be shifted left by 4 bits. If FIFO mode(ADC_FIFO_CTL[0]) and left_shift_en

(ADC_LS_CTL[0]) are set, the data read from the FIFO will be shifted left by 4 bits. Left_shifted data can be acquired in ADC_LS_CTL[31:16].

Eg: There are 4 data in the FIFO. If you want to get the left shift value of each data, you need to read the FIFO 4 times. Each time you read the FIFO, read the ADC_LS_CTL to get the corresponding shift value.

3.14.5.12 Collect bias voltage

When sub_bias_en=1(ADC_SUB_CTL[0]), set the bias(ADC_SUB_CTL[31:16]) voltage. Read data_left_bias(ADC_DATA_SUB[16:0]) to get its corresponding bias voltage code.

3.14.6 ADC Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
ADC Base Address: ADC_BA				
ADC_DAT	ADC_BA+0x00	R	A/D Data Register	0x0000_0000
ADC_CTL	ADC_BA+0x20	R/W	A/D Control Register	0x0000_0000
ADC_CHEN	ADC_BA+0x24	R/W	A/D Channel Enable Register	0x0000_0000
ADC_CMP0	ADC_BA+0x28	R/W	A/D Compare Register 0	0x0000_0000
ADC_CMP1	ADC_BA+0x2C	R/W	A/D Compare Register 1	0x0000_0000
ADC_STATUS	ADC_BA+0x30	R/W	A/D Status Register	0x0000_0000
ADC_TRGDLY	ADC_BA+0x44	R/W	A/D Trigger Delay Control Register	0x0000_0000
ADC_EXTSMPT	ADC_BA+0x48	R/W	A/D Sampling Time Counter Register	0x0000_0200
ADC_SEQCTL	ADC_BA+0x4C	R/W	A/D PWM Sequential Mode Control Register	0x0000_0000
ADC_SEQDAT1	ADC_BA+0x50	R	A/D PWM Sequential Mode First Result Register	0x0000_0000
ADC_SEQDAT2	ADC_BA+0x54	R	A/D PWM Sequential Mode Second Result Register	0x0000_0000
ADC_CTL2	ADC_BA+0x58	R/W	A/D Control Register 2	0x0101_0000
ADC_BV_CTL	ADC_BA+0x6C	R/W	A/D Bias Voltage Control Register	0x0000_0100

3.14.7 ADC Register Description

3.14.7.1 ADC Data Register (ADC_DAT)

Register	Offset	R/W	Description	Reset Value
ADC_DAT	ADC_BA+0x00	R	A/D Data Register	0x0000_0000

Bits	Description	
[31:18]	Reserved	Reserved.
[17]	VALID	Valid Flag, Just active when fifo_en=0; This bit is set to 1 when ADC conversion is completed and cleared by hardware after the ADC_DAT register is read. 0 = Data in RESULT[11:0] bits not valid. 1 = Data in RESULT[11:0] bits valid.
[16]	OV	Over Run Flag, Just active when fifo_en=0 If converted data in RESULT[11:0] has not been read before the new conversion result is loaded to this register, OV is set to 1. It is cleared by hardware after the ADC_DAT register is read. 0 = Data in RESULT[11:0] is recent conversion result. 1 = Data in RESULT[11:0] overwrote.
[15:0]	RESULT	A/D Conversion Result This field contains conversion result of ADC. Note： left_shift_en =0, sub_bias_en =0, fifo_en =0; result is just a 12-bit adc normal data left_shift_en =0, sub_bias_en =0, fifo_en =1; result is just a 12-bit adc fifo data left_shift_en =0, sub_bias_en =1, fifo_en =0; result is just a 12-bit adc normal data; bit13-bit15 is the sign bit left_shift_en =0, sub_bias_en =1, fifo_en =1; result is just a 12-bit adc fifo data; bit13-bit15 is the sign bit left_shift_en =1, sub_bias_en =0, fifo_en =0; result is just a 16-bit adc normal data left_shift_en =1, sub_bias_en =0, fifo_en =1; result is just a 16-bit adc fifo data left_shift_en =1, sub_bias_en =1, fifo_en =0; result is just a 15-bit adc normal data; bit15 is the sign bit left_shift_en =1, sub_bias_en =1, fifo_en =1; result is just a 15-bit adc normal data; bit15 is the sign bit

3.14.7.2 ADC Control Register (ADC_CTL)

Register	Offset	R/W	Description	Reset Value
ADC_CTL	ADC_BA+0x20	R/W	A/D Control Register	0x0800_0000

Bits	Description	
[31:28]	Reserved	Reserved
[27:16]	BIAS	12bit ADC data left shift to be 16bit data,this need left_shift_en set this function need use biasing curing of hardware default:0x800
[15:14]	Reserved	Reserved
[13]	SUB_BIAS_EN	0:left shift function disable 1:left shift function enable
[12]	LEFT_SHIFT_EN	Sub_bias_en: 0 0:MSB add 4bit0 ;{4'd0,fifo_pop_data[11:0]} 1:LSB add 4bit; {fifo_pop_data[11:0],4'd0 } Sub_bias_en: 1 0:MSB add 4bit0 ;{4'{sign_bit},fifo_pop_data[11:0]} 1:LSB add 4bit; {sign_bit,fifo_pop_data[11:0],3'd0 }
[11]	SWTRG	Software Trigger A/D Conversion Start SWTRG bit can be set to 1 from two sources: software and external pin STADC. SWTRG will be cleared to 0 by hardware automatically after conversion complete. 0 = Conversion stopped and A/D converter entered idle state. 1 = Conversion start.
[10]	FIFO_FULL_OR_HALF	Control DMA request in FIFO full state or half full state; 0:half full state 1:full state
[9]	FIFO_EN	ADC FIFO enable or disable 0:disable ADC FIFO 1:enable ADC FIFO
[8]	HWTRGEN	Hardware External Trigger Enable Bit Enable or disable triggering of A/D conversion by external STADC pin. If external trigger is enabled, the SWTRG bit can be set to 1 by the selected hardware trigger source. 0= External trigger Disabled. 1= External trigger Enabled.
[7]	Reserved	Reserved.
[6]	HWTRGCOND	Hardware External Trigger Condition This bit decides whether the external pin STADC trigger event is falling or raising edge. The signal must be kept at stable state at least 4 PCLKs at high and low state for edge trigger. 0 = Falling edge. 1 = Raising edge.
[5:4]	HWTRGSEL	Hardware Trigger Source Select Bit 00 = A/D conversion is started by external STADC pin. 11 = A/D conversion is started by PWM trigger.

		Others = Reserved. Note: Software should disable TRGEN and SWTRG before change TRGS.
[3:2]	Reserved	Reserved.
[1]	ADCIEN	A/D Interrupt Enable Bit A/D conversion end interrupt request is generated if ADCIEN bit is set to 1. 0 = A/D interrupt function Disabled. 1 = A/D interrupt function Enabled.
[0]	ADCEN	A/D Converter Enable Bit 0 = A/D Converter Disabled. 1 = A/D Converter Enabled. Note: Before starting A/D conversion function, this bit should be set to 1. Clear it to 0 to disable A/D converter analog circuit to save power consumption.

3.14.7.3 ADC Channel Enable Register (ADC_CHEN)

Register	Offset	R/W	Description	Reset Value
ADC_CHEN	ADC_BA+0x24	R/W	A/D Channel Enable Register	0x0000_0000

Bits	Description	
[31:12]	Reserved	Reserved.
[11]	CHEN11	VBG_0P6 check Enable Bit 0 = VBG_0P6 check Disabled. 1 = VBG_0P6 check Enabled.
[10]	CHEN10	1/4VDD check Enable Bit 0 = 1/4VDD check Disabled. 1 = 1/4VDD check Enabled.
[9]	CHEN9	VBG_VT check Enable Bit 0 = VBG_VT check Disabled. 1 = VBG_VT check Enabled.
[8]	CHEN8	VBG_1P2 check Enable Bit 0 = VBG_1P2 check Disabled. 1 = VBG_1P2 check Enabled.
[7]	CHEN7	Analog Input Channel 7 Enable Bit 0 = Channel 7 Disabled. 1 = Channel 7 Enabled.
[6]	CHEN6	Analog Input Channel 6 Enable Bit 0 = Channel 6 Disabled. 1 = Channel 6 Enabled.
[5]	CHEN5	Analog Input Channel 5 Enable Bit 0 = Channel 5 Disabled. 1 = Channel 5 Enabled.
[4]	CHEN4	Temp2 Enable Bit 0 = Temp2 Disabled. 1 = Temp2 Enabled.
[3]	CHEN3	Analog Input Channel 3 Enable Bit 0 = Channel 3 Disabled. 1 = Channel 3 Enabled.
[2]	CHEN2	Analog Input Channel 2 Enable Bit 0 = Channel 2 Disabled. 1 = Channel 2 Enabled.
[1]	CHEN1	Analog Input Channel 1 Enable Bit 0 = Channel 1 Disabled. 1 = Channel 1 Enabled.
[0]	CHEN0	Analog Input Channel 0 Enable Bit 0 = Channel 0 Disabled. 1 = Channel 0 Enabled. Note: If software enables more than one channel, the channel with the smallest number will be selected and the other enabled channels will be ignored.

3.14.7.4 A/D Compare Register 0/1 (ADC_CMP0/1)

Register	Offset	R/W	Description	Reset Value
ADC_CMP0	ADC_BA+0x28	R/W	A/D Compare Register 0	0x0000_0000
ADC_CMP1	ADC_BA+0x2C	R/W	A/D Compare Register 1	0x0000_0000

Bits	Description	
[31:28]	Reserved	Reserved.
[27:16]	CMPDAT	Comparison Data The 12-bit data is used to compare with conversion result of specified channel.
[15:12]	Reserved	Reserved.
[11:8]	CMPMCNT	Compare Match Count When the specified A/D channel analog conversion result matches the compare condition defined by CMPCOND[2], the internal match counter will increase 1. When the internal counter reaches the value to (CMPMCNT+1), the ADCMPFx bit will be set.
[7]	Reserved	Reserved.
[6:3]	CMPCH	Compare Channel Selection Set this field to select which channel's result to be compared. Note: Valid setting of this field is channel 0~7.
[2]	CMPCOND	Compare Condition 0 = Set the compare condition as that when a 12-bit A/D conversion result is less than the 12-bit CMPDAT (ADC_CMPx[25:16]), the internal match counter will increase one. 1 = Set the compare condition as that when a 12-bit A/D conversion result is greater or equal to the 12-bit CMPDAT (ADC_CMPx[25:16]), the internal match counter will increase one. Note: When the internal counter reaches the value to (CMPMCNT+1), the ADCMPFx bit will be set.
[1]	ADCMPIE	A/D Compare Interrupt Enable Bit If the compare function is enabled and the compare condition matches the setting of CMPCOND and CMPMCNT, ADCMPIE bit will be asserted, in the meanwhile, if ADCMPIE is set to 1, a compare interrupt request is generated. 0 = Compare function interrupt Disabled. 1 = Compare function interrupt Enabled.
[0]	ADCMPEN	A/D Compare Enable Bit Set 1 to this bit to enable comparing CMPDAT (ADC_CMPx[25:16]) with specified channel conversion results when converted data is loaded into the ADC_DAT register. 0 = Compare function Disabled. 1 = Compare function Enabled.

3.14.7.5 A/D Status Register (ADC_STATUS)

Register	Offset	R/W	Description	Reset Value
ADC_STATUS	ADC_BA+0x30	R/W	A/D Status Register	0x0000_0000

Bits	Description	
[31]	Reserved	Reserved
[30]	FIFO_HALF_F	FIFO half full falg Note:This bit can be cleared to 0 by software writing 1.
[29]	FIFO_OV_F	FIFO overflow flag. Note:This bit can be cleared to 0 by software writing 1.
[28]	FIFO_EMPTY_F	FIFO empty flag； FIFO empty flag,this means FIFO is empty ,then can push or can not pop FIFO Note:This bit can be cleared to 0 by software writing 1.
[27]	FIFO_FULL_F	FIFO full flag. Note:This bit can be cleared to 0 by software writing 1.
[26]	ADCMPF1	A/D Compare Flag 1 When the selected channel A/D conversion result meets the setting condition in ADC_CMP1, this bit is set to 1. 0 = Conversion result in ADC_DAT does not meet the ADC_CMP1 setting. 1 = Conversion result in ADC_DAT meets the ADC_CMP1 setting. Note: This bit can be cleared to 0 by software writing 1.
[25]	ADCMPF0	A/D Compare Flag 0 When the selected channel A/D conversion result meets the setting condition in ADC_CMP0, this bit is set to 1. 0 = Conversion result in ADC_DAT does not meet the ADC_CMP0 setting. 1 = Conversion result in ADC_DAT meets the ADC_CMP0 setting. Note: This bit can be cleared to 0 by software writing 1.
[24]	ADCF	A/D Conversion End Flag A status flag that indicates the end of A/D conversion. ADIF is set to 1 When A/D conversion ends. Note: This bit can be cleared to 0 by software writing 1.
[23]	Reserved	Reserved.
[22]	FIFO_HALF_IF	FIFO half full interrupt. Note:This bit can be cleared to 0 by software writing 1.
[21]	FIFO_OV_IF	FIFO overflow interrupt. Note:This bit can be cleared to 0 by software writing 1.
[20]	FIFO_EMPTY_IF	FIFO empty interrupt. Note:This bit can be cleared to 0 by software writing 1.
[19]	FIFO_FULL_IF	FIFO full occur interrupt,then software deal the data;fifo_full_int is set to 1 when fifo is full. Note:This bit can be cleared to 0 by software writing 1.
[18]	ADCF_ONE_CH	PWM sequence in ADC one channel ,when adc convert end,then adcf_one_ch is 1；
[17]	ADCF_OC_CLR	adcf_one_ch flag clear select 0:software clear 1:hardware clear

[16]	OV	Overrun Flag (Read Only) It is a mirror to OV bit in ADC_DAT register.
[15]	ADCMPIF1	ADCMPIF1 interrupt MSK 0:msk enable; 1: msk disable; Default:0
[14]	ADCMPIF0	ADCMPIF0 interrupt MSK 0:msk enable; 1: msk disable; Default:0
[13]	ADIF_MSK	Adif interrupt MSK 0:msk enable; 1: msk disable; Default:0
[12]	FIFO_HALF_IF_MSK	fifo_half_if interrupt MSK; 0:msk enable; 1: msk disable; Default:0
[11]	FIFO_OV_IF_MSK	fifo_ov_if interrupt MSK; 0:msk enable; 1: msk disable; Default:0
[10]	FIFO_EMPTY_IF_MSK	fifo_emptyl_if interrupt MSK; 0:msk enable; 1: msk disable; Default:0
[9]	FIFO_FULL_IF_MSK	fifo_full_if interrupt MSK; 0:msk enable; 1: msk disable; Default:0
[8]	VALID	Data Valid Flag (Read Only) It is a mirror of VALID bit in ADC_DAT register.
[7:4]	CHANNEL	Current Conversion Channel (Read Only) This filed reflects the current conversion channel when BUSY=1. When BUSY=0, it shows the number of the next converted channel.
[3]	BUSY	BUSY/IDLE (Read Only) This bit is mirror of as SWTRG bit in ADC_CTL 0 = A/D converter is in idle state. 1 = A/D converter is busy at conversion.
[2]	ADCMPIF1	A/D Compare Interrupt Flag 1 When the selected channel A/D conversion result meets the setting condition in ADC_CMP1, this bit is set to 1. 0 = Conversion result in ADC_DAT does not meet the ADC_CMP1 setting. 1 = Conversion result in ADC_DAT meets the ADC_CMP1 setting. Note: This bit can be cleared to 0 by software writing 1.
[1]	ADCMPIF0	A/D Compare Interrupt Flag 0 When the selected channel A/D conversion result meets the setting condition in ADC_CMP0, this bit is set to 1.

		0 = Conversion result in ADC_DAT does not meet the ADC_CMP0 setting. 1 = Conversion result in ADC_DAT meets the ADC_CMP0 setting. Note: This bit can be cleared to 0 by software writing 1.
[0]	ADIF	A/D Conversion End Interrupt Flag A status flag that indicates the end of A/D conversion. ADIF is set to 1 When A/D conversion ends. Note: This bit can be cleared to 0 by software writing 1.

3.14.7.6 A/D Trigger Delay Controller Register (ADC_TRGDLY)

Register	Offset	R/W	Description	Reset Value
ADC_TRGDLY	ADC_BA+0x44	R/W	A/D Trigger Delay Control Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	DELAY	PWM Trigger Delay Timer Set this field will delay ADC start conversion time after PWM trigger. PWM trigger delay time is (4 * DELAY) * system clock.

3.14.7.7 A/D Sampling Register (ADC_EXTSMPT)

Register	Offset	R/W	Description	Reset Value
ADC_EXTSMPT	ADC_BA+0x48	R/W	A/D Sampling Time Counter Register	0x0000_0200

Bits	Description	
[31:10]	Reserved	Reserved.
[9:0]	EXTSMPT	Additional ADC Sample Clock If the ADC input is unstable, user can set this register to increase the sampling time to get a stable ADC input signal. The default sampling time is 0x200 ADC clocks. 0x200 ADC clock number will be inserted to lengthen the sampling clock. Set 0 will disable ADC

3.14.7.8 A/D PWM Sequential Register (ADC_SEQCTL)

Register	Offset	R/W	Description	Reset Value
ADC_SEQCTL	ADC_BA+0x4C	R/W	A/D PWM Sequential Mode Control Register	0x0000_0000

Bits	Description	
[31:20]	Reserved	Reserved.
[19:16]	TRG2CTL	PWM Trigger Source Selection For TRG2CTL[3:2] 00 = PWM Trigger source is PWM0_CH0. 01 = PWM Trigger source is PWM0_CH 2. 10 = PWM Trigger source is PWM0_CH 4. 11 = PWM Trigger source is PWM0_CH 6. PWM Trigger Type Selection for TRG2CTL[1:0] 00 = Rising of the selected PWM. 01 = Center of the selected PWM. 10 = Falling of the selected PWM. 11 = Period of the selected PWM. **Note:** PWM trigger source is valid for 1-shunt type and 2/3-shunt type and one channel mode .
[15:12]	Reserved	Reserved.
[11:8]	TRG1CTL	PWM Trigger Source Selection For TRG1CTL[3:2] 00 = PWM Trigger source is PWM0_CH 0. 01 = PWM Trigger source is PWM0_CH 2. 10 = PWM Trigger source is PWM0_CH 4. 11 = PWM Trigger source is PWM0_CH 6. PWM Trigger Type Selection for TRG1CTL[1:0] 00 = Rising of the selected PWM. 01 = Center of the selected PWM. 10 = Falling of the selected PWM. 11 = Period of the selected PWM. **Note:** PWM trigger source is valid for 1-shunt and 2/3-shunt type and one channel.
[7]	Reserved	Reserved
[6]	SEQ_ONE_CH_EN	PWM sequence in ADC one channel 1:enable function 0:disable function
[5]	TRG_SEL	TRG1CTL or TRG2CTL select for 1-shunt sequential mode. 0 = using TRG1CTL to trigger sequential conversion; 1 = using TRG2CTL to trigger sequential conversion;
[4]	DELAY_EN	1-shunt and 2/3 shunt mode: ADC delay time inserted before 2nd conversion. 0 = ADC delay time don't insert delay time; 1 = ADC delay time inserted before each conversion. **Note:** One channel pwm sequence mode : delay_en set "0",need not delay;
[3:2]	MODESEL	ADC Sequential Mode Selection 00 = Issue ADC_INT after Channel 0 then Channel 1 conversion finishes when SEQEN =1. 01 = Issue ADC_INT after Channel 1 then Channel 2 conversion finishes when SEQEN =1.

		10 = Issue ADC_INT after Channel 0 then Channel 2 conversion finishes when SEQEN =1. 11 = reserved
[1]	SEQTYPE	ADC Sequential Mode Type 0 = 2/3-shunt type 1 = 1-shunt type
[0]	SEQEN	ADC Sequential shut Mode Enable Bit When ADC sequential mode is enabled, two of three ADC channels from 0 to 2 will automatically convert analog data in the sequence of channel [0, 1] or channel[1, 2] or channel[0, 2] defined by MODESEL (ADC_SEQCTL[3:2]). 0 = ADC sequential shut mode Disabled. 1 = ADC sequential shut mode Enabled.

3.14.7.9 A/D PWM Sequential Mode Result Register (ADC_SEQDAT1/2)

Register	Offset	R/W	Description	Reset Value
ADC_SEQDAT1	ADC_BA+0x50	R	A/D PWM Sequential Mode First Result Register1	0x0000_0000
ADC_SEQDAT2	ADC_BA+0x54	R	A/D PWM Sequential Mode Second Result Register1	0x0000_0000

Bits	Description	
[31:18]	Reserved	Reserved.
[17]	VALID	Valid Flag This bit is set to 1 when ADC conversion is completed and cleared by hardware after the ADC_SEQDATx register is read. 0 = Data in RESULT[11:0] bits not valid. 1 = Data in RESULT[11:0] bits valid.
[16]	OV	Over Run Flag If converted data in RESULT[11:0] has not been read before the new conversion result is loaded to this register, OV is set to 1. It is cleared by hardware after the ADC_SEQDATx register is read. 0 = Data in RESULT[11:0] is recent conversion result. 1 = Data in RESULT[11:0] overwritten.
[15:12]	Reserved	Reserved.
[11:0]	RESULT	A/D PWM Sequential Mode Conversion Result This field contains conversion result of ADC.

3.14.7.10 ADC Control Register 2 (ADC_CTL2)

Register	Offset	R/W	Description	Reset Value
ADC_CTL2	ADC_BA+0x58	R/W	A/D Control Register 2	0x7552_0A00

Bits	Description	
[31:30]	Reserved	Reserved.
[29:28]	ICTL_CMP	ADC_ICTL_CMP, Default : 3
[27:26]	Reserved	Reserved.
[25:24]	ICTL_VCM	ADC_ICTL_VCM, Default : 1
[23:22]	Reserved	Reserved.
[21:20]	ICTL_VREF	ADC_ICTL_VREF, Default : 1
[19:18]	Reserved	Reserved.
[17]	SEL_VREF	ADC_SEL_VREF, Default : 1
[16]	EN_BUFTST	ADC_EN_BUFTST, Default : 0
[15:14]	Reserved	Reserved.
[13:11]	CLK_DIV_DUTY	ADC Clock Divider Could not be set as 0. Default : 1
[10:8]	CLK_DIV	ADC Clock Divider Could not be set as 0. Default : 2
[7:3]	Reserved	Reserved.
[2]	DMA_EN	ADC DMA enable, default:0
[1]	SEL_SH	1: External Sample Hold (SH) width with HOLD2(Internal Signal) 0: SH is SH. Default : 0
[0]	TEST_MODE	ADC Test enable

3.14.7.11 ADC Bias Voltage Control Register(ADC_BV_CTL)

Register	Offset	R/W	Description	Reset Value
ADC_BV_CTL	ADC_BA+0x6C	R/W	A/D Bias Voltage Control Register	0x0000_0100

Bits	Description	
[31:9]	Reserved	Reserved
[8:3]	ADC_REFGEN_CTLV	Bias voltage resistance array control bit ; Default:6'h20
[2]	ADC_FLTR_RES_SHRT	Channel voltage test enable ; Default:1'b0
[1]	ADC_FLTR_RES_ACT	ADC channel BUF enable ; Default:1'b0
[0]	ADC_REFGEN_EN	Bias voltage module enable signal ; Default:1'b0

3.15 Serial Peripheral Interface (SPI)

3.15.1 Overview

The SSP is a master or slave interface for synchronous serial communication with peripheral devices that have either Motorola SPI, National Semiconductor Microwire or Texas Instruments synchronous serial interfaces.

The SSP performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information through the AMBA APB interface. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. Serial data is transmitted on SSPTXD and received on SSPRXD.

The SSP includes a programmable bit rate clock divider and prescaler to generate the serial output clock SSPCLKOUT from the input clock SSPCLK. Bit rates are supported to 2MHz and higher, subject to choice of frequency for SSPCLK and the maximum bit rate is determined by peripheral devices.

The SSP operating mode, frame format, and size are programmed through the control registers CTL_REG_0 and CTL_REG_1.

Four individually maskable interrupt outputs are generated:

- TX_INT requests servicing of the transmit buffer
- RX_INT requests servicing of the receive buffer
- RX_OV_INT indicates an overrun condition in the receive FIFO
- RX_TM_INT indicates that a timeout period expired while data was present in the receive FIFO

A single combined interrupt, SSP_COM_INT output, is asserted if any of the individual interrupts are asserted and unmasked.

In addition to the above interrupts, a set of DMA signals are provided for interfacing with a DMA controller.

Depending on the operating mode selected, the SSPFSSOUT output operates as an active HIGH frame synchronization output for Texas Instruments synchronous serial frame format or an active LOW slave select for SPI and Microwire.

3.15.2 Features

- Support 3 SPI device
- Support 4wire SPI device
- The PrimeCell SSP has the following features:
  - Compliance to the AMBA Specification (Rev 2.0)
  - Master or slave operation
  - Programmable clock bit rate and prescale
  - Separate transmit and receive first-in, first-out memory buffers, 16 bits wide, 8 locations deep
  - Programmable choice of interface operation, SPI, Microwire, or TI synchronous serial
  - Programmable data frame size from 4 to 16 bits
  - Independent masking of transmit FIFO, receive FIFO, and receive overrun interrupts
  - Support for Direct Memory Access (DMA)
  - transmit tx fifo data item in MSB/ LSB first order due to register configuration
  - receive rx fifo data item in MSB/ LSB first order due to register configuration
- The features of the Motorola SPI-compatible interface are:
  - full duplex, four-wire synchronous transfers
  - programmable clock polarity and phase
- The features of the National Semiconductor Microwire interface are:
  - half-duplex transfer using 8-bit control message
- The features of the Texas Instruments synchronous serial interface are:
  - full-duplex, four-wire synchronous transfer
  - transmit data pin tristateable when not transmitting.

3.15.3 Timing Parameters

1. SPI IP As Master

- SPO=0，SPH=0

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rising edge	-	2T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	20.83	T	-	ns
t_{SCLK}	SCLK period	2T	2T	-	ns
t_{DSU}	Setup time, DIN vaild before SCLK rising edge	-	-	-	-
t_{DHO}	Hold time, DIN vaild after SCLK faliing edge	-	-	-	-

- SPO=0，SPH=1

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	-	T	-	ns
t_{CSLSCL}	CS falling edge to first SCLK faliing edge	-	2T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	-	2T	-	ns
t_{SCLK}	SCLK period	2T	2T	-	ns
t_{DSU}	Setup time, DIN vaild before SCLK faliing edge	-	-	-	-
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	-	-	-	-

- SPO=1，SPH=0

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	-	3T	-	ns
t_{CSLSCL}	CS falling edge to first SCLK faliing edge	-	2T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	-	2T	-	ns
t_{SCLK}	SCLK period	2T	2T	-	ns
t_{DSU}	Setup time, DIN vaild before SCLK faliing edge	-	-	-	-
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	-	-	-	-

- SPO=1，SPH=1

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	-	2T	-	ns
t_{CSLSCL}	CS falling edge to first SCLK faliing edge	-	T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	-	2T	-	ns
t_{SCLK}	SCLK period	2T	2T	-	ns
t_{DSU}	Setup time, DIN vaild before SCLK faliing edge	-	-	-	-
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	-	-	-	-

2. SPI IP As Slave

- SPO=0，SPH=0

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	6T	6T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	3T	3T	-	ns
t_{SCLK}	SCLK period	6T	6T	-	ns
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	3T	3T	-	ns

- SPO=0，SPH=1

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	3T	3T	-	ns
t_{CSLSCL}	CS falling edge to first SCLK faliing edge	6T	6T	-	ns
t_{SCLK}	SCLK period	6T	6T	-	ns
t_{SCLCS}	Final SCLK falling edge to CS rising edge	6T	6T	-	ns
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	3T	3T	-	ns

- SPO=1，SPH=0

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	-	9T	-	ns
t_{CSLSCL}	CS falling edge to first SCLK faliing edge	-	6T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	-	6T	-	ns
t_{SCLK}	SCLK period	6T	6T	-	ns

t_{DSU}	Setup time, DIN vaild before SCLK faliing edge	-	-	-	-
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	-	-	-	-

- SPO=1，SPH=1

Symbol	Description	Parameter			Unit
		Min	Typ	Max	
T	SPI IP operating cycle	20.83	T	-	ns
t_{CSLSCH}	CS falling edge to first SCLK rsiing edge	-	6T	-	ns
t_{CSLSCL}	CS falling edge to first SCLK faliing edge	-	3T	-	ns
t_{SCLCSH}	Final SCLK falling edge to CS rising edge	-	6T	-	ns
t_{SCLK}	SCLK period	6T	6T	-	ns
t_{DSU}	Setup time, DIN vaild before SCLK faliing edge	-	-	-	-
t_{DHO}	Hold time, DIN vaild after SCLK rising edge	-	-	-	-

3.15.4 Block Diagram

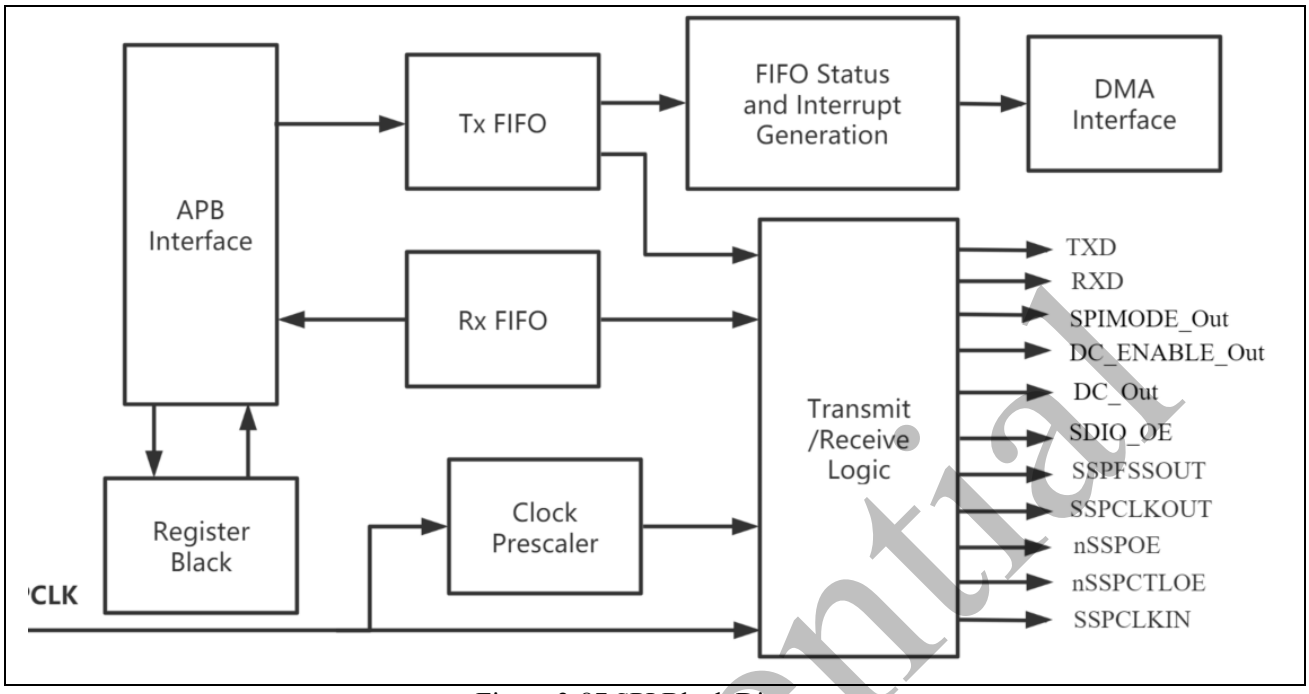


Figure 3-87 SPI Block Diagram

APB interface

It generates read and write decodes for accesses to status and control registers, and transmit and receive FIFO memories.

Register block

It stores data written, or to be read, across the APB interface.

Clock prescaler

It provides the serial output clock SSPCLKOUT with divide SSPCLK by a factor of 2-254.

TX/RX FIFO

TX/RX FIFO is a 16-bit wide, 8-locations deep, First-In, First-Out (FIFO) memory buffer.

Transmit and receive logic

When working as a master, the transmit logic reads the data from TX FIFO successively, it converts parallel input signals into serial output signals. Then the data stream output to the external slave by SSPTXD pin. Moreover, the master can receive the data from slave though SSPRXD pin and convert serial signals to parallel signals. The parallel signals will be extracted and stored into the RX FIFO.

When working as a slave, the SSPCLKIN clock is provided by an attached master and used to time its transmission and reception sequences.

FIFO state and interrupt generation logic

3.15.5 Functional Description

This chapter declk_rateibes the functional behavior of SSP Synchronous Serial Port in detail. The functions of the SSP are declk_rateibed in the following sections:

- AMBA APB interface
- Register block
- Clock prescaler
- Transmit FIFO
- Receive FIFO
- Transmit and receive logic
- Interrupt generation logic
- Synchronizing registers and logic
- DMA interface on page

3.15.5.1 AMBA APB interface

The AMBA APB interface generates read and write decodes for accesses to status and control registers, and transmit and receive FIFO memories.

The AMBA APB is a local secondary bus that provides a low-power extension to the higher bandwidth AMBA Advanced High-performance Bus (AHB) within the AMBA system hierarchy. The AMBA APB groups narrow-bus peripherals to avoid loading the system bus and provides an interface using memory-mapped registers, which are accessed under programmed control.

3.15.5.2 Register block

The register block stores data written or to be read across the AMBA APB interface.

3.15.5.3 Clock prescaler

When configured as a master, an internal prescaler, comprising two free-running reloadable serially linked counters, is used to provide the serial output clock SSPCLKOUT.

You can program the clock prescaler, through the CLK_PSC register, to divide SSPCLK by a factor of 2 to 254 in steps of two. By not utilizing the least significant bit of the CLK_PSC register, division by an odd number is not possible and this ensures a symmetrical (equal mark space ratio) clock is generated.

The output of the prescaler is further divided by a factor of 1 to 256, through the programming

of the CTL_REG_0 control register, to give the final master output clock SSPCLKOUT.

3.15.5.4 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. CPU data written across the AMBA APB interface are stored in the buffer until read out by the transmit logic.

When configured as a master or a slave parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master respectively, through the SSPTXD pin.

SSP supports converting transmit FIFO data item to be transmitted in an MSB/ LSB first order as tx_lsb configuration and compatible with data_size(data size select) for both SSP master and slave.

As default, tx_lsb=0, data item of transmit FIFO is transmitted in an MSB to LSB order.

If set tx_lsb=1, data item of transmit FIFO is transmitted in an LSB to MSB order.

3.15.5.5 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface are stored in the buffer until read out by the CPU across the AMBA APB interface.

When configured as a master or slave, serial data received through the SSPRXD pin is registered prior to parallel loading into the attached slave or master receive FIFO respectively.

SSP supports converting receive FIFO data item to be read in an MSB/ LSB first order as rx_lsb configuration and compatible with data_size for both SSP master and slave.

As default, rx_lsb=0, data item of receive FIFO is read in an MSB to LSB order, that is first received bit put into MSB, and last received bit put into LSB.

If set rx_lsb=1, data item of receive FIFO is read in an LSB to MSB order, that is first received bit put into LSB, and last received bit put into MSB.

3.15.5.6 Transmit and receive logic

When configured as a master, the clock to the attached slaves is derived from a divided down version of SSPCLK through the prescaler operations declk_rateibed previously. The master transmit logic successively reads a value from its transmit FIFO and performs parallel to serial conversion on it. Then the serial data stream and frame control signal, synchronized to SSPCLKOUT, are output through the SSPTXD pin to the attached slaves. The master receive

logic performs serial to parallel conversion on the incoming synchronous SSPRXD data stream, extracting and storing values into its receive FIFO, for subsequent reading through the APB interface.

When configured as a slave, the SSPCLKIN clock is provided by an attached master and used to time its transmission and reception sequences. The slave transmit logic, under control of the master clock, successively reads a value from its transmit FIFO, performs parallel to serial conversion, then output the serial data stream and frame control signal through the slave SSPTXD pin. The slave receive logic performs serial to parallel conversion on the incoming SSPRXD data stream, extracting and storing values into its receive FIFO, for subsequent reading through the APB interface.

3.15.5.7 Interrupt generation logic

Four individual maskable, active HIGH interrupts are generated by the SSP. A combined interrupt output is also generated as an OR function of the individual interrupt requests.

You can use the single combined interrupt with a system interrupt controller that provides another level of masking on a per-peripheral basis. This allows use of modular device drivers that always know where to find the interrupt source control register bits.

The individual interrupt requests could also be used with a system interrupt controller that provides masking for the outputs of each peripheral. In this way, a global interrupt controller service routine would be able to read the entire set of sources from one wide register in the system interrupt controller. This is attractive where the time to read from the peripheral registers is significant compared to the CPU clock speed in a real-time system.

The peripheral supports both the above methods.

The transmit and receive dynamic data-flow interrupts, TX_INT and RX_INT, are separated from the status interrupts so that data can be read or written in recpolnse to the FIFO trigger levels.

3.15.5.8 DMA interface

The PrimeCell SSP provides an interface to connect to a DMA controller.

3.15.5.9 Synchronizing registers and logic

The SSP supports both asynchronous and synchronous operation of the clocks, PCLK and SSPCLK. Synchronization registers and hand shaking logic have been implemented, and are active at all times. This has a minimal impact on performance or area. Synchronization of

control signals is performed on both directions of data flow, that is from the PCLK to the SSPCLK domain and from the SSPCLK to the PCLK domain.

3.15.6 SSP operation

The operation of the SSP is declk_rateibed in the following sections:

- Interface reset
- Configuring the SSP
- Enable SSP operation
- Clock ratios
- Programming the CTL_REG_0 Control Register
- Programming the CTL_REG_1 Control Register
- Frame format
- Texas Instruments synchronous serial frame format
- Motorola SPI frame format
- National Semiconductor Microwire frame format
- Examples of master and slave configurations
- DMA interface

3.15.6.1 Interface reset

The SSP is reset by the global reset signal PRESETn and a block-specific reset signal nSSPRST. An external reset controller must use PRESETn to assert nSSPRST asynchronously and negate it synchronously to SSPCLK. PRESETn must be asserted LOW for a period long enough to reset the slowest block in the on-chip system, and then taken HIGH again. The SSP requires PRESETn to be asserted LOW for at least one period of PCLK.

3.15.6.2 Configuring the SSP

Following reset, the SSP logic is disabled and must be configured when in this state.

Control registers CTL_REG_0 and CTL_REG_1 need to be programmed to configure the peripheral as a master or slave operating under one of the following protocols:

- Motorola SPI
- Texas Instruments SSI
- National Semiconductor.

The bit rate, derived from the external SSPCLK, requires the programming of the clock prescale register CLK_PSC.

3.15.6.3 Enable SSP operation

You can either prime the transmit FIFO, by writing up to eight 16-bit values when the PrimeCell SSP is disabled, or allow the transmit FIFO service request to interrupt the CPU. Once enabled, transmission or reception of data begins on the transmit (SSPTXD) and receive (SSPRXD) pins.

3.15.6.4 Clock ratios

There is a constraint on the ratio of the frequencies of PCLK to SSPCLK. The frequency of SSPCLK must be less than or equal to that of PCLK. This ensures that control signals from the SSPCLK domain to the PCLK domain are certain to get synchronized before one frame duration:

FSSPCLK <= FPCLK.

In the slave mode of operation, the SSPCLKIN signal from the external master is double synchronized and then delayed to detect an edge. It takes three SSPCLKs to detect an edge on SSPCLKIN. SSPTXD has less setup time to the falling edge of SSPCLKIN on which the master is sampling the line. The setup and hold times on SSPRXD with reference to SSPCLKIN must be more conservative to ensure that it is at the right value when the actual sampling occurs within the SSPMS. To ensure correct device operation, SSPCLK must be at least 8 times faster than the maximum expected frequency of SSPCLKIN.

The frequency selected for SSPCLK must accommodate the desired range of bit clock rates. The ratio of minimum SSPCLK frequency to SSPCLKOUT maximum frequency in the case of the slave mode is 8 and for the master mode it is 2.

To generate a maximum bit rate of 1.8432Mbps in the Master mode, the frequency of SSPCLK must be at least 3.6864MHz. With an SSPCLK frequency of 3.6864MHz, the CLK_PSC register has to be programmed with a value of 2 and the clk_rate[7:0] field in the CTL_REG_0 register needs to be programmed as zero.

To work with a maximum bit rate of 1.8432Mbps in the slave mode, the frequency of SSPCLK must be at least 14.75MHz. With an SSPCLK frequency of 14.75MHz, the CLK_PSC register can be programmed with a value of 2 and the clk_rate[7:0] field in the CTL_REG_0 register can be programmed as 3. Similarly the ratio of SSPCLK maximum frequency to SSPCLKOUT minimum frequency is 254 x 256.

The minimum frequency of SSPCLK is governed by the following equations, both of which have to be satisfied:

FSSPCLK(min) >= 2 x FSSPCLKOUT(max) [for master mode]

FSSPCLK(min) >= 8 x FSSPCLKIN(max) [for slave mode]

The maximum frequency of SSPCLK is governed by the following equations, both of which have to be satisfied:

FSSPCLK(max) <= 254 x 256 x FSSPCLKOUT(min) [for master mode]

FSSPCLK(max) <= 254 x 256 x FSSPCLKIN(min) [for slave mode]

3.15.6.5 Programming the CTL_REG_0 Control Register

The CTL_REG_0 register is used to:

- program the serial clock rate
- select one of the three protocols
- select the data word size (where applicable).

The Serial Clock Rate (clk_rate) value, in conjunction with the CLK_PSC clock prescale divisor value (clk_psc), is used to derive the SSP transmit and receive bit rate from the external SSPCLK.

The frame format is programmed through the FRF bits and the data word size through the data_size bits.

Bit phase and polarity, applicable to Motorola SPI format only, are programmed through the cpha and cpol bits.

3.15.6.6 Programming the CTL_REG_1 Control Register

The CTL_REG_1 register is used to:

- select master or slave mode
- enable a loop back test feature
- enable the SSP peripheral
- MSB/ LSB of tx fifo data item transmit first configuration
- MSB/ LSB of rx fifo data item receive first configuration

To configure the SSP as a master, clear the CTL_REG_1 register master or slave selection bit (MS) to 0, which is the default value on reset.

Setting the CTL_REG_1 register MS bit to 1 configures the SSP as a slave. When configured as a slave, enabling or disabling of the SSP SSPTXD signal is provided through the CTL_REG_1 slave mode SSPTXD output disable bit (sl_out_dis). This can be used in some multi-slave environments where masters might parallel broadcast.

To enable the operation of the PrimeCell SSP set the Synchronous Serial Port Enable (SSE)

bit to 1.

To fix the transmit order of tx fifo data item, set tx_lsb, 0 for MSB to LSB, 1 for LSB to MSB.

To fix the receive order of rx fifo data item, set rx_lsb, 0 for MSB to LSB, 1 for LSB to MSB.

3.15.6.7 Bit rate generation

The serial bit rate is derived by dividing down the input clock SSPCLK. The clock is first divided by an even prescale value clk_psc from 2 to 254, which is programmed in CLK_PSC. The clock is further divided by a value from 1 to 256, which is 1 + clk_rate, where clk_rate is the value programmed in CTL_REG_0.

The frequency of the output signal bit clock SSPCLKOUT is defined below:

$$FSSPCLKOUT = \frac{f_{ssp\_apb\_clk}}{clk\_psc \times (1 + clk\_rate)}$$

For example, if SSPCLK is 3.6864MHz, and clk_psc = 2, then SSPCLKOUT has a frequency range from 7.2kHz to 1.8432MHz.

3.15.6.8 Frame format

Each data frame is between 4 and 16 bits long depending on the size of data programmed, and is transmitted starting with the MSB/ LSB(register configurable). There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Motorola SPI
- National Semiconductor Microwire.
- Motorola SPI(3wire-9bit)

For all three formats, the serial clock (SSPCLKOUT) is held inactive while the SSP is idle, and transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSPCLKOUT is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Motorola SPI and National Semiconductor Microwire frame formats, the serial frame (SSPFSSOUT) pin is active LOW, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the SSPFSSOUT pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For

this frame format, both the SSP and the off-chip slave device drive their output data on the rising edge of SSPCLKOUT, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor Microwire format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, recpolnds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

3.15.6.9 Texas Instruments synchronous serial frame format

Figure 3-88 shows the Texas Instruments synchronous serial frame format for a single transmitted frame, tx_lsb=0 and rx_lsb=0.

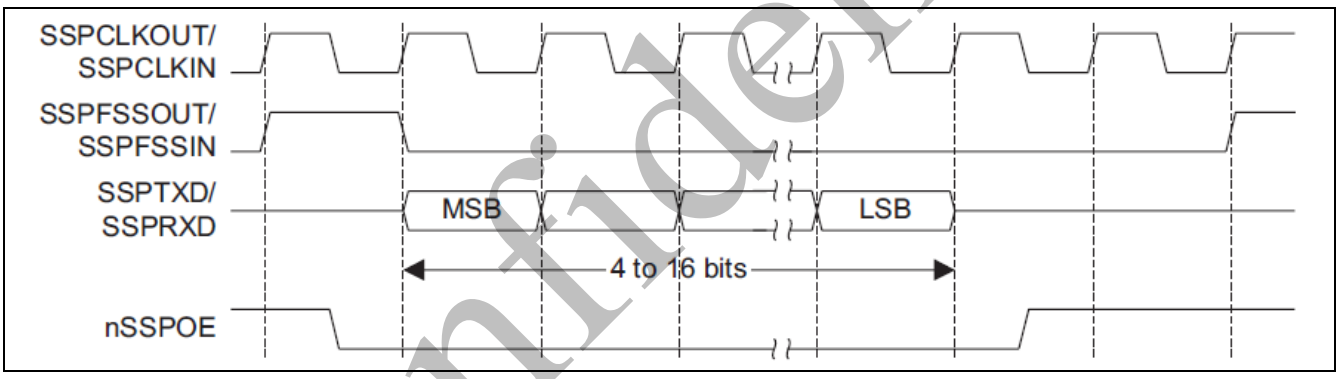


Figure 3-88 Texas Instruments synchronous serial frame format (single transfer), tx_lsb=0, rx_lsb=0

In this mode, SSPCLKOUT and SSPFSSOUT are forced LOW, and the transmit data line SSPTXD is tristated whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, SSPFSSOUT is pulsed HIGH for one SSPCLKOUT period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. As tx_lsb=0 and rx_lsb=0, on the next rising edge of SSPCLKOUT, the MSB of 4 to 16-bit data frame is shifted out on the SSPTXD pin. Likewise, the MSB of the received data is shifted onto the SSPRXD pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSPCLKOUT. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSPCLKOUT after the LSB has been latched.

Figure 3-89 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

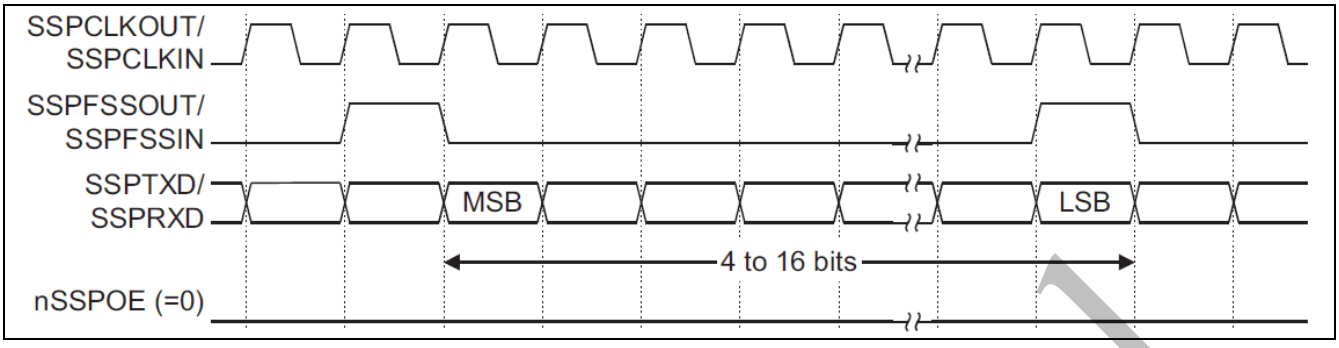
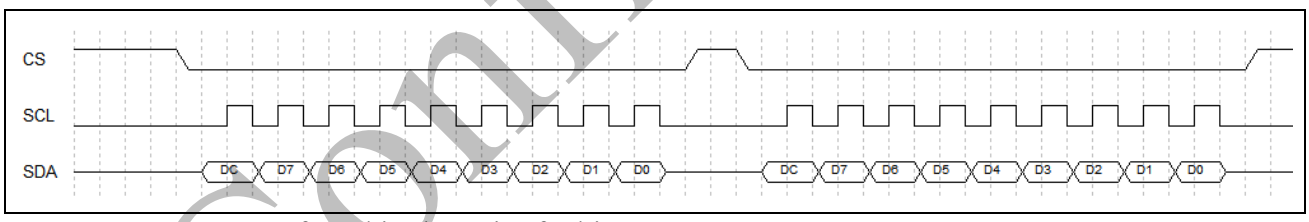


Figure 3-89 TI synchronous serial frame format (continuous transfer), tx_lsb=0, rx_lsb=0

3.15.6.10 3 wire SPI format

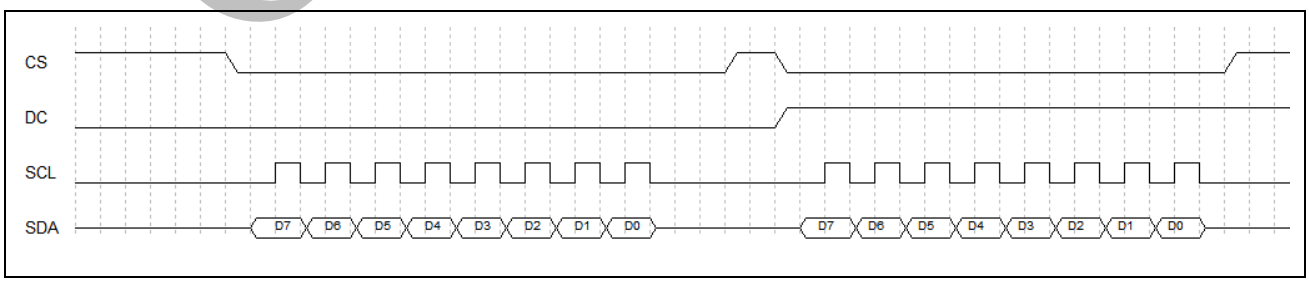
In 3-wire mode, the distinction between data and command is determined by the value of the Data/Command (DC) bit. When DC=0, it indicates that the data being transmitted to the slave is a command. When DC=1, it indicates that the data being transmitted to the slave is a parameter.

DC	DATA
0	command
1	parameter



Data are transferred in the unit of 9 bits.

Figure 3-90 3-wire SPI write/read operation, with no dc out



Data / Command is recognized with DC pin. Data are transferred in the unit of 8 bits.

Figure 3-91 3-wire SPI write/read operation, with dc out

3.15.6.11 Motorola SPI frame format

The Motorola SPI interface is a four-wire interface where the SSPFSSOUT signal behaves as a slave select. The main feature of the Motorola SPI format is that the inactive state and phase of the SSPCLKOUT signal are programmable through the cpol and cpha bits within the SSPclk_rate0 control register.

cpol, clock polarity

When the cpol clock polarity control bit is LOW, it produces a steady state low value on the SSPCLKOUT pin. If the cpol clock polarity control bit is HIGH, a steady state high value is placed on the SSPCLKOUT pin when data is not being transferred.

cpha, clock phase

The cpha control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.

When the cpha phase control bit is LOW, data is captured on the first clock edge transition. If the cpha clock phase control bit is HIGH, data is captured on the second clock edge transition. For continuous back-to-back SPI transactions, if cpha=0, SSPSSFOUT turns high for one cycle of SSPCLKOUT for each SPI transaction; if cpha=1, SSPSSFOUT keeps low during continuous transmission. If master transmit fifo get empty, then SSPSSFOUT turns high before next master transmit fifo tansaction com.

If cpol^cpha=0, master and slave both sample SSPRXD on posedge of SSPCLKOUT/SSPCLKIN, and drive SSPTXD on negedge of SSPCLKOUT/SSPCLKIN.

If cpol^cpha=1, master and slave both sample SSPRXD on negedge of SSPCLKOUT/SSPCLKIN, and drive SSPTXD on posedge of SSPCLKOUT/SSPCLKIN.

3.15.6.12 Motorola SPI Format with cpol=0, cpha=0

Single and continuous transmission signal sequences for Motorola SPI format with cpol=0, cpha=0, tx_lsb=0 and rx_lsb=0 are shown in Figure 3-92 and Figure 3-93.

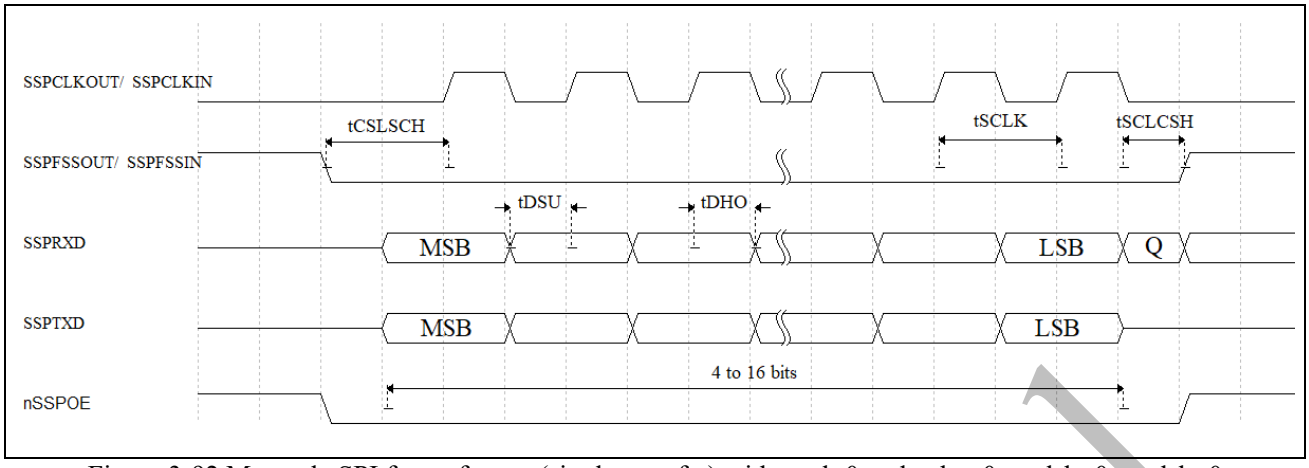


Figure 3-92 Motorola SPI frame format (single transfer) with cpol=0 and cpha=0, tx_lsb=0, rx_lsb=0

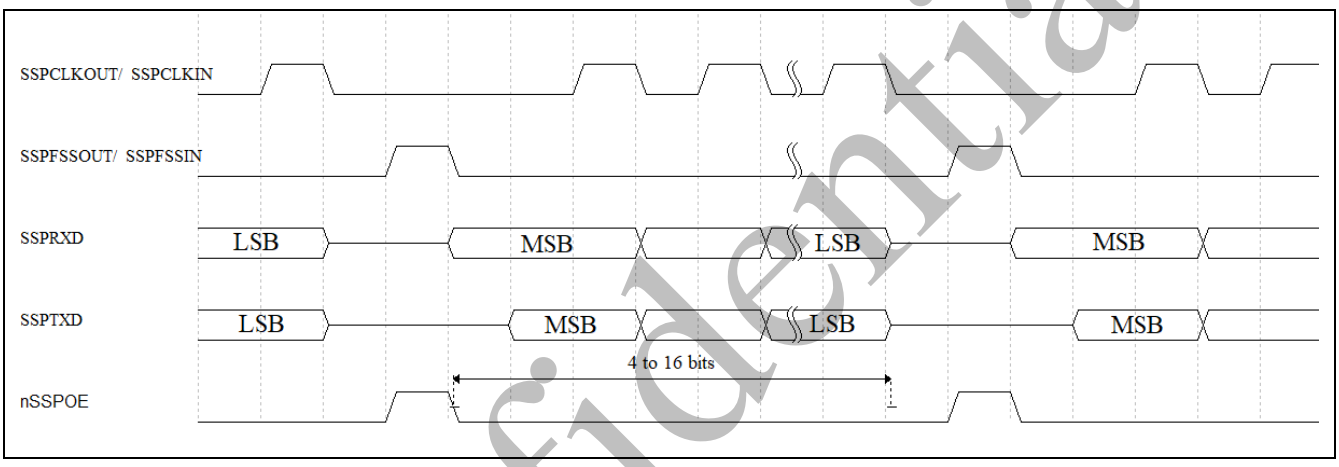


Figure 3-93 Motorola SPI frame format (continuous transfer) with cpol=0 and cpha=0, tx_lsb=0, rx_lsb=0

In this configuration, during idle periods:

- The SSPCLKOUT signal is forced LOW
- SSPFSSOUT is forced HIGH
- The transmit data line SSPTXD is arbitrarily forced LOW
- The nscpole pad enable signal is forced HIGH, making the transmit pad high impedance
- When the SSP is configured as a master, the nsspctloe line is driven LOW, enabling the SSPCLKOUT pad (active LOW enable)
- When the SSP is configured as a slave, the nsspctloe line is driven HIGH, disabling the SSPCLKOUT pad (active LOW enable)

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW. This causes slave data to be enabled onto the SSPRXD input line of the master. The nScpolE line is driven LOW, enabling the master SSPTXD output pad.

One half SSPCLKOUT period later, valid master data is transferred to the SSPTXD pin. Now that both the master and slave data have been set, the SSPCLKOUT master clock pin goes HIGH after one further half SSPCLKOUT period.

The data is now captured on the rising and propagated on the falling edges of the SSPCLKOUT signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSPFSSOUT signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the cpha bit is logic zero. Therefore the master device must raise the SSPFSSIN pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSPFSSOUT pin is returned to its idle state one SSPCLKOUT period after the last bit has been captured.

3.15.6.13 Motorola SPI Format with cpol=0, cpha=1

The transfer signal sequence for Motorola SPI format with cpol=0, cpha=1, tx_lsb=0 and rx_lsb=1 is shown in Figure 3-94.

In this configuration, during idle periods:

- The SSPCLKOUT signal is forced LOW
- SSPFSSOUT is forced HIGH
- The transmit data line SSPTXD is arbitrarily forced LOW
- The nscpole pad enable signal is forced HIGH, making the transmit pad high impedance
- When the SSP is configured as a master, the nsspctloe line is driven LOW, enabling the SSPCLKOUT pad (active LOW enable)
- When the SSP is configured as a slave, the nsspctloe line is driven HIGH, disabling the SSPCLKOUT pad (active LOW enable)

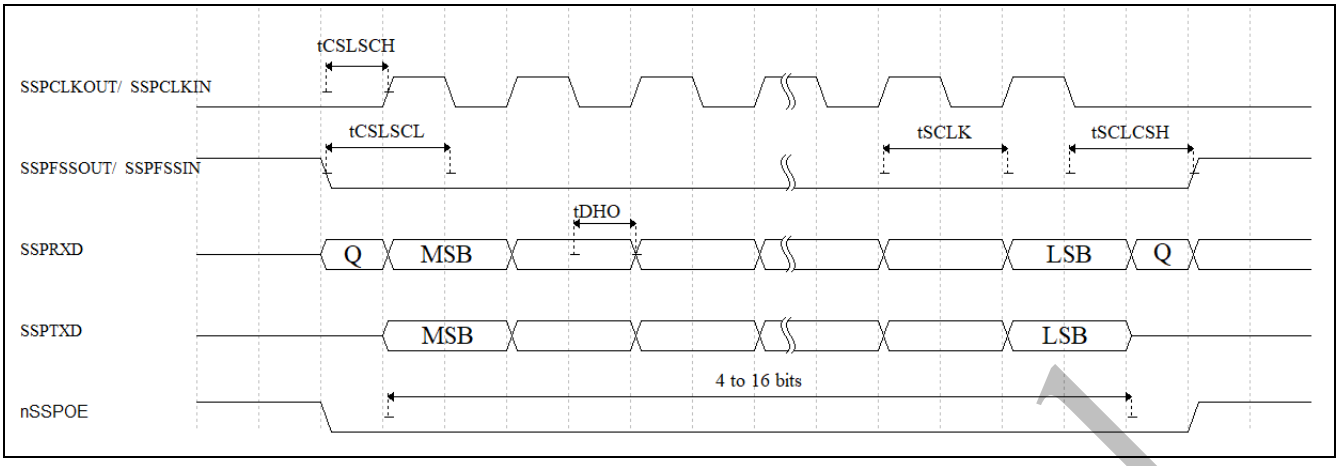


Figure 3-94 Motorola SPI frame format with cpol=0 and cpha=1, tx_lsb=0, rx_lsb=1

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW. The nScpolE line is driven LOW, enabling the master SSPTXD output pad. After a further one half SSPCLKOUT period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SSPCLKOUT is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSPCLKOUT signal.

In the case of a single word transfer, after all bits have been transferred, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured. For continuous back-to-back transfers, the SSPFSSOUT pin is held LOW between successive data words and termination is the same as that of the single word transfer.

3.15.6.14 Motorola SPI Format with cpol=1, cpha=0

The transmission signal sequences for Motorola SPI format with cpol=1, cpha=0, tx_lsb=1 and rx_lsb=1 are shown in Figure 3-95.

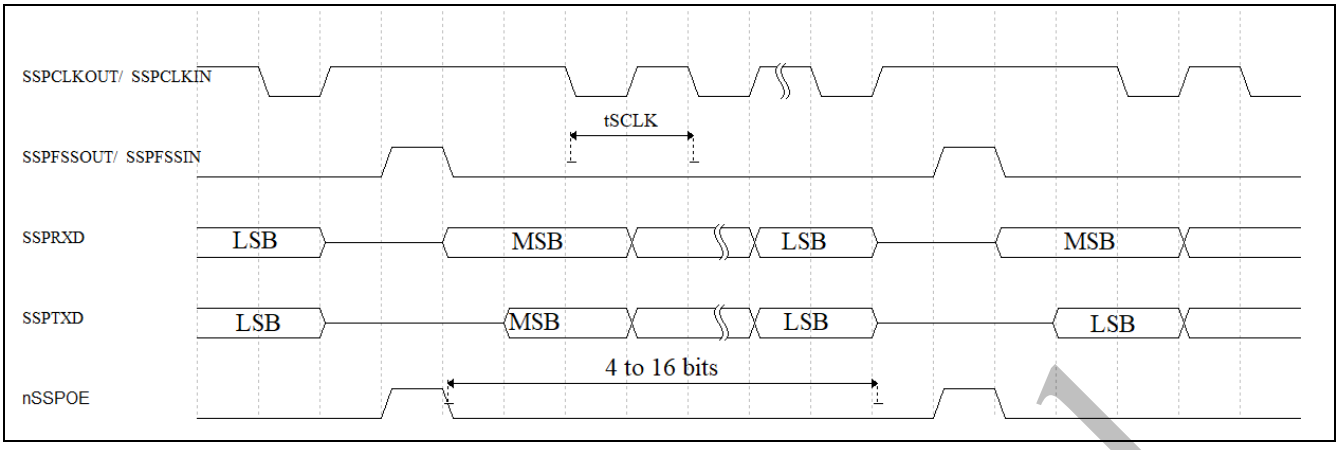


Figure 3-95 Motorola SPI frame format with cpol=1 and cpha=0, tx_lsb=1, rx_lsb=1 (continuous transfer)

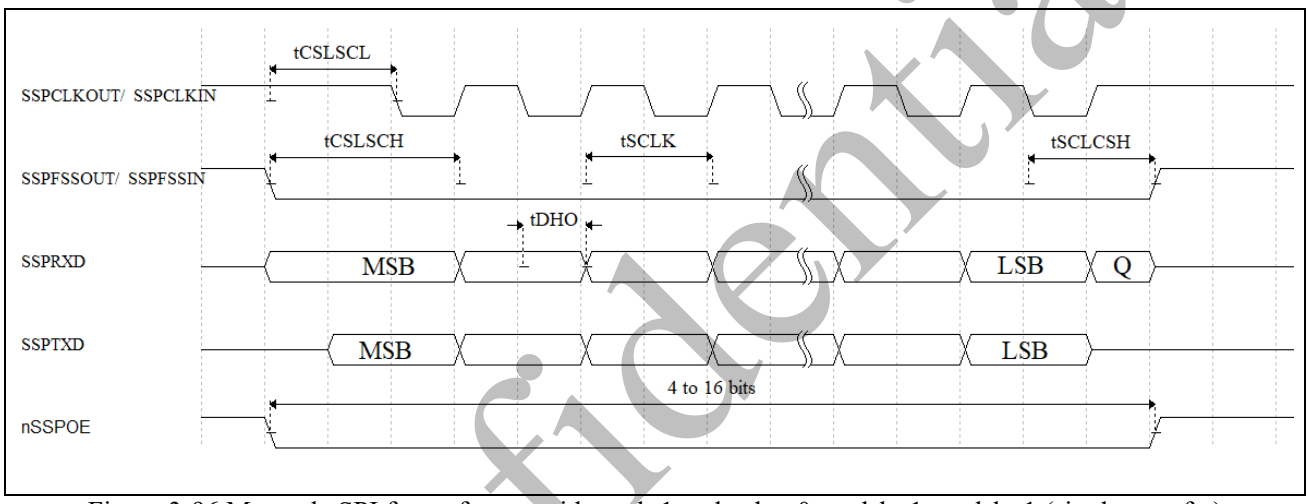


Figure 3-96 Motorola SPI frame format with cpol=1 and cpha=0, tx_lsb=1, rx_lsb=1 (single transfer)

In this configuration, during idle periods

- The SSPCLKOUT signal is forced HIGH
- SSPFSSOUT is forced HIGH
- The transmit data line SSPTXD is arbitrarily forced LOW
- The nscpole pad enable signal is forced HIGH, making the transmit pad high
- Impedance
- When the SSP is configured as a master, the nsspctloe line is driven LOW, enabling the SSPCLKOUT pad (active LOW enable)
- When the SSP is configured as a slave, the nsspctloe line is driven HIGH, disabling the SSPCLKOUT pad (active LOW enable)

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW, which causes slave data to be immediately transferred onto the SSPRXD line of the master. The nScpolE line is driven

LOW, enabling the master SSPTXD output pad.

One half period later, valid master data is transferred to the SSPTXD line. Now that both the master and slave data have been set, the SSPCLKOUT master clock pin becomes LOW after one further half SSPCLKOUT period. This means that data is captured on the falling edges and be propagated on the rising edges of the SSPCLKOUT signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSPFSSOUT signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the cpha bit is logic zero. Therefore the master device must raise the SSPFSSIN pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSPFSSOUT pin is returned to its idle state one SSPCLKOUT period after the last bit has been captured.

3.15.6.15 Motorola SPI Format with cpol=1, cpha=1

The transfer signal sequence for Motorola SPI format with cpol=1, cpha=1, tx_lsb=1 and rx_lsb=0 is shown in Figure 3-97.

In this configuration, during idle periods:

- The SSPCLKOUT signal is forced HIGH
- SSPFSSOUT is forced HIGH
- The transmit data line SSPTXD is arbitrarily forced LOW
- The nscpole pad enable signal is forced HIGH, making the transmit pad high impedance
- When the SSP is configured as a master, the nsspctloe line is driven LOW, enabling the SSPCLKOUT pad (active LOW enable)
- When the SSP is configured as a slave, the nsspctloe line is driven HIGH, disabling the SSPCLKOUT pad (active LOW enable)

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW. The nScpolE line is driven LOW, enabling the master SSPTXD output pad. After a further one half SSPCLKOUT period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SSPCLKOUT is enabled with a falling edge transition. Data is then captured on the

rising edges and propagated on the falling edges of the SSPCLKOUT signal.

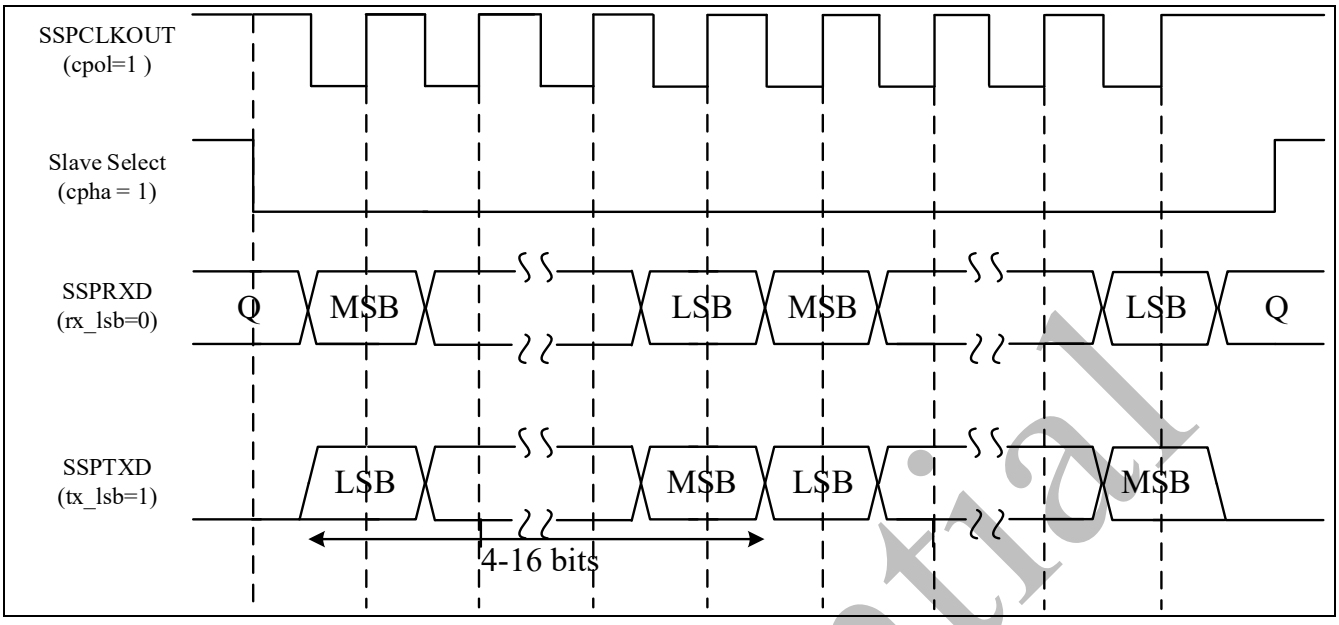


Figure 3-97 Motorola SPI frame format with cpol=1 and cpha=1, tx_lsb=1, rx_lsb=0

After all bits have been transferred, in the case of a single word transmission, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

For continuous back-to-back transmissions, the SSPFSSOUT pins remains in its active LOW state, until the final bit of the last word has been captured, and then returns to its idle state as declk_rateibed above.

For continuous back-to-back transfers, the SSPFSSOUT pin is held LOW between successive data words and termination is the same as that of the single word transfer.

3.15.6.16 National Semiconductor Microwire frame format

Figure 3-98 shows the National Semiconductor Microwire frame format (tx_lsb=0, rx_lsb=0), again for a single frame. Figure 9 shows the same format when back to back frames are transmitted.

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmission, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, recpolnds with the required data. The returned data is 4 to

16 bits in length, making the total frame length anywhere from 13 to 25 bits.

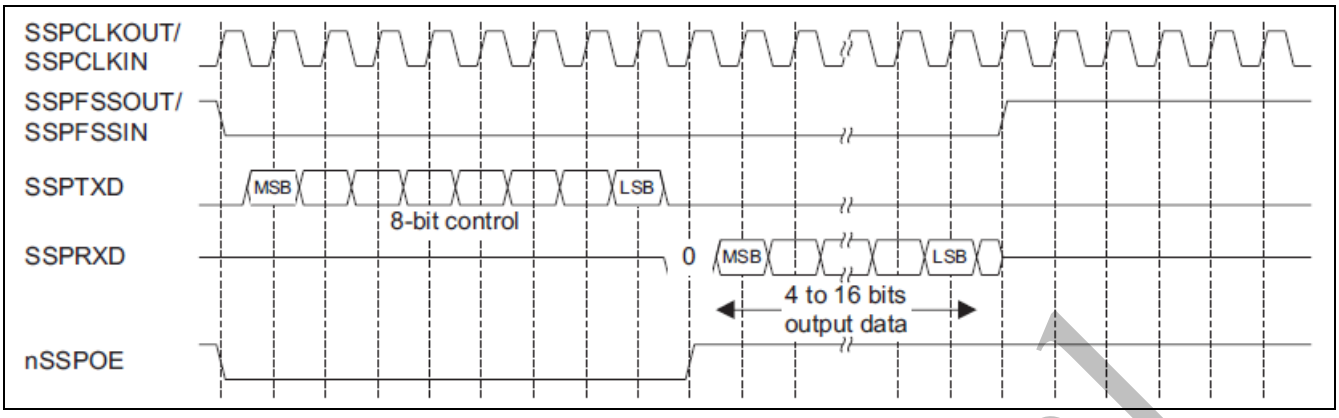


Figure 3-98 Microwire frame format (single transfer), tx_lsb=0 and rx_lsb=0

In this configuration, during idle periods:

- The SSPCLKOUT signal is forced LOW
- SSPFSSOUT is forced HIGH
- The transmit data line SSPTXD is arbitrarily forced LOW
- The nscpole pad enable signal is forced HIGH, making the transmit pad high impedance.

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSPFSSOUT causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame(tx_lsb=0) to be shifted out onto the SSPTXD pin. SSPFSSOUT remains LOW for the duration of the frame transmission. The SSPRXD pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSPCLKOUT. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave recpolnds by transmitting data back to the SSP. Each bit is driven onto SSPRXD line on the falling edge of SSPCLKOUT. The SSP in turn latches each bit on the rising edge of SSPCLKOUT. At the end of the frame, for single transfers, the SSPFSSOUT signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO. For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSPFSSOUT line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB(rx_lsb=0) of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SSPCLKOUT, after the LSB of the frame has been latched into the SSP.

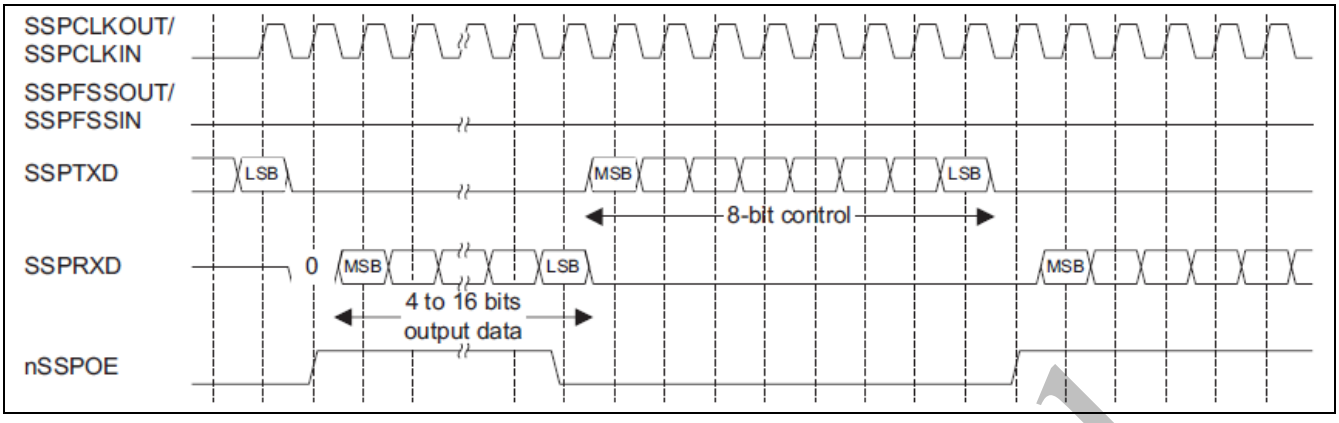


Figure 3-99 icrowire frame format (continuous transfers), tx_lsb=0 and rx_lsb=0

Setup and hold time requirements on SSPFSSIN with respect to SPCLKIN in Microwire mode.

In the Microwire mode, the PrimeCell SSP slave samples the first bit of receive data on the rising edge of SSPCLKIN after SSPFSSIN has gone LOW. Masters that drive a free-running SSPCKLIN must ensure that the SSPFSSIN signal has sufficient setup and hold margins with respect to the rising edge of SSPCLKIN.

Figure 3-100 on illustrates these setup and hold time requirements. With respect to the SSPCLKIN rising edge on which the first bit of receive data is to be sampled by the SSP slave, SSPFSSIN must have a setup of at least two times the period of SSPCLKIN on which the SSP operates. With respect to the SSPCLKIN rising edge previous to this edge, SSPFSSIN must have a hold of at least one SSPCLKIN period.

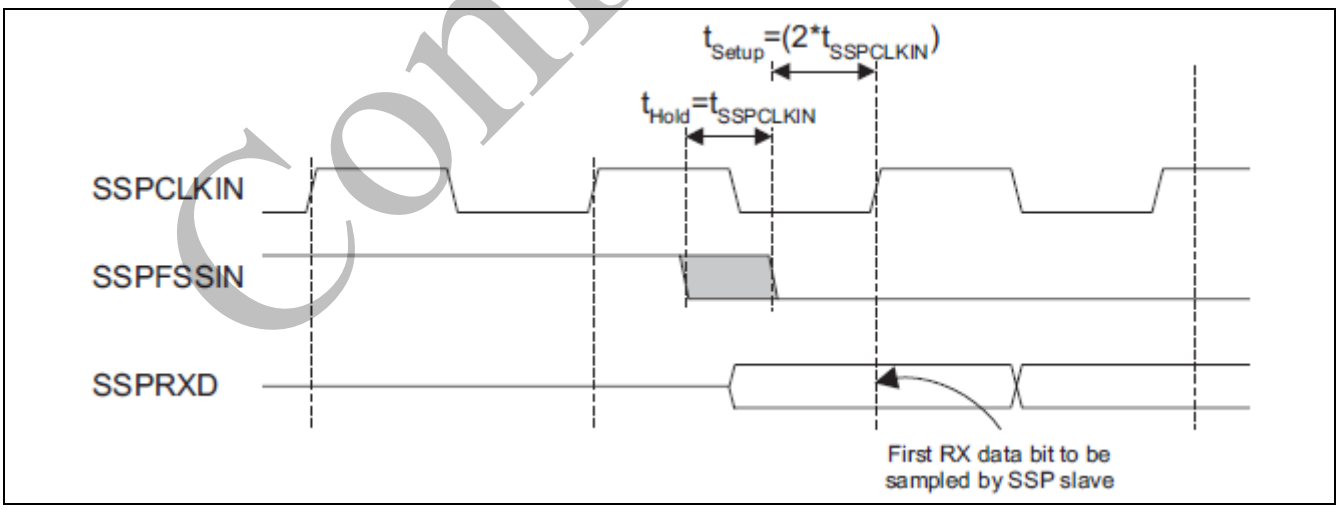


Figure 3-100 Microwire frame format, SSPFSSIN input setup and hold requirements

3.15.6.17 Examples of master and slave configurations

Figure 3-101, Figure 3-102 and Figure 3-103 show how the SSP peripheral can be connected to other synchronous serial peripherals, when it is configured as a master or slave.

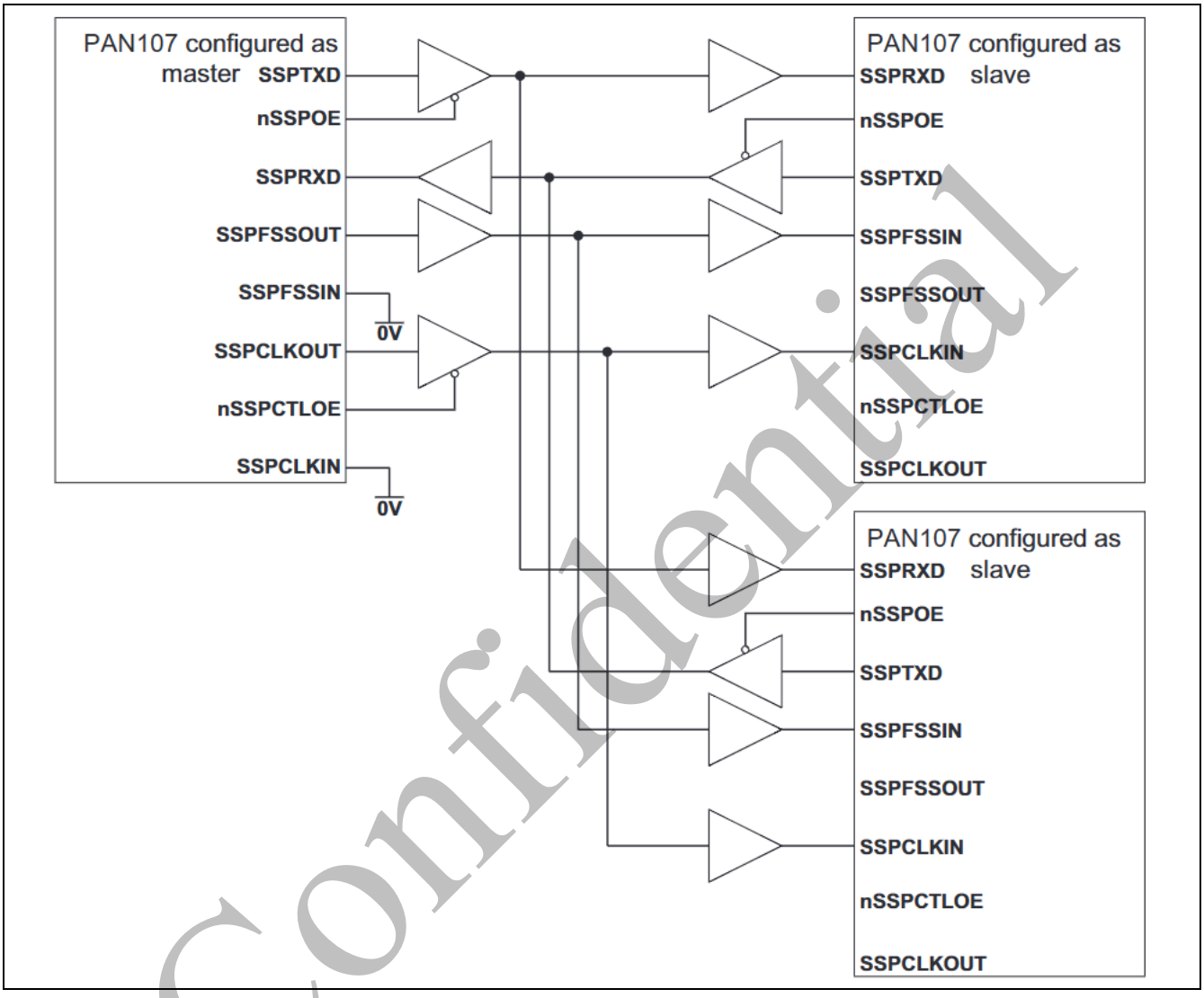


Figure 3-101 SSP master coupled to two slaves

Figure 3-101 shows the SSP instanced three times, as a single master and two slaves. The master can broadcast to the two slaves through the master SSPTXD line. In recpolnse, only one slave drives its nScpolE signal HIGH, thereby enabling its SSPTXD data onto the SSPRXD line of the master.

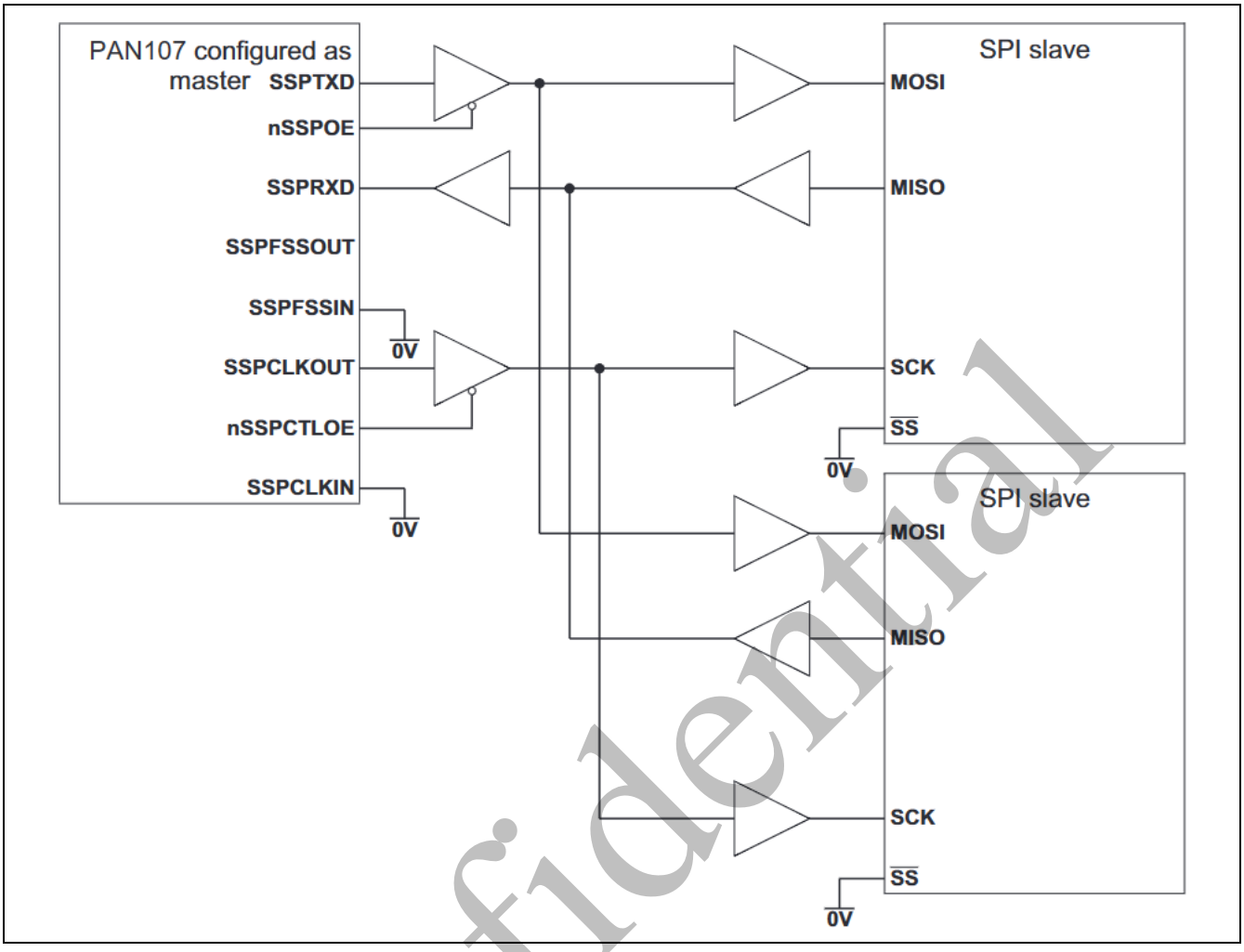


Figure 3-102 SSP master coupled to two slaves

Figure 3-102 shows how an SSP, configured as master, interfaces to two Motorola SPI slaves. Each SPI Slave Select (SS) signal is permanently tied LOW and configures them as slaves. Similar to the above operation, the master can broadcast to the two slaves through the master SSP SSPTXD line. In recpolnse, only one slave drives its SPI MISO port onto the SSPRXD line of the master.

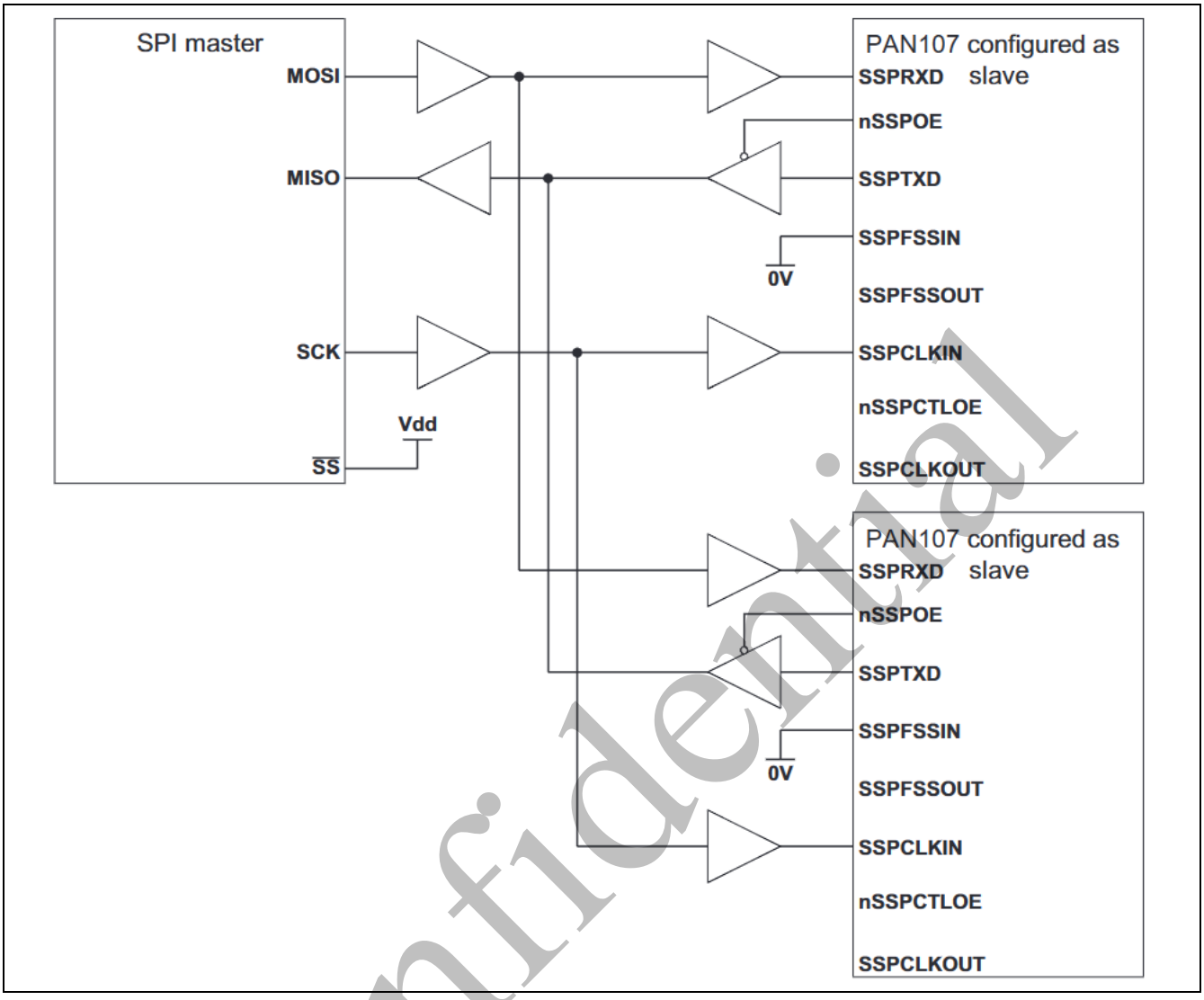


Figure 3-103 SPI master coupled to two SSP slaves

Figure 3-103 shows a Motorola SPI configured as a master and interfaced to two instances of SSP configured as slaves. In this case the slave Select Signal (SS) is permanently tied HIGH and configures it as a master. The master can broadcast to the two slaves through the master SPI MOSI line and in recpolnse, only one slave drives its nScpolE signal LOW. This enables its SSPTXD data onto the MISO line of the master.

3.15.6.18 DMA interface

The SSP provides an interface to connect to the DMA controller. The DMA operation of the SSP is controlled through the SSP DMA control register(DMA_CTL). The DMA interface includes the following signals, for receive:

SSPRXDMASREQ

Single-character DMA transfer request, asserted by the SSP. This signal is asserted when the

receive FIFO contains at least one character.

SSPRXDMABREQ

Burst DMA transfer request, asserted by the SSP. This signal is asserted when the receive FIFO contains four or more characters.

SSPRXDMACLR

DMA request clear, asserted by the DMA controller to clear the receive request signals. If DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data in the burst.

The DMA interface includes the following signals, for transmit:

SSPTXDMASREQ

Single-character DMA transfer request, asserted by the SSP. This signal is asserted when there is at least one empty location in the transmit FIFO.

SSPTXDMABREQ

Burst DMA transfer request, asserted by the SSP. This signal is asserted when the transmit FIFO contains four or less characters.

SSPTXDMACLR

DMA request clear, asserted by the DMA controller to clear the transmit request signals. If DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data in the burst.

The burst transfer and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there is more data than the watermark level of four in the receive FIFO, the burst transfer request and the single transfer request are asserted. When the amount of data left in the receive FIFO is less than the watermark level, the single request only is asserted. This is useful for situations where the number of characters left to be received in the stream is less than a burst.

For example, say 19 characters have to be received. The DMA controller then transfers four bursts of four characters and three single transfers to complete the stream.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is deasserted, a request signal can become active again, depending on the conditions declk_rateibed above. All request signals are deasserted if the SSP is disabled or the DMA enable signal is cleared.

Table 3-17 shows the trigger points for DMABREQ, for both the transmit and receive FIFOs.

Table 3-17 DMA trigger points for the transmit and receive FIFOs

Watermark level	Burst length	
	Transmit (number of empty locations)	Receive (number of filled locations)
1/2	4	4

Figure 3-104 shows the timing diagram for both a single transfer request and a burst transfer request with the appropriate DMA clear signal. The signals are all synchronous to PCLK.

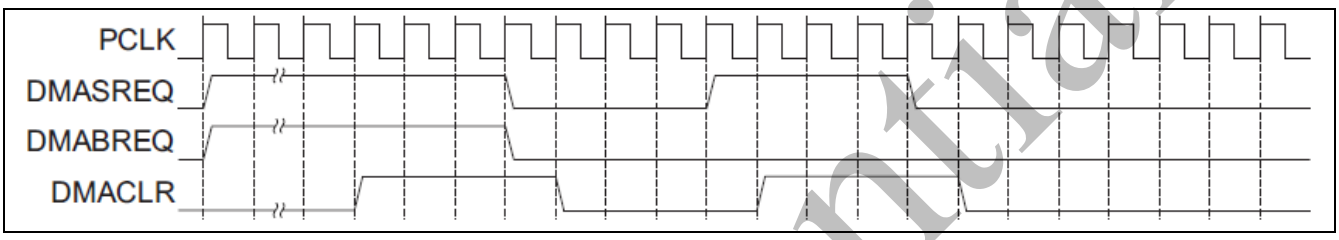


Figure 3-104 DMA transfer waveforms

3.15.6.19 Interrupts

There are five interrupts generated by SSP. The specific interrupt signals are declk_rateibed in Table 2. Four of these are individual, maskable, active-HIGH interrupts as follows:

- RX_INT
- TX_INT
- RX_OV_INT
- RX_TM_INT
- SSP_COM_INT

You can mask each of the four individual maskable interrupts by setting the appropriate bits in the INT_MASK register. Setting the appropriate mask bit HIGH enables the interrupt.

Provision of the individual outputs in addition to a combined interrupt output, enables the use of either a global interrupt service routine, or modular device drivers to handle interrupts.

The transmit and receive dynamic dataflow interrupts TX_INT and RX_INT have been separated from the status interrupts, so that data can be read or written in recpolnse to only the FIFO trigger levels.

The status of the individual interrupt sources can be read from SSPRIS and INT_STUS registers.

Table 3-18 Interrupt Signals

Port Name	Declk_rateiption	Asserted condition	Notes
RX_INT	SPI receive FIFO service interrupt request	≥ 4 entries in RX FIFO	-
TX_INT	SPI transmit FIFO service interrupt request	≤ 4 entries in TX FIFO	This enables operation in either of the following ways: Data can be written to the transmit FIFO prior to enabling the PrimeCell SSP and the interrupts The PrimeCell SSP and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.
RX_OV_INT	SPI receive overrun interrupt request	RX FIFO is already full and an additional data frame is received	Data is over-written in the receive shift register, but not the FIFO.
RX_TM_INT	SPI time out interrupt request	RX FIFO is not empty and the SPI has remained idle for a fixed 32 bit period	This interrupt is deasserted if the receive FIFO becomes empty by subsequent reads, or if new data is received on SSPRXD. It can also be cleared by writing to the RTIC bit in the INT_CLEAR register.
SSP_COM_INT	combined single interrupt	Any of the four individual interrupts above are asserted and enabled	An OR function of the individual masked sources

3.15.7 SSP Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Desrciption	Reset Value
SSP Base Address: SSP0_BA =0x4000_1000 SSP1_BA =0x4001_1000				
CTL_REG_0	SSPx_BA+0x00	R/W	Control Register 0	0x0000_0000
CTL_REG_1	SSPx_BA+0x04	R/W	Control Register 1	0x0000_0080
W_R_DATA	SSPx_BA+0x08	R/W	Receive FIFO(read) and Transmit FIFO Data Register	Configurable
STATUS_REG	SSPx_BA+0x0C	R	Status Register	0x0000_0003
CLK_PSC	SSPx_BA+0x10	R/W	Clock Prescale Register	0x0000_0000
INT_MASK	SSPx_BA+0x14	R/W	Interrupt Mask Set and Clear Register	0x0000_0000
INT_STUS_RAW	SSPx_BA+0x18	R	Raw Interrupt Status Register	0x0000_0008
INT_STUS	SSPx_BA+0x1C	R	Masked Interrupt Status Register	0x0000_0000
INT_CLEAR	SSPx_BA+0x20	W	Interrupt Clear Register	0x0000_0000
DMA_CTL	SSPx_BA+0x24	R/W	DMA Control Register	0x0000_0000

3.15.8 SSP Register Description

3.15.8.1 Control Register 0 (CTL_REG_0)

Register	Offset	R/W	Desrciption	Reset Value
CTL_REG_0	SSPx_BA+0x00	R/W	Control Register 0	0x0000_0000

Bits	Description	
[31:16]	reserved	reserved
[15:8]	CLK_RATE	Serial Clock Rate. The value clk_psc is used to generate the transmit and receive bit rate of the SSP. The bit rate is: $\frac{f_{ssp\_apb\_clk}}{clk\_psc \times (1 + clk\_rate)}$ where clk_psc is an even value from 2-254, programmed through the CLK_PSC register and clk_rate is a value from 0-255
[7]	CPHA	Serial Clock Phase. For Motorola SPI frame format only 1'b0: Data are captured on the first edge of the serial clock. Slave Select pulse high between each spi bus data item. 1'b1: Data are captured on the second edge of the serial clock. Slave Select keep low for continuous transmission. Note:only in 3-wire SPI mode, cpha =0 (cannot be modified unless the slave supports this mode)
[6]	CPOL	Serial Clock Polarity. For Motorola SPI frame format only 1'b0: Inactive state of serial clock is low 1'b1: Inactive state of serial clock is high Note:only in 3-wire SPI mode, cpol =0 (cannot be modified unless the slave supports this mode)
[5:4]	FAE_FOMT	Frame format: 2'b00: Motorola SPI frame format 2'b01: TI synchronous serial frame format 2'b10: National Microwire frame format 2'b11: Reserved, undefined operation. Note:only in 3-wire SPI mode, FRF=[0, 0](cannot be modified unless the slave supports this mode)
[3:0]	DATA_SIZE	Data Size Select: 4'b0000 ~ 4'b0010: Reserved. 4'b0011: 4-bit data. 4'b0100: 5-bit data. 4'b0101: 6-bit data. 4'b0110: 7-bit data. 4'b0111: 8-bit data. 4'b1000: 9-bit data. 4'b1001: 10-bit data. 4'b1010: 11-bit data. 4'b1011: 12-bit data.

		4’b1100: 13-bit data. 4’b1101: 14-bit data. 4’b1110: 15-bit data. 4’b1111: 16-bit data. Note:only in 3-wire mode, the default is 8 bit(cannot be modified unless the slave supports this mode)

3.15.8.2 Control Register 1 (CTL_REG_1)

Register	Offset	R/W	Desrciption	Reset Value
CTL_REG_1	SSPx_BA+0x04	R/W	Control Register 1	0x0000_0080

Bits	Desrciption	
[15:11]	Reserved	Reserved
[10]	SPI_MODE	determine the transmission mode of SPI 1'b0: 4-wire SPI (default) 1'b1: 3-wire SPI
[9]	DC_ENABLE	only used in 3-wire SPI mode, distinguish whether there is a DC output line 1'b0: no DC output line (default) 1'b1: exist a DC output line
[8]	DC_TYPE	only used in 3-wire SPI mode, control signal type of C/D 1'b0: Command (default) 1'b1: Data
[7]	WR_TYPE	only used in 3-wire SPI mode, read or write control signals 1'b0: read 1'b1: write (default)
[6]	Reserved	Reserved
[5]	RX_LSB	control receive order of receive fifo data item, compatible with data_size of CTL_REG_0 1'b0: read data item of receive fifo in an MSB to LSB order 1'b1: read data item of receive fifo in an LSB to MSB order For national microwire frame format, ssp slave receive fifo data item is always 8 bit, data width of ssp master receive fifo data item is set by data_size of CTL_REG_0. Note:only in 3-wire SPI mode, rx_lsb=0(cannot be modified unless the slave supports this mode)
[4]	TX_LSB	control transmit order of transmit fifo data item, compatible with data_size of CTL_REG_0 1'b0: transmit data item of transmit fifo in an MSB to LSB order 1'b1: transmit data item of transmit fifo in an LSB to MSB order For national microwire frame format, ssp master transmit fifo data item is always 8 bit, data width of ssp slave transmit fifo data item is set by data_size of CTL_REG_0. Note:only in 3-wire SPI mode, tx_lsb=0(cannot be modified unless the slave supports this mode)
[3]	SL_OUT_DIS	Slave-mode Output Disable Bit. This bit is relevant only in the slave mode, MS=1. In multiple-slave systems, it is possible for a SPI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto its serial output line. In such systems the RXD lines from multiple slaves could be tied together.To operate in such systems, the sl_out_dis bit can be set if the SPI slave is not supposed to drive the SSPTXD line: 0 = SPI can drive the SSPTXD output in slave mode. 1 = SPI must not drive the SSPTXD output in slave mode.
[2]	MA_SL_SEL	Master or Slave Mode Select. This bit can be modified only when the SPI is disabled, SSE=0:

		0 = Device configured as master, default. 1 = Device configured as slave. Note:only in 3-wire SPI mode, MS=0(cannot be modified)
[1]	SPI_ENABLE	Synchronous Serial Port Enable: 0 = SPI operation disabled. 1 = SPI operation enabled
[0]	LOOPBACK	Loop Back Mode: 0 = Normal serial port operation enabled. 1 = Output of transmit serial shifter is connected to input of receive serial shifter internally.

3.15.8.3 Data Register (W_R_DATA)

Register	Offset	R/W	Desrciption	Reset Value
W_R_DATA	SSPx_BA+0x08	R/W	Receive FIFO(read) and Transmit FIFO Data Register	0x0000_0000

Bits	Desrciption	
[15:0]	W_R_DATA	Transmit/Receive FIFO: Read = Receive FIFO. Write = Transmit FIFO. You must right-justify data when the SSP is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by transmit logic. The receive logic automatically right-justifies.

3.15.8.4 Status Register (STATUS_REG)

Register	Offset	R/W	Desrciption	Reset Value
STATUS_REG	SSPx_BA+0x0C	R	Status Register	0x0000_0003

Bits	Desrciption	
[15:5]	Reserved	Reserved
[4]	BUSY	SPI Busy Flag: 0 = SPI is idle. 1 = SPI is currently transmitting and/or receiving a frame or the transmit FIFO is not empty
[3]	RX_FULL	Receive FIFO Full: 0 = Receive FIFO is not full. 1 = Receive FIFO is full.
[2]	RX_NOT_EMPTY	Receive FIFO Not Empty: 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty
[1]	TX_NOT_FULL	Transmit FIFO Not Full 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full.
[0]	TX_EMPTY	Transmit FIFO Empty 0 = Transmit FIFO is not empty. 1 = Transmit FIFO is empty

3.15.8.5 Clock Prescale Register (CLK_PSC)

Register	Offset	R/W	Desrciption	Reset Value
CLK_PSC	SSPx_BA+0x10	R/W	Clock Prescale Register	0x0000_0000

Bits	Desrciption	
[15:8]	Reserved	Reserved
[7:0]	CLK_PSC	Clock Prescale Divisor. Must be an even number from 2-254, depending on the frequency of SSPCLK. The least significant bit always returns zero on reads.

3.15.8.6 Interrupt Mask Set and Clear Register (INT_MASK)

In this register, on a read it gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the correcpolnding mask.

Register	Offset	R/W	Desrciption	Reset Value
INT_MASK	SSPx_BA+0x14	R/W	Interrupt Mask Set and Clear Register	0x0000_0000

Bits	Desrciption	
[15:4]	Reserved	Reserved
[3]	TX_INT_MSK	Transmit FIFO interrupt mask: 0 = Transmit FIFO half empty or less condition interrupt is masked. 1 = Transmit FIFO half empty or less condition interrupt is not masked.
[2]	RX_INT_MSK	Receive FIFO interrupt mask: 0 = Receive FIFO half full or less condition interrupt is masked. 1 = Receive FIFO half full or less condition interrupt is not masked.
[1]	RX_TMINT_MSK	Receive timeout interrupt mask: 0 = Receive FIFO not empty and no read prior to timeout period interrupt is masked. 1 = Receive FIFO not empty and no read prior to timeout period interrupt is not masked.
[0]	RX_OVINT_MSK	Receive overrun interrupt mask: 0 = Receive FIFO written to while full condition interrupt is masked. 1 = Receive FIFO written to while full condition interrupt is not masked.

3.15.8.7 Raw Interrupt Status Register (INT_STUS_RAW)

Register	Offset	R/W	Desrciption	Reset Value
INT_STUS_RAW	SSPx_BA+0x18	R/W	Raw Interrupt Status Register	0x0000_0008

Bits	Desrciption	
[15:4]	Reserved	Reserved
[3]	TX_INT_RAW	Gives the raw interrupt state, prior to masking, of the *TX_INT* interrupt
[2]	RX_INT_RAW	Gives the raw interrupt state, prior to masking, of the *RX_INT* interrupt
[1]	RX_TMINT_RAW	Gives the raw interrupt state, prior to masking, of the *RX_TM_INT* interrupt
[0]	RX_OVINT_RAW	Gives the raw interrupt state, prior to masking, of the *RX_OV_INT* interrupt

3.15.8.8 Masked Interrupt Status Register (INT_STUS)

Register	Offset	R/W	Desrciption	Reset Value
INT_STUS	SSPx_BA+0x1C	R	Masked Interrupt Status Register	0x0000_0000

Bits	Desrciption	
[15:4]	Reserved	Reserved
[3]	TX_INT_FLAG	Gives the transmit FIFO masked interrupt state, after masking, of the *TX_INT* interrupt
[2]	RX_INT_FLAG	Gives the receive FIFO masked interrupt state, after masking, of the *RX_INT* interrupt
[1]	RX_TMINT_FLAG	Gives the receive timeout masked interrupt state, after masking, of the *RX_TM_INT* interrupt
[0]	RX_OVINT_FLAG	Gives the receive over run masked interrupt status, after masking, of the *RX_OV_INT* interrupt

3.15.8.9 Interrupt Clear Register (INT_CLEAR)

Register	Offset	R/W	Desrciption	Reset Value
INT_CLEAR	SSPx_BA+0x20	R	Interrupt Clear Register	0x0000_0000

Bits	Desrciption	
[15:2]	Reserved	Reserved
[1]	CL_RX_TMINT_FLG	Clears the *RX_TM_INT* interrupt
[0]	CL_RX_OVINT_FLG	Clears the *RX_OV_INT* interrupt

3.15.8.10 DMA Control Register (DMA_CTL)

Register	Offset	R/W	Desrciption	Reset Value
DMA_CTL	SSPx_BA+0x24	R	DMA Control Register	0x0000_0000

Bits	Desrciption	
[15:2]	Reserved	Reserved
[1]	DMA_TX_EN	Transmit DMA Enable. 0 = DMA for the transmit FIFO is disabled. 1 = DMA for the transmit FIFO is enabled.
[0]	DMA_RX_EN	Receive DMA Enable. 0 = DMA for the receive FIFO is disabled. 1 = DMA for the receive FIFO is enabled.

3.16 UART Controller (UART)

3.16.1 Overview

The UART is a programmable Universal Asynchronous Receiver/Transmitter (UART). It converts parallel input signals into serial output signals.There is no CLK line. It transfers data by TX and RX line. UART achieves data identification by start bit, stop bit and baud rate

3.16.2 Features

- Support 2 uart device
- 9-bit serial data support
- Programmable fractional baud rate support
- Support DMA
- Auto Flow Control mode, as specified in the 16750 standard
- Loop back mode that enables greater testing of Modem Control and Auto Flow Control features
- FIFO support, the depths of Transmit and receive FIFO is 16
- Programmable serial data baud rate as calculated by the following:

  *baud rate = (serial clock frequency)/(16*divisor)*
- Configurable stop bits：support for 1 , 1.5 or 2 stop bits

3.16.3 Block Diagram

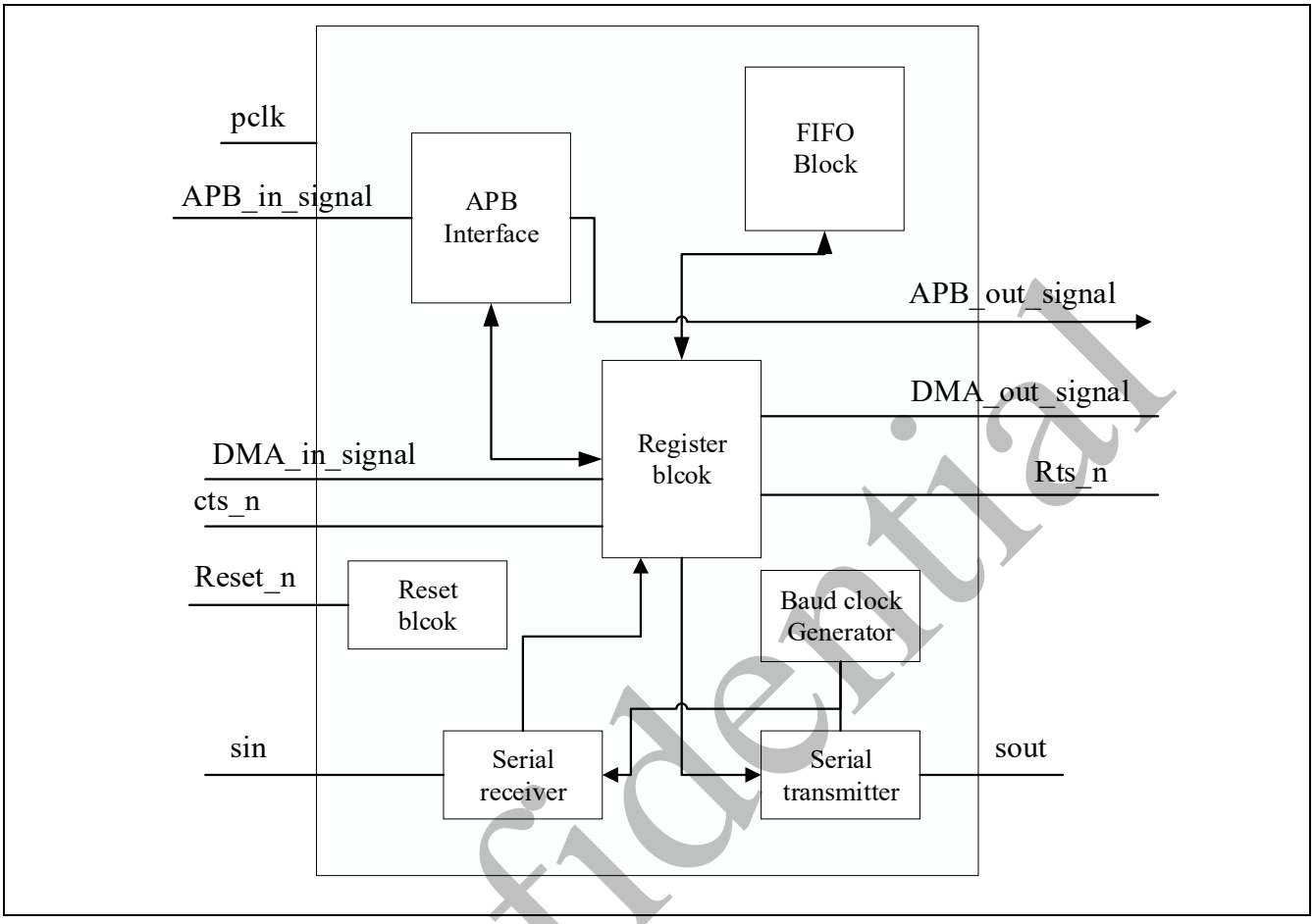


Figure 3-105 UART Controller Block Diagram

3.16.4 Functional Description

3.16.4.1 UART (RS232) Serial Protocol

Because the serial communication between the UART and a selected device is asynchronous, additional bits (start and stop) are added to the serial data to indicate the beginning and end. Utilizing these bits allows two devices to be synchronized. This structure of serial data accompanied by start and stop bits is referred to as a character, as shown in Figure 3-106.

An additional parity bit can be added to the serial character. This bit appears after the last data bit and before the stop bit(s) in the character structure in order to provide the UART with the ability to perform simple error checking on the received data.

All the bits in the transmission are transmitted for exactly the same time duration; the exception to this is the half-stop bit when 1.5 stop bits are used. This duration is referred to as a Bit Period or Bit Time; one Bit Time equals sixteen baud clocks.

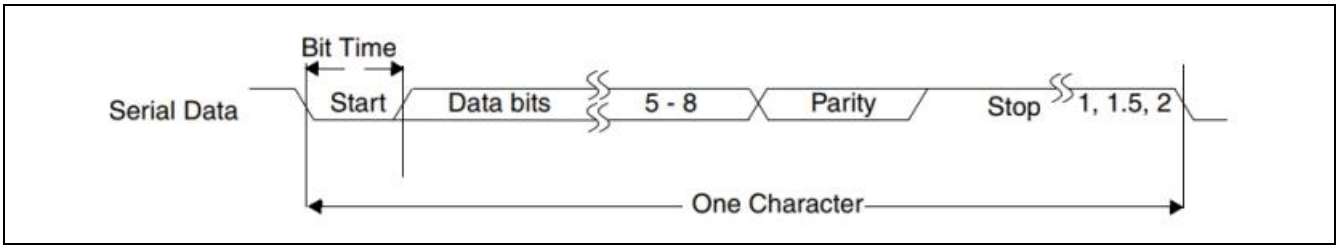


Figure 3-106 Serial Data Format

To ensure stability on the line, the receiver samples the serial input data at approximately the midpoint of the Bit Time once the start bit has been detected. Because the exact number of baud clocks is known for which each bit was transmitted, calculating the midpoint for sampling is not difficult; that is, every sixteen baud clocks after the midpoint sample of the start bit. Together with serial input debouncing, this sampling helps to avoid the detection of false start bits. Short glitches are filtered out by debouncing, and no transition is detected on the line. If a glitch is wide enough to avoid filtering by debouncing, a falling edge is detected. However, a start bit is detected only if the line is again sampled low after half a bit time has elapsed. Figure 3-107 shows the sampling points of the first two bits in a serial character.

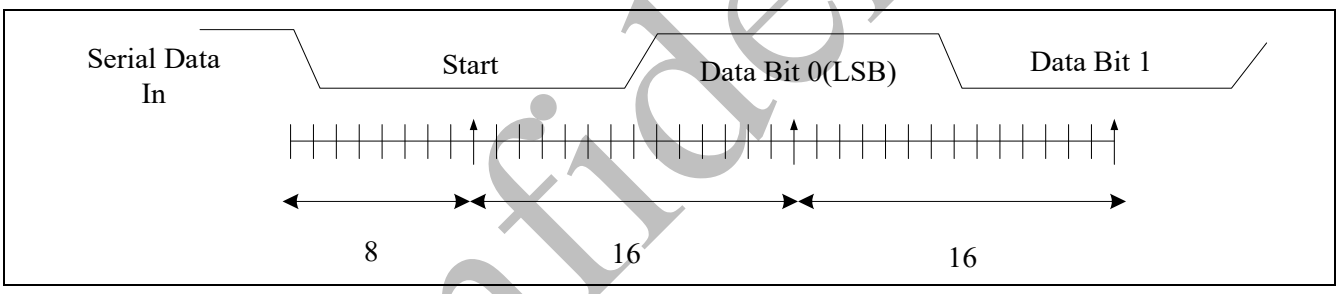


Figure 3-107 Receiver Serial Data Sample Points

3.16.4.2 UART 9-bit Data Transfer

The UART can be configured to have 9-bit data transfer in both transmit and receive mode. The 9th bit in the character appears after the 8th bit and before the parity bit in the character. By enabling 9-bit data transfer mode, UART can be used in multi-drop systems where one master is connected to multiple slaves in a system. The master communicates with one of the slaves. When the master wants to transfer a block of data to a slave, it first sends an address byte to identify the target slave. The address byte is differentiated from the data byte by setting the 9th bit of the data byte to 1. If the 9th bit is set to 0, then the character represents an address byte. All the slave systems compare the address byte with their own address and only the target slave (in which the address has matched) is enabled to receive data from the master. The master then starts transmitting data bytes to the target slave. The non-addressed slave systems ignore the incoming data until a new address byte is received.

3.16.4.3 UART Fractional Baud Rate Support

UART supports fractional baud rate that enables a user to program the fractional part of the divisor value to generate fractional baud rate that results in reduced frequency error. The UART interface usage has been evolving to include ever increasing baud rate speeds. The UART needs to be software configurable to handle the baud rates within 2% frequency error. The Baud rate of UART is controlled by PCLK and the Divisor Latch Register (DLH and DLL).

The baud rate is determined by the following factors:

- Serial clock operating frequency
- The desired baud rate.
- The baud rate generator divisor value, DIVISOR (composed of DLH & DLL registers).
- The acceptable Baud-rate error, %ERROR

The programmable fractional baud rate divisor enables a finer resolution of baud clock than the conventional integer divider. The programmable fractional baud clock divider allows for the programmability of both an integer divisor as well as fractional component. The average frequency of the baud clock from the fractional baud rate divisor is dependent upon both the integer divisor and the fractional component, thereby providing a finer resolution to the average frequency of the baud clock.

Baud Rate Divisor = Serial Clock Frequency / (16∗Required Baud Rate) $=BRD_I+BRD_F$ (1)

Where,

BRD_I - Integer part of the divisor.

BRD_F - Fractional part of the divisor.

Calculating the Fractional Value Error

Following is a sample for calculating the fractional value error.

Consider the following values:

- Required Baud Rate (RBR) = 115200
- Serial Clock (PCLK) = 64MHz
- DLF_SIZE = 4

Then, as per equation (1), Baud Rate Divisor (BRD) is as follows:

$$BRD = \frac{64M}{16\times 115200} = 34.72222 \quad (2)$$

In (2), the integer and fractional parts are as follows:

- Integer part (BRD_I) = 34
- Fractional part (BRD_F) = 0.72222

Therefore, Baud Rate Divisor Latch Fractional Value (DLF) is as follows:

$$DLF = BRD_F \times 2^{DLF\_SIZE} = 0.72222 \times 16 = 12(\text{roundoff value}) \quad (3)$$

The Generated Baud Rate Divider (GD) is as follows:

$$GD = BRD_I + \frac{DLF}{2^{DLF\_SIZE}} = 34 + \frac{12}{16} = 34.75 \quad (4)$$

Therefore, the Generated Baud Rate (GBR) is as follows:

$$GBR = \frac{64M}{16 \times 34.75} = 115107.914 \quad (5)$$

Now the error is calculated as follows:

$$Error = \frac{|RBR - GBR|}{RBR} = 0.000799 \quad (6)$$

The error percentage is as follows:

$$Error\% = 0.000799 \times 100 = 0.0799$$

3.16.4.4 FIFO support

Data can be written to the transmit FIFO as normal; However no serial transmission occurs in this mode normal operation halted and thus no data leave the FIFO. The data that has been written to the transmit FIFO can be read back with the Transmit FIFO Read (TFR) register, which when read gives the current data at the top of the transmit FIFO.

Similarly, data can be read from the receive FIFO as normal. Since the normal operation of the UART is halted in this mode, data must be written to the receive FIFO so the data can be read back. Data is written to the receive FIFO using the Receive FIFO Write (RFW) register. The upper two bits of the 10-bit register are used to write framing error and parity error detection information to the receive FIFO, as follows:

- RFW[9] indicates framing error
- RFW[8] indicates parity error

Although these bits cannot be read back through the Receive Buffer Register, they can be checked by reading the Line Status Register and checking the corresponding bits when the data in question is at the top of the receive FIFO.

3.16.4.5 UART Interrupts

Assertion of the UART interrupt output signal (intr)—a positive-level interrupt—occurs whenever one of the several prioritized interrupt types are enabled and active. When an interrupt occurs, the master accesses the IIR register.

The following interrupt types can be enabled with the IER register:

- Receiver Error
- Receiver Data Available
- Character Timeout (in FIFO mode only)
- Transmitter Holding Register Empty at/below threshold (in Programmable THR_EMPTY interrupt mode)
- Busy Detect Indication

These interrupt types are covered in more detail in Table 3-19.

Table 3-19 Interrupt Type

Interrupt ID	Priority Level	Interrupt Type
0001	-	None
0110	Highest	Receiver line status
0100	Second	Received data available
1100	Second	Character timeout indication
0010	Third	Transmit holding register empty
0000	Fourth	Modem status
0111	Fifth	Busy detect indication

3.16.4.6 Auto Flow Control

Figure 3-108 shows a block diagram of the Auto Flow Control functionality. When Auto SEND_REQ and Auto CLR_SEND are enabled, the rts_n output is forced inactive when the receiver FIFO level reaches the threshold set by FCR[7:6]. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space.

When Auto SEND_REQ is enabled, the rts_n output is forced inactive (high) when the receiver FIFO level reaches the threshold set by FCR[7:6]. Otherwise, the rts_n output is forced inactive (high) when the FIFO is almost full, where “almost full” refers to two available slots in the FIFO. When rts_n is connected to the cts_n input of another UART device, the other UART stops sending serial data until the receiver FIFO has available space; that is, until it is completely empty.

The selectable receiver FIFO threshold values are:

- 1
- ¼
- ½
- 2 less than full

Since one additional character can be transmitted to the UART after rts_n has become inactive—due to data already having entered the transmitter block in the other UART—setting the threshold to "2 less than full" allows maximum use of the FIFO with a safety zone of one character. Once the receiver FIFO becomes completely empty by reading the Receiver Buffer Register (RBR), rts_n again becomes active (low), signaling the other UART to continue sending data.

When Auto CLR_SEND is enabled (active), the UART transmitter is disabled whenever the cts_n input becomes inactive (high); this prevents overflowing the FIFO of the receiving UART. If the cts_n input is not inactivated before the middle of the last stop bit, another character is transmitted before the transmitter is disabled. While the transmitter is disabled, the transmitter FIFO can still be written to, and even overflowed.

Therefore, when using this mode, the following happens:

- UART status register can be read to check if transmit FIFO is full (USR[1] set to 0)
- Current FIFO level can be read using TFL register
- Programmable THR_EMPTY Interrupt mode must be enabled to access "FIFO full" status using Line Status Register (LSR)

When using the "FIFO full" status, software can poll this before each write to the Transmitter FIFO.

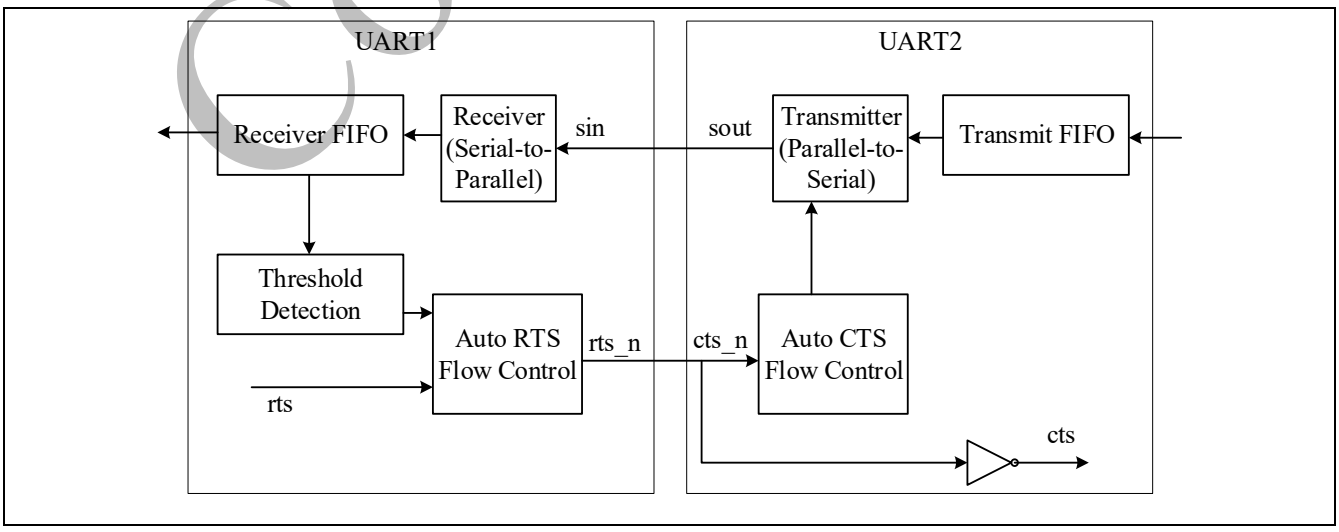


Figure 3-108 Auto Flow Control Block Diagram

Auto SEND_REQ and Auto CLR_SEND are described as follows:

- Auto SEND_REQ – Becomes active when the following occurs:
  - SEND_REQ (MCR[1] bit and MCR[5]bit are both set)
  - FIFOs are enabled (FCR[0]) bit is set)
- Auto CLR_SEND – becomes active when the following occurs:
  - AFC_EN (MCR[5] bit = 1)
  - FIFOs are enabled through FIFO Control Register FCR[0] bit

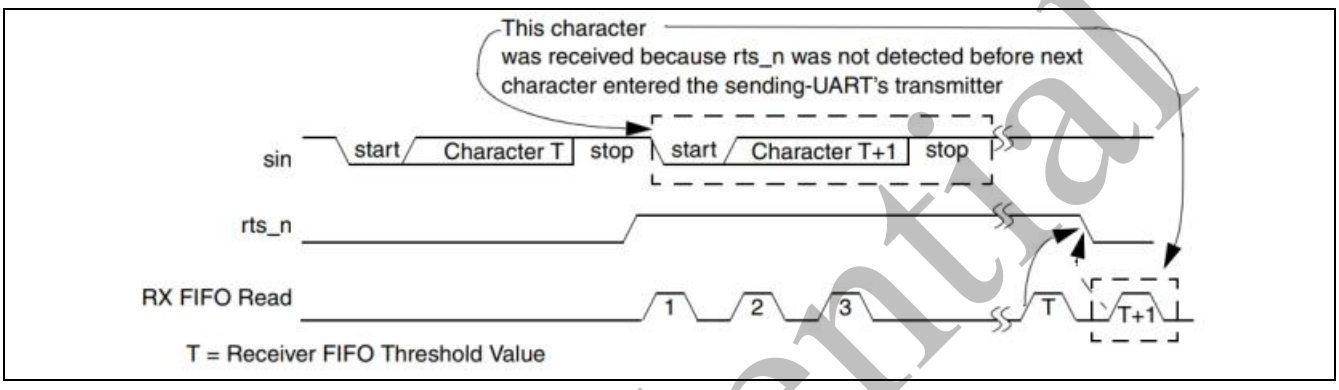


Figure 3-109 Auto SEND_REQ Timing

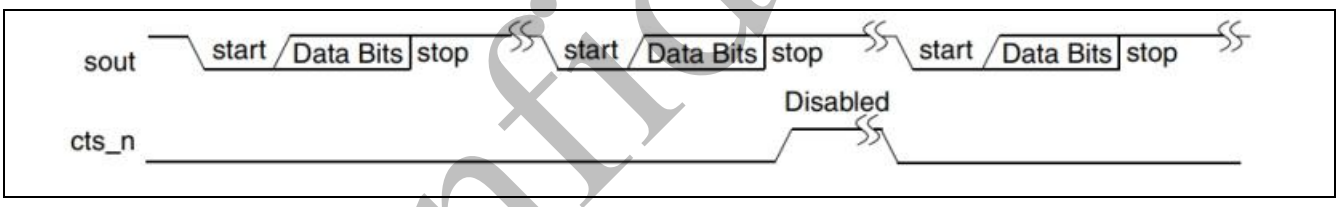


Figure 3-110 Auto CLR_SEND Timing

3.16.4.7 Programmable THR_EMPTY Interrupt

When Programmable THR_EMPTY Interrupt mode is selected, it can be enabled using the Interrupt Enable Register (IER[7]).When FIFOs and THR_EMPTY mode are enabled, the THR_EMPTY Interrupts and dma_tx_req_n are active at, and below, a programmed transmitter FIFO empty threshold level, as opposed to empty, as shown in the flowchart in Figure 3-111.

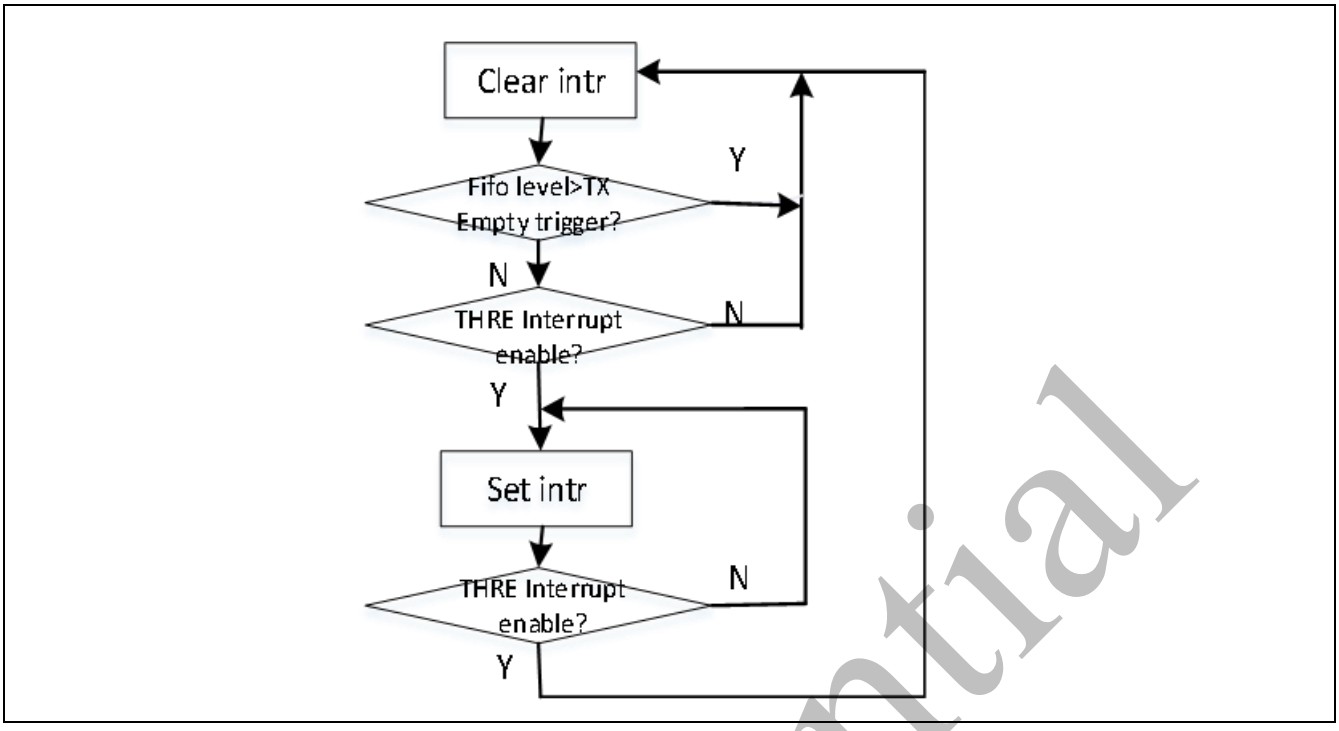


Figure 3-111 Flowchart of Interrupt Generation for Programmable THR_EMPTY Interrupt Mode

The threshold level is programmed into FCR[5:4]. Available empty thresholds are:

- Empty
- 2
- 1/4
- 1/2

Selection of the best threshold value depends on the system's ability to begin a new transmission sequence in a timely manner. However, one of these thresholds should be optimal for increasing system performance by preventing the transmitter FIFO from running empty. For threshold setting details, refer to FCR.

In addition to the interrupt change, the Line Status Register (LSR[5]) also switches from indicating that the transmitter FIFO is empty to the FIFO being full. This allows software to fill the FIFO for each transmit sequence by polling LSR[5] before writing another character. The flow then allows the transmitter FIFO to be filled whenever an interrupt occurs and there is data to transmit, rather than waiting until the FIFO is completely empty. Waiting until the FIFO is empty causes a reduction in performance whenever the system is too busy to respond immediately. Further system efficiency is achieved when this mode is enabled in combination with Auto Flow Control.

3.16.4.8 DMA Support

DMA Interface Signal

The UART supports DMA signaling with use of the signal described in Table 3-20.

Table 3-20 UART DMA Signal Description

Port Name	I/O	Description
dma_tx_ack_n	I	DMA Transmit Acknowledge (Active High)
dma_rx_ack_n	I	DMA Receive Acknowledge (Active High)
dma_tx_req_n	O	Transmit Buffer Ready (Active High)
dma_tx_single_n	O	DMA Transmit FIFO Single (Active High)
dma_rx_req_n	O	Receive Buffer Ready (Active High)
dma_rx_single_n	O	DMA Receive FIFO Single (Active High)

Example DMA Flow

The extra handshaking signals are explained in the following DMA flow for a UART. As a block flow control device, the DMA Controller is programmed by the processor with the number of data items (block size) that are to be transmitted or received by the UART; this is programmed into the BLOCK_TS field of the CTLx register. The block is broken into a number of transactions, each initiated by a request from the UART. The DMA Controller must also be programmed with the number of data items (in this case, UART FIFO entries) to be transferred for each DMA request. This is also known as the burst transaction length, and is programmed into the SRC_MSIZE/DEST_MSIZE fields of the DMA CTLx register for source and destination, respectively.

Figure 3-112 shows a single block transfer, where the block size programmed into the DMA Controller is 12 and the burst transaction length is set to 4.

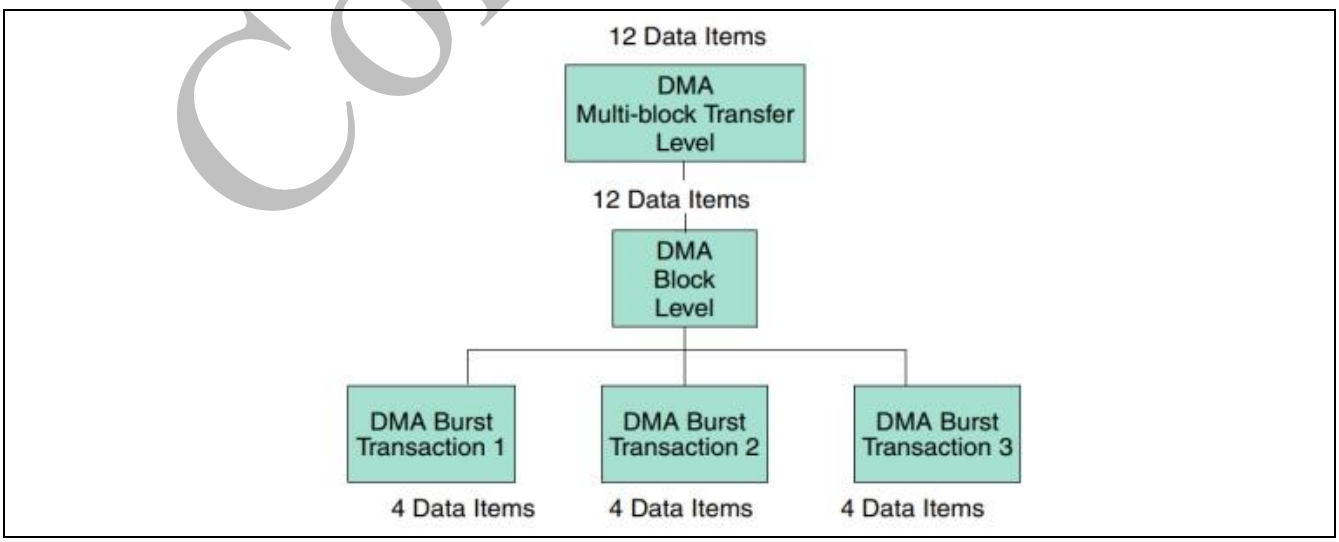


Figure 3-112 Breakdown of DMA Transfer into Burst Transactions

Block Size : DMA.CTLx.BLOCK_TS=12

Number of data items per source burst transaction : DMA.CTLx.SRC_MSIZE = 4

UART.FCR[7:6] = 01 = FIFO 1/4 full = DMA.CTLx.SRC_MSIZE

In this case, the block size is a multiple of the burst transaction length. Therefore, the DMA block transfer consists of a series of burst transactions. If the UART makes a transmit request to this channel, four data items are written to the UART transmit FIFO. Similarly, if the UART makes a receive request to this channel, four data items are read from the UART receive FIFO. Three separate requests must be made to this DMA channel before all twelve data items are written or read.

When the block size programmed into the DMA Controller is not a multiple of the burst transaction length, as shown in Figure 3-113, a series of burst transactions followed by single transactions are needed to complete the block transfer.

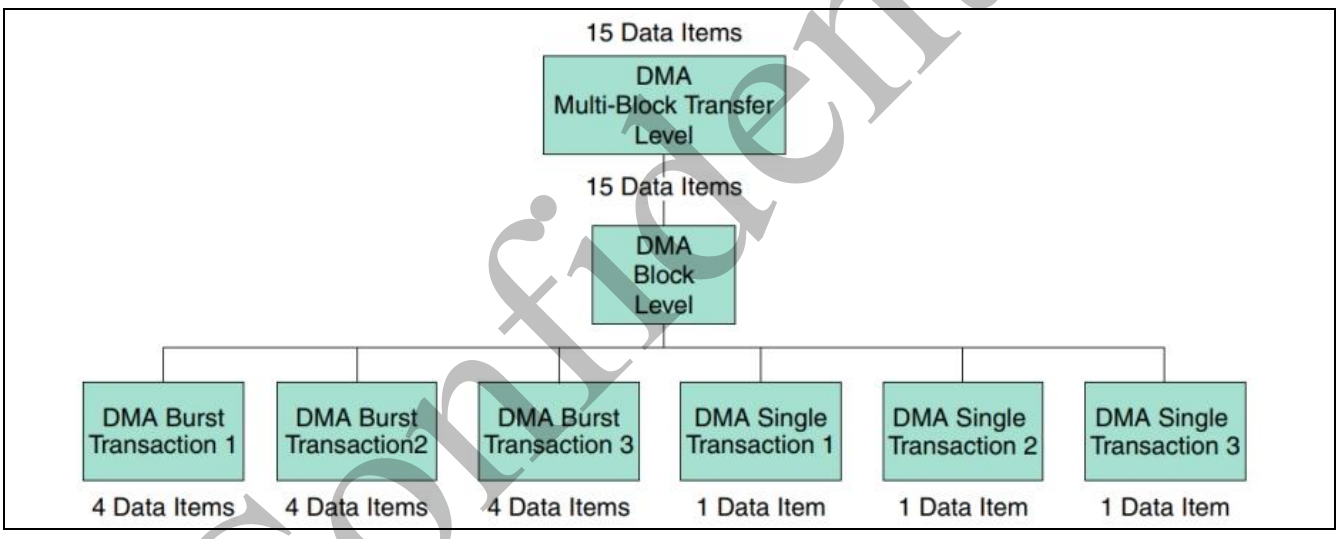


Figure 3-113 Breakdown of DMA Transfer into Single and Burst Transactions

Block Size : DMA.CTLx.BLOCK_TS=15

Number of data items per burst transaction : DMA.CTLx.DEST_MSIZE = 4

UART.FCR[5:4] = 10 =FIFO 1/4 full = 4 = DMA.CTLx.DEST_MSIZE

Transmit Watermark Level and Transmit FIFO Underflow

During UART serial transfers, transmit FIFO requests are made to the DMA whenever the number of entries in the transmit FIFO is less than or equal to the decoded level of the Transmit Empty Trigger (TXE_TRIG) of the FCR register (bits 5:4); this is known as the watermark level. The DMA responds by writing a burst of data to the transmit FIFO buffer, of length CTLx.DEST_MSIZE.

Data should be fetched from the DMA often enough for the transmit FIFO to perform serial transfers continuously; that is, when the FIFO begins to empty, another DMA request should be triggered. Otherwise the FIFO runs out of data (underflow). To prevent this condition, you must set the watermark level correctly.

Choosing Transmit Watermark Level

Consider the example where the following assumption is made:

DMA.CTLx.DEST_MSIZE = FIFO_DEPTH - UART.FCR[5:4]

The number of data items to be transferred in a DMA burst is equal to the empty space in the Transmit FIFO. Consider two different watermark level settings.

Case 1: FCR[5:4] = 01 — decodes to 2

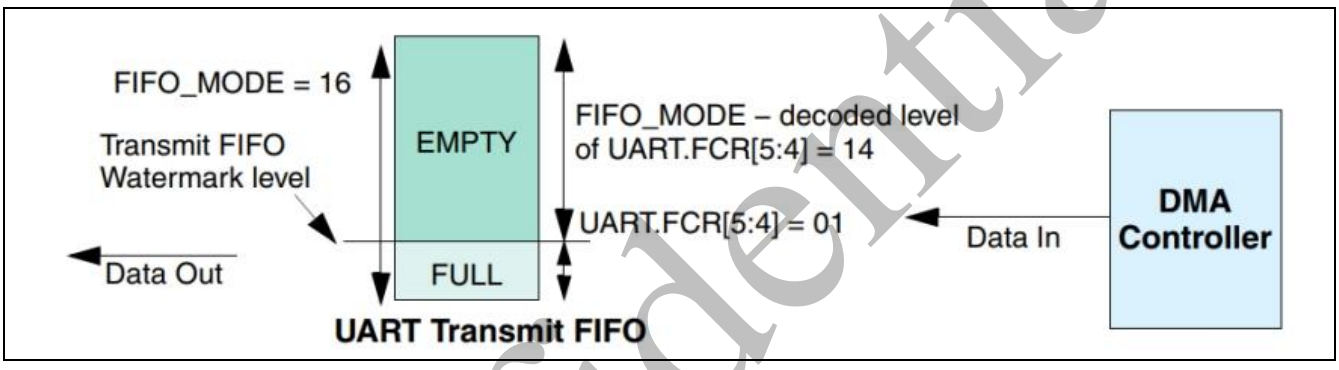


Figure 3-114 Case 1 Watermark Levels

- Transmit FIFO watermark level = decoded level of UART.FCR[5:4] = 2
- DMA.CTLx.DEST_MSIZE = FIFO_MODE - UART.FCR[5:4] = 14
- UART transmit FIFO_MODE = 16
- DMA.CTLx.BLOCK_TS = 56

Therefore, the number of burst transactions needed equals the block size divided by the number of data items per burst:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 56/14 = 4

The number of burst transactions in the DMA block transfer is 4., but the watermark level—decoded level of UART.FCR[5:4] is quite low. Therefore, the probability of a UART underflow is high where the UART serial transmit line needs to transmit data, but where there is no data left in the transmit FIFO. This occurs because the DMA has not had time to service the DMA request before the transmit FIFO becomes empty.

Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8)

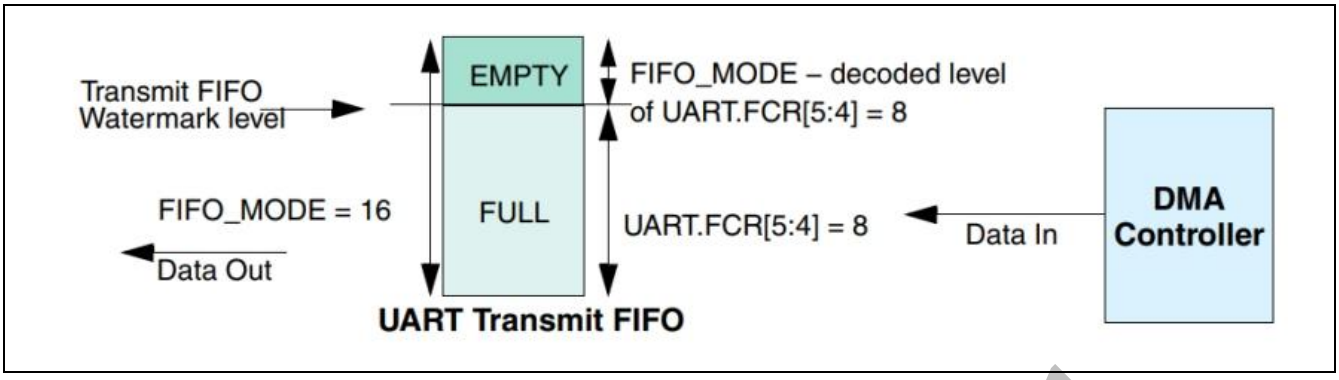


Figure 3-115 Case 2 Watermark Levels

- Transmit FIFO watermark level = decoded level of UART.FCR[5:4] = 8
- DMA.CTLx.DEST_MSIZE = FIFO_MODE - UART.FCR[5:4] = 8
- UART transmit FIFO_MODE = 16
- DMA.CTLx.BLOCK_TS = 56

Number of burst transactions in Block:

DMA.CTLx.BLOCK_TS/DMA.CTLx.DEST_MSIZE = 56/8 = 7

In this block transfer, there are seven destination burst transactions in a DMA block transfer, but the watermark level — decoded level of UART.FCR[5:4] — is high. Therefore, the probability of a UART underflow is low because the DMA controller has enough time to service the destination burst transaction request before the UART transmit FIFO becomes empty.

Thus, the second case has a lower probability of underflow at the expense of more burst transactions per block. This provides a potentially greater amount of AMBA bursts per block and worse bus utilization than Case 1.

Therefore, the goal in choosing a watermark level is to minimize the number of transactions per block, while at the same time keeping the probability of an underflow condition to an acceptable level. In practice, this is a function of the ratio of: rate of UART data transmission : rate of DMA response to destination burst requests.

For example, both of the following increases the rate at which the DMA controller can respond to burst transaction requests:

- Promoting channel to highest priority channel in DMA
- Promoting DMA master interface to highest priority master in AMBA layer

This in turn enables the user to decrease the watermark level, which improves bus utilization without compromising the probability of an underflow occurring.

Selecting DEST_MSIZE and Transmit FIFO Overflow

As can be seen from Figure 3-115, programming DMA.CTLx.DEST_MSIZE to a value greater than the watermark level that triggers the DMA request can cause overflow when there is not enough space in the UART transmit FIFO to service the destination burst request. Therefore, use the following in order to avoid overflow:

DMA.CTLx.DEST_MSIZE< = UART.FIFO_DEPTH - decoded level of UART.FCR[5:4](1)

In Case 2: FCR[5:4] = 11 — FIFO 1/2 full (decodes to 8), the amount of space in the transmit FIFO at the time the burst request is made is equal to the destination burst length, DMA.CTLx.DEST_MSIZE. Thus, the transmit FIFO can be full, but not overflowed, at the completion of the burst transaction. Therefore, for optimal operation, DMA.CTLx.DEST_MSIZE should be set at the FIFO level that triggers a transmit DMA request; that is:

DMA.CTLx.DEST_MSIZE = UART.FIFO_DEPTH - decoded level of UART.FCR[5:4](2)

Adhering to equation (2) reduces the number of DMA bursts needed for a block transfer, which in turn improves CPU bus utilization.

Receive Watermark Level and Receive FIFO Overflow

During UART serial transfers, receive FIFO requests are made to the DMA whenever the number of entries in the receive FIFO is at or above the decoded level of Receiver Trigger (RCVR_TRIG) of the FCR[7:6]. This is known as the watermark level. The DMA responds by fetching a burst of data from the receive FIFO buffer of length CTLx.SRC_MSIZE.

Data should be fetched by the DMA often enough for the receive FIFO to accept serial transfers continuously; that is, when the FIFO begins to fill, another DMA transfer is requested. Otherwise, the FIFO fills with data (overflow). To prevent this condition, you must correctly set the watermark level.

Choosing the Receive Watermark Level

Similar to choosing the transmit watermark level described earlier, the receive watermark level-decoded level of FCR[7:6]—should be set to minimize the probability of overflow. It is a trade-off between the number of DMA burst transactions required per block versus the probability of an overflow occurring.

Selecting SRC_MSIZE and Receive FIFO Underflow

As can be seen in Figure 3-116, programming a source burst transaction length greater than the watermark level can cause underflow when there is not enough data to service the source burst request. Therefore, equation (3) below must be adhered to in order to avoid underflow.

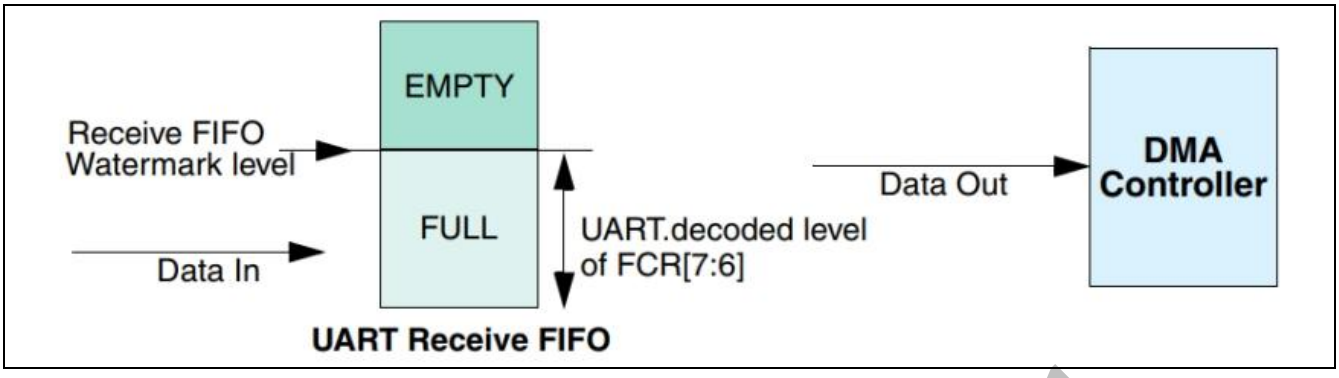


Figure 3-116 UART Receive FIFO

If the number of data items in the receive FIFO is equal to the source burst length at the time the burst request is made – DMA.CTLx.SRC_MSIZE – the receive FIFO can be emptied, but not under-flowed, at the completion of the burst transaction. For optimal operation, DMA.CTLx.SRC_MSIZE should be set at the watermark level; that is:

DMA.CTLx.SRC_MSIZE = decoded level of FCR[7:6] (3)

Adhering to equation (3) reduces the number of DMA bursts in a block transfer, and this in turn can improve CPU bus utilization.

3.16.5 Programing Example

3.16.5.1 Flowchart for UART Transmit Programming Example

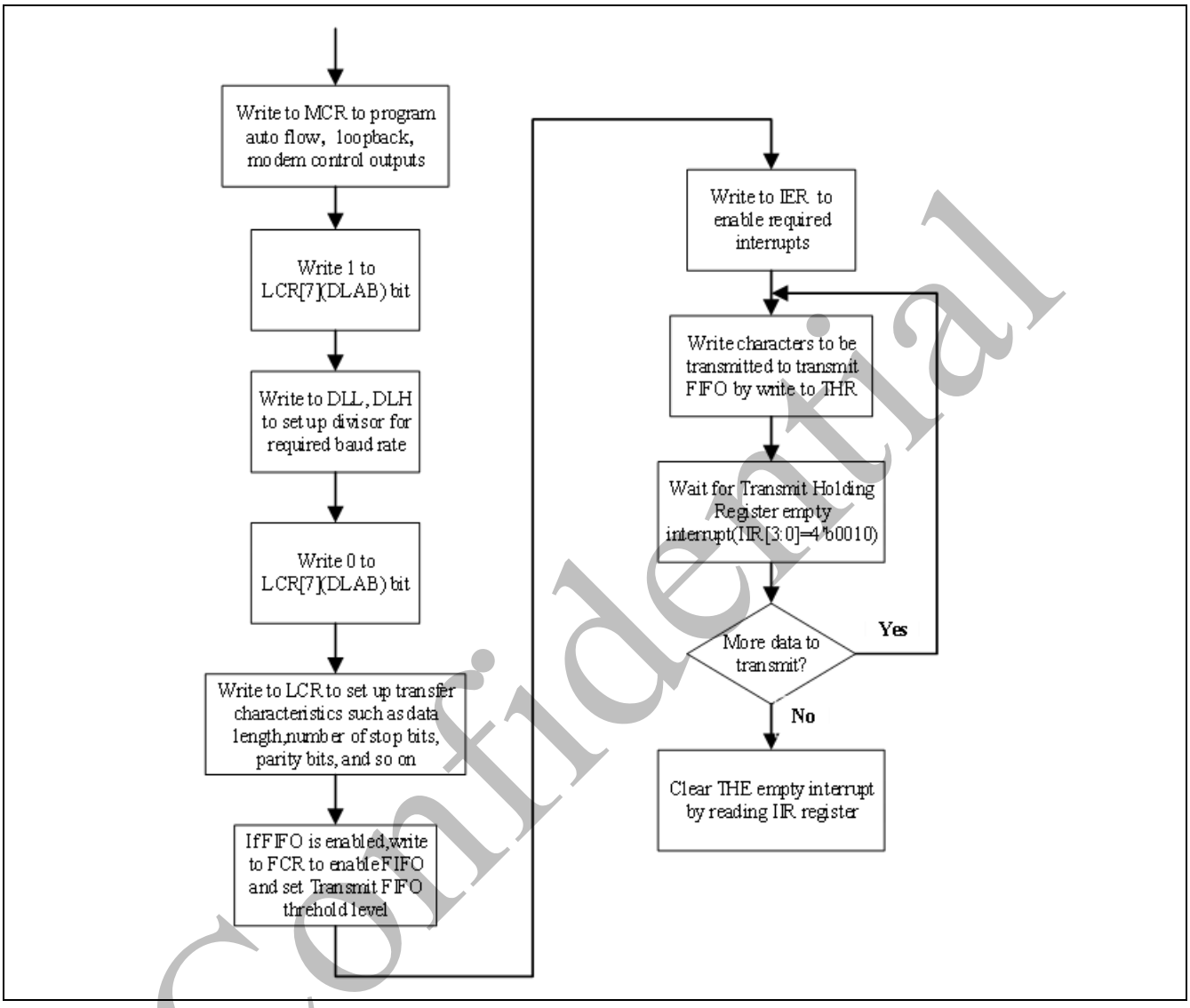


Figure 3-117 Flowchart for UART Transmit Programming Example

3.16.5.2 Flowchart for UART Receive Programming Example

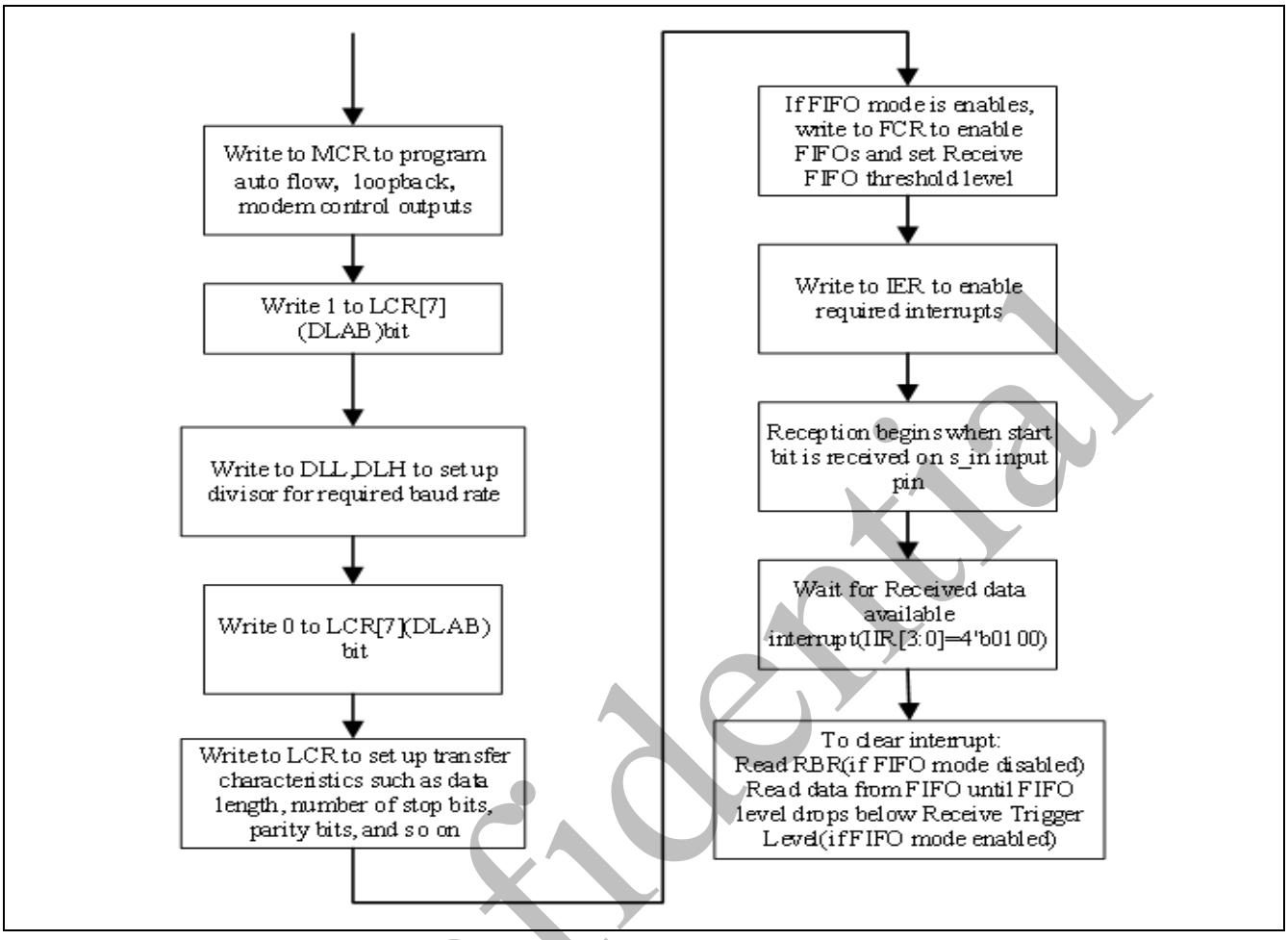


Figure 3-118 Flowchart for UART Receive Programming Example

3.16.6 UART Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
UART Base Address: UART0_BA = 0x4010_0000 UART1_BA = 0x4010_1000				
RBR	UARTx_BA +0x00	R	Receive Buffer Register	0x0000_0000
THR	UARTx_BA +0x00	W	Transmit Holding Register	0x0000_0000
DLL	UARTx_BA +0x00	R/W	Divisor Latch (Low)	0x0000_0000
DLH	UARTx_BA +0x04	R/W	Divisor Latch (High)	0x0000_0000
IER	UARTx_BA +0x04	R/W	Interrupt Enable Register	0x0000_0000
IIR	UARTx_BA +0x08	R	Interrupt Identification Register	0x0000_0001
FCR	UARTx_BA +0x08	W	FIFO Control Register	0x0000_0000
LCR	UARTx_BA +0x0C	R/W	Line Control Register	0x0000_0000
MCR	UARTx_BA +0x10	R/W	Modem Control Register	0x0000_0000
LSR	UARTx_BA +0x14	R	Line Status Register	0x0000_0060
MSR	UARTx_BA +0x18	R	Modem Status Register	0x0000_0000
SCR	UARTx_BA +0x1C	R/W	Scratchpad Register	0x0000_0000
USR	UARTx_BA +0x7C	R	UART Status Register	0x0000_0006
TFL	UARTx_BA +0x80	R	Transmit FIFO Level	0x0000_0000
RFL	UARTx_BA +0x84	R	Receive FIFO Level	0x0000_0000
HTX	UARTx_BA +0xA4	R/W	Halt TX	0x0000_0000
DMASA	UARTx_BA +0xA8	W	DMA Software Acknowledge	0x0000_0000
DLF	UARTx_BA +0xC0	R/W	Divisor Latch Fractional Value.	0x0000_0000
RAR	UARTx_BA +0xC4	R/W	Receive Address Register	0x0000_0000
TAR	UARTx_BA +0xC8	R/W	Transmit Address Register	0x0000_0000
LCR_EXT	UARTx_BA +0xCC	R/W	Line Extended Control Register	0x0000_0000

3.16.7 UART Register Description

3.16.7.1 Receive Buffer Register (RBR)

This register can be accessed only when the DIV_LATCH_ACC bit (LCR[7]) is cleared.

Register	Offset	R/W	Description	Reset Value
RBR x = 0, 1	UARTx_BA+0x00	R	Receive Buffer Register	0x0000_0000

Bits	Description	
[31:9]	Reserved	Reserved.
[8]	RBRM	Receive Buffer register (MSB 9th bit) Data byte received on the serial input port (sin) in UART mode for the MSB 9th bit.
[7:0]	RBRL	Receive Buffer Register (LSB 8 bits) Data byte received on the serial input port (sin) in UART mode, or the serial infrared input (sir_in) in infrared mode. The data in this register is valid only if DATA_RDY bit (LSR[0]) is set. If FIFOs are enabled (FCR[0] set to 1), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data character arrives, then the data already in the FIFO is preserved, but any incoming data are lost and an over-run error occurs.

3.16.7.2 Transmit Holding Register (THR)

This register can be accessed only when the DIV_LATCH_ACC bit (LCR[7]) is cleared.

Register	Offset	R/W	Description	Reset Value
THR x = 0, 1	UARTx_BA+0x00	W	Transmit Holding Register	0x0000_0000

Bits	Description	
[31:9]	Reserved	Reserved.
[8]	THRM	Transmit Holding Register (MSB 9th bit) Data to be transmitted on the serial output port (sout) in UART mode for the MSB 9^{th} bit.
[7:0]	THRL	Transmit Holding Register (LSB 8 bits) Data to be transmitted on the serial output port (sout) in UART mode or the serial infrared output (sir_out_n) in infrared mode. Data should only be written to the THR_EMPTY bit (LSR[5]) is set. If FIFOs are enabled (FCR[0] = 1) and THR_EMPTY is set, x number of characters of data may be written to the THR before the FIFO is full. The number x(default=16) is determined by the value of FIFO Depth that you set during configuration. Any attempt to write data when the FIFO is full results in the write data being lost.

3.16.7.3 Divisor Latch Low (DLL)

This register can be accessed only when the DIV_LATCH_ACC bit (LCR[7]) is cleared.

Register	Offset	R/W	Description	Reset Value
DLL x = 0, 1	UARTx_BA+0x00	R/W	Divisor Latch Low	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	DLL	Divisor Latch (Low) Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. **Note** that with the Divisor Latch Registers (DLL and DLH) set to 0, the baud clock is disabled and no serial communications occur. Also, once the DLL is set, at least 8clock cycles of the slowest UART clock should be allowed to pass before transmitting or receiving data.

3.16.7.4 Divisor Latch High (DLH)

This register can be accessed only when the DIV_LATCH_ACC bit (LCR[7]) is cleared.

Register	Offset	R/W	Description	Reset Value
DLH x = 0, 1	UARTx_BA+0x04	R/W	Divisor Latch High Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	DLH	Divisor Latch (High) Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. **Note** that with the Divisor Latch Registers (DLL and DLH) set to 0, the baud clock is disabled and no serial communications occur. Also, once the DLH is set, at least 8 clock cycles of the slowest UART clock should be allowed to pass before transmitting or receiving.

3.16.7.5 Interrupt Enable Register (IER)

This register can be accessed only when the DIV_LATCH_ACC bit (LCR[7]) is cleared.

Register	Offset	R/W	Description	Reset Value
IER x = 0, 1	UARTx_BA+0x04	R/W	Interrupt Enable Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved.
[7]	THR_EMPTY_IER	Programmable THR_EMPTY Interrupt Mode Enable. This is used to enable/disable the generation of THR_EMPTY Interrupt. 0–disabled 1–enabled
[6:4]	Reserved	Reserved
[3]	MS_IER	Enable Modem Status Interrupt. This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt. 0–disabled 1–enabled
[2]	RLS_IER	Enable Receiver Line Status Interrupt. This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0–disabled 1–enabled
[1]	THR_EMPTY_IER	Enable Transmit Holding Register Empty Interrupt. This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 – disabled 1 – enabled
[0]	RDA_IER	Enable Received Data Available Interrupt. This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt(if in FIFO mode and FIFOs enabled). These are the second highest priority interrupts. 0 – disabled 1 – enabled

3.16.7.6 Interrupt Identity Register (IIR)

Register	Offset	R/W	Description	Reset Value
IIR x = 0, 1	UARTx_BA+0x08	R	Interrupt Identity Register	0x0000_0001

Bits	Description	
[31:8]	Reserved	Reserved.
[7:6]	FIFOS_EN	FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00 – disabled 11 – enabled
[5:4]	Reserved	Reserved
[3:0]	ID_IRQ	Interrupt ID. This indicates the highest priority pending interrupt which can be one of the following types: 0000 – modem status 0001 – no interrupt pending 0010 – THR empty 0100 – received data available 0110 – receiver line status 1100 – character timeout The interrupt priorities are split into several levels that are detailed in Table 3-21.

Table 3-21 Interrupt Control Functions

ID_IRQ	Priorities Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0001	-	None	None	-
0110	Highest	Receive line status	Overrun/parity/ framing errors, break interrupt, or address received interrupt	Reading the line status register. In addition to LSR read, the Receiver line status is also cleared when RX_FIFO is read
0100	Second	Received data available	Receiver data available (non FIFO mode or FIFOs disabled) or RCVR FIFO trigger level reached (FIFO mode and FIFOs enabled)	Reading the receiver buffer register (non-FIFO mode or FIFOs disabled) or the FIFO drops below the trigger level (FIFO mode and FIFOs enabled)
1100	Second	Character timeout indication	No characters in or out of the RCVR FIFO during the last 4 character times and there is at least 1 character in it during this time	Reading the receiver buffer register
0010	Third	Transmit holding register empty	Transmitter holding register empty (Prog. THR_EMPTY Mode disabled) or XMIT FIFO at or below threshold	Reading the IIR register (if source of interrupt); or, writing into THR (FIFOs or THR_EMPTY Mode not selected or disabled) or XMIT

			(Prog. THR_EMPTY Mode enabled)	FIFO above threshold (FIFOs and THR_EMPTY Mode selected and enabled)
0000	Fourth	Modem status	Clear to send or data set ready or ring indicator or data carrier detect. Note that if auto flow control mode is enabled, a change in CLR_SEND (that is, D_CLR_SEND set) does not cause an interrupt.	Reading the Modem status register
0111	Fifth	Busy detect indication	UART_16550_COMPATIBLE= NO and master has tried to write to the Line Control Register while the UART is busy(USR[0] is set to 1).	Reading the UART status register

3.16.7.7 FIFO Control Register (FCR)

Register	Offset	R/W	Description	Reset Value
FCR x = 0, 1	UARTx_BA+0x08	W	FIFO Control Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7:6]	RCVR_TRIG	RCVR Trigger. This is used to select the trigger level in the receiver FIFO at which the Received Data Available Interrupt is generated. In auto flow control mode, this trigger is used to determine when the rts_n signal is de-asserted. It also determines when the dma_rx_req_n signal is asserted in certain modes of operation. 00 – 1 character in the FIFO 01 – FIFO ¼ full 10 – FIFO ½ full 11 – FIFO 2 less than full
[5:4]	TXE_TRIG	TX Empty Trigger. This is used to select the empty threshold level at which the THR_EMPTY Interrupts are generated when the mode is active. It also determines when the dma_tx_req_n signal is asserted when in certain modes of operation. 00 – FIFO empty 01 – 2 characters in the FIFO 10 – FIFO ¼ full 11 – FIFO ½ full
[3]	Reserved	Reserved
[2]	XMIT_FIFO_RST	XMIT FIFO Reset. This resets the control portion of the transmit FIFO and treats the FIFO as empty. This also de-asserts the DMA TX request and single signals. **Note** that this bit is 'self-clearing'. It is not necessary to clear this bit.
[1]	RCVR_FIFO_RST	RCVR FIFO Reset. This resets the control portion of the receive FIFO and treats the FIFO as empty. This also de-asserts the DMA RX request and single signals. **Note** that this bit is 'self-clearing'. It is not necessary to clear this bit.
[0]	FIFO_EN	FIFO Enable. This enables/disables the transmit (XMIT) and receive (RCVR) FIFOs. Whenever the value of this bit is changed both the XMIT and RCVR controller portion of FIFOs is reset.

3.16.7.8 Line Control Register (LCR)

Register	Offset	R/W	Description	Reset Value
LCR x = 0, 1	UARTx_BA+0x0C	R/W	Line Control Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7]	DIV_LATCH_ACC	Divisor Latch Access Bit. This bit is used to enable reading and writing of the Divisor Latch register (DLL and DLH) to set the baud rate of the UART. This bit must be cleared after initial baud rate setup in order to access other registers.
[6]	BREAK_CNTRL	Break Control Bit. This is used to cause a break condition to be transmitted to the receiving device. If set to 1, the serial output is forced to the spacing (logic 0) state. When not in Loop back Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. When in Loop back Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.
[5]	STICK_PARITY	Stick Parity. This bit is used to force parity value. When PARITY_EN, PARITY_SEL_OE, and Stick Parity are set to 1,the parity bit is transmitted and checked as logic 0. If PARITY_EN and Stick Parity are set to 1 and PARITY_SEL_OE is a logic 0, then parity bit is transmitted and checked as a logic 1. If this bit is set to 0, Stick Parity is disabled.
[4]	PARITY_SEL_OE	Even Parity Select. This is used to select between even and odd parity to be transmitted or checked, when parity is enabled (PARITY_EN set to 1). 0 – odd parity 1 – even parity
[3]	PARITY_EN	Parity Enable. This bit is used to enable and disable parity generation and detection in transmitted and received serial character respectively. 0 – parity disabled 1 – parity enabled
[2]	NUM_STOP	Number of stop bits. This is used to select the number of stop bits per character that the peripheral transmits and receives. 0 – 1 stop bit 1 – 1.5 stop bits when DATA_LEN_SEL (LCR[1:0]) is 0, else 2 stop bit **Note** that regardless of the number of stop bits selected, the receiver checks only the first stop bit.
[1:0]	DATA_LEN_SEL	Data Length Select. When *DATA_LEN_SEL_EXT* in LCR_EXT is set to 0, this register is used to select the number of data bits per character that the peripheral transmits and receives. The number of bits that may be selected are as follows: 00 – 5 bits

		01 – 6 bits 10 – 7 bits 11 – 8 bits

3.16.7.9 Modem Control Register (MCR)

Register	Offset	R/W	Description	Reset Value
MCR x = 0, 1	UARTx_BA+0x10	R/W	Modem Control Register	0x0000_0000

Bits	Description	
[31:7]	Reserved	Reserved
[6]	SIR_EN	SIR Mode Enable. (Write Protected) 0 – IrDA SIR Mode disabled 1 – IrDA SIR Mode enabled
[5]	AFC_EN	Auto Flow Control Enable. 0 – Auto Flow Control Mode disabled 1 – Auto Flow Control Mode enabled
[4]	LOOP_BACK	Loop Back Bit. This is used to put the UART into a diagnostic mode for test purposes. If operating in UART mode (SIR_MODE != Enabled or not active, MCR[6] set to 0), data on the sout line is held high, while serial data output is looped back to the sin line, internally. In this mode all the interrupts are fully functional. Also, in loop back mode, the modem control inputs (dsr_n, cts_n, ri_n, dcd_n) are disconnected and the modem control outputs (dtr_n, rts_n, out1_n, out2_n) are looped back to the inputs, internally.
[3]	OUTPUT2	OUTPUT2. This is used to directly control the user-designated Output2 (out2_n) output. The value written to this location is inverted and driven out on out2_n, that is: 0 – out2_n de-asserted (logic 1) 1 – out2_n asserted (logic 0) **Note** that in Loop back mode (MCR[4] set to 1), the out2_n output is held inactive high while the value of this location is internally looped back to an input.
[2]	OUTPUT1	OUTPUT1. This is used to directly control the user-designated Output1 (out1_n) output. The value written to this location is inverted and driven out on out1_n, that is: 0 – out1_n de-asserted (logic 1) 1 – out1_n asserted (logic 0) **Note** that in Loop back mode (MCR[4] set to 1), the out1_n output is held inactive high while the value of this location is internally looped back to an input.
[1]	SEND_REQ	Request to Send. This is used to directly control the Request to Send (rts_n) output. The Request To Send (rts_n) output is used to inform the modem or data set that the UART is ready to exchange data. When Auto SEND_REQ Flow Control is not enabled (MCR[5] set to 0), the rts_n signal is set low by programming MCR[1] (SEND_REQ) to a high.In Auto Flow Control, (MCR[5] set to 1) and FIFOs enable (FCR[0] set to 1), the rts_n output is controlled in the same way, but is also gated with the receiver FIFO threshold trigger (rts_n is inactive high when above the threshold) only when the RTC Flow Trigger is disabled; otherwise it is gated by the receiver FIFO almost-full trigger, where “almost full” refers to two available slots in the FIFO (rts_n is inactive high when

		above the threshold). The rts_n signal is de-asserted when MCR[1] is set low. Note that in Loop back mode (MCR[4] set to 1), the rts_n output is held inactive high while the value of this location is internally looped back to an input.
[0]	DATA_TERM_RDY	Data Terminal Ready. This is used to directly control the Data Terminal Ready (dtr_n) output. The value written to this location is inverted and driven out on dtr_n, that is: 0 – dtr_n de-asserted (logic 1) 1 – dtr_n asserted (logic 0) The Data Terminal Ready output is used to inform the modem or data set that the UART is ready to establish communications. Note that in Loop back mode (MCR[4] set to 1), the dtr_n output is held inactive high while the value of this location is internally looped back to an input.

3.16.7.10 Line Status Register (LSR)

Register	Offset	R/W	Description	Reset Value
LSR x = 0, 1	UARTx_BA+0x14	R	Line Status Register	0x0000_0060

Bits	Description	
[31:9]	Reserved	Reserved
[8]	ADDR_RX	Address Received bit If 9-bit data mode (LCR_EXT[0]=1) is enabled, this bit is used to indicate that the 9th bit of the receive data is set to 1. This bit can also be used to indicate whether the incoming character is an address or data. 1 - Indicates that the character is an address. 0 - Indicates that the character is data. In the FIFO mode, since the 9th bit is associated with the received character, it is revealed when the character with the 9th bit set to 1 at the top of the FIFO list. Reading the LSR clears the 9th bit. **Note**: You must ensure that an interrupt gets cleared (reading LSR register) before the next address byte arrives. If there is a delay in clearing the interrupt, then software will not be able to distinguish between multiple address related interrupt.
[7]	RX_FIFO_ERR	Receiver FIFO Error bit. This bit is only relevant when FIFOs are enabled (FCR[0] set to 1). This is used to indicate if there is at least one parity error, framing error, or break indication in the FIFO. 0 – no error in RX FIFO 1 – error in RX FIFO This bit is cleared when the LSR is read and the character with the error is at the top of the receiver FIFO and there are no subsequent errors in the FIFO.
[6]	TX_EMPTY	Transmitter Empty bit. If in FIFO mode and FIFOs enabled (FCR[0] set to 1), this bit is set whenever the Transmitter Shift Register and the FIFO are both empty. If FIFOs are disabled, this bit is set whenever the Transmitter Holding Register and the Transmitter Shift Register are both empty.
[5]	THR_EMPTY	Transmit Holding Register Empty bit. If THR_EMPTY mode is disabled (IER[7] set to 0) and regardless of FIFO's being implemented/enabled or not, this bit indicates that the THR or TX FIFO is empty. This bit is set whenever data is transferred from the THR or TX FIFO to the transmitter shift register and no new data has been written to the THR or TX FIFO. This also causes a THR_EMPTY Interrupt to occur, if the THR_EMPTY Interrupt is enabled. If THR_EMPTY_MODE_USER = Enabled and both modes are active (IER[7] set to 1 and FCR[0] set to 1 respectively), the functionality is switched to indicate the transmitter FIFO is full, and no longer controls THR_EMPTY interrupts, which are then controlled by the FCR[5:4] threshold setting.
[4]	BREAK_IRQ	Break Interrupt bit. This is used to indicate the detection of a break sequence on the serial input data.If in UART mode (SIR_MODE = Disabled), it is set whenever the serial input, sin, is held in a logic '0' state for longer than the sum of start time + data bits + parity + stop

		bits. A break condition on serial input causes one and only one character, consisting of all 0s, to be received by the UART. In FIFO mode, the character associated with the break condition is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the LSR clears the BREAK_IRQ bit. **Note**: If a FIFO is full when a break condition is received, a FIFO overrun occurs. The break condition and all the information associated with it—parity and framing errors—is discarded; any information that a break character was received is lost.
[3]	FRAM_ERR	Framing Error bit. This is used to indicate the occurrence of a framing error in the receiver.A framing error occurs when the receiver does not detect a valid NUM_STOP bit in the received data.In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error is at the top of the FIFO. When a framing error occurs, the UART tries to resynchronize. It does this by assuming that the error was due to the start bit of the next character and then continues receiving the other bit; that is, data, and/or parity and stop. It should be noted that the Framing Error (FRAM_ERR) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BREAK_IRQ) bit (LSR[4]). This happens because the break character implicitly generates a framing error by holding the sin input to logic 0 for longer than the duration of a character. 0 – no framing error 1 – framing error Reading the LSR clears the FRAM_ERR bit.
[2]	PARITY_ERR	Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PARITY_EN) bit (LCR[3]) is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the parity error arrives at the top of the FIFO.It should be noted that the Parity Error (PARITY_ERR) bit (LSR[2]) can be set if a break interrupt has occurred, as indicated by Break Interrupt (BREAK_IRQ) bit (LSR[4]). In this situation, the Parity Error bit is set if parity generation and detection is enabled (LCR[3]=1) and the parity is set to odd (LCR[4]=0). 0 – no parity error 1 – parity error Reading the LSR clears the PARITY_ERR bit.
[1]	OVER_ERR	Overrun error bit. This is used to indicate the occurrence of an overrun error. This occurs if a new data character was received before the previous data was read. In the non-FIFO mode, the OVER_ERR bit is set when a new character arrives in the receiver before the previous character was read from the RBR. When this happens, the data in the RBR is overwritten. In the FIFO mode, an overrun error occurs when the FIFO is full and a new character arrives at the receiver. The data in the FIFO is retained and the data in the receive shift register is lost. 0 – no overrun error 1 – overrun error Reading the LSR clears the OVER_ERR bit.
[0]	DATA_RDY	Data Ready bit. This is used to indicate that the receiver contains at least one character in the RBR

		or the receiver FIFO. 0 – no data ready 1 – data ready This bit is cleared when the receiver FIFO is empty, in FIFO mode.

3.16.7.11 Modem Status Register (MSR)

Register	Offset	R/W	Description	Reset Value
MSR x = 0, 1	UARTx_BA+0x18	R	Modem Status Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7]	DATA_CAR_DET	Data Carrier Detect. This is used to indicate the current state of the modem control line dcd_n. This bit is the complement of dcd_n. When the Data Carrier Detect input (dcd_n) is asserted it is an indication that the carrier has been detected by the modem or data set. 0 – dcd_n input is de-asserted (logic 1) 1 – dcd_n input is asserted (logic 0) In Loop back Mode (MCR[4] set to 1), DATA_CAR_DET is the same as MCR[3] (Out2).
[6]	RING_INDIC	Ring Indicator. This is used to indicate the current state of the modem control line ri_n.This bit is the complement of ri_n. When the Ring Indicator input (ri_n) is asserted it is an indication that a telephone ringing signal has been received by the modem or data set. 0 – ri_n input is de-asserted (logic 1) 1 – ri_n input is asserted (logic 0) In Loop back Mode (MCR[4] set to 1), RING_INDIC is the same as MCR[2] (Out1).
[5]	DATA_SET_RDY	Data Set Ready. This is used to indicate the current state of the modem control line dsr_n.This bit is the complement of dsr_n. When the Data Set Ready input (dsr_n) is asserted it is an indication that the modem or data set is ready to establish communications with the UART. 0 – dsr_n input is de-asserted (logic 1) 1 – dsr_n input is asserted (logic 0) In Loopback Mode (MCR[4] set to 1), DATA_SET_RDY is the same as MCR[0] (DATA_TERM_RDY).
[4]	CLR_SEND	Clear to Send. This is used to indicate the current state of the modem control line cts_n. This bit is the complement of cts_n. When the Clear to Send input (cts_n) is asserted it is an indication that the modem or data set is ready to exchange data with the UART. 0 – cts_n input is de-asserted (logic 1) 1 – cts_n input is asserted (logic 0) In Loop back Mode (MCR[4] = 1), CLR_SEND is the same as MCR[1] (SEND_REQ).
[3]	D_DATA_CAR_DET	Delta Data Carrier Detect. This is used to indicate that the modem control line dcd_n has changed since the last time the MSR was read. 0 – no change on dcd_n since last read of MSR 1 – change on dcd_n since last read of MSR Reading the MSR clears the D_DATA_CAR_DET bit. In Loop back Mode (MCR[4] = 1), D_DATA_CAR_DET reflects changes on MCR[3] (Out2).

		Note, if the D_DATA_CAR_DET bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the D_DATA_CAR_DET bit is set when the reset is removed if the dcd_n signal remains asserted.
[2]	TERI	Trailing Edge of Ring Indicator. This is used to indicate that a change on the input ri_n (from an active-low to an inactive-high state) has occurred since the last time the MSR was read. 0 – no change on ri_n since last read of MSR 1 – change on ri_n since last read of MSR Reading the MSR clears the TERI bit. In Loop back Mode (MCR[4] = 1), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.
[1]	D_DATA_SET_RDY	Delta Data Set Ready. This is used to indicate that the modem control line dsr_n has changed since the last time the MSR was read. 0 – no change on dsr_n since last read of MSR 1 – change on dsr_n since last read of MSR Reading the MSR clears the D_DATA_SET_RDY bit. In Loop back Mode (MCR[4] = 1), D_DATA_SET_RDY reflects changes on MCR[0] (DATA_TERM_RDY). **Note**, if the D_DATA_SET_RDY bit is not set and the dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the D_DATA_SET_RDY bit is set when the reset is removed if the dsr_n signal remains asserted.
[0]	D_CLR_SEND	Delta Clear to Send. This is used to indicate that the modem control line cts_n has changed since the last time the MSR was read. 0 – no change on cts_n since last read of MSR 1 – change on cts_n since last read of MSR Reading the MSR clears the D_CLR_SEND bit. In Loop back Mode (MCR[4] = 1), D_CLR_SEND reflects changes on MCR[1] (SEND_REQ). **Note**, if the D_CLR_SEND bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the D_CLR_SEND bit is set when the reset is removed if the cts_n signal remains asserted.

3.16.7.12 Scratchpad Register (SCR)

Register	Offset	R/W	Description	Reset Value
SCR x = 0, 1	UARTx_BA+0x1C	R/W	Scratchpad Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7:0]	SCR	This register is for programmers to use as a temporary storage space. It has no defined purpose in the UART.

3.16.7.13 UART Status Register (USR)

Register	Offset	R/W	Description	Reset Value
USR x = 0, 1	UARTx_BA+0x7C	R	UART Status Register	0x0000_0006

Bits	Description	
[31:5]	Reserved	Reserved
[4]	RX_FIFO_FULL	Receive FIFO Full. This is used to indicate that the receive FIFO is completely full. 0 – Receive FIFO not full 1 – Receive FIFO Full This bit is cleared when the RX FIFO is no longer full.
[3]	RX_FIFO_NE	Receive FIFO Not Empty. This is used to indicate that the receive FIFO contains one or more entries. 0 – Receive FIFO is empty 1 – Receive FIFO is not empty This bit is cleared when the RX FIFO is empty.
[2]	TX_FIFO_E	Transmit FIFO Empty. This is used to indicate that the transmit FIFO is completely empty. 0 – Transmit FIFO is not empty 1 – Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty.
[1]	TX_FIFO_NF	Transmit FIFO Not Full. This is used to indicate that the transmit FIFO in not full. 0 – Transmit FIFO is full 1 – Transmit FIFO is not full This bit is cleared when the TX FIFO is full.
[0]	Reserved	Reserved

3.16.7.14 Transmit FIFO Level (TFL)

Register	Offset	R/W	Description	Reset Value
TFL x = 0, 1	UARTx_BA+0x80	R	Transmit FIFO Level	0x0000_0000

Bits	Description	
[31:5]	Reserved	Reserved
[4:0]	TFL	Transmit FIFO Level. This indicates the number of data entries in the transmit FIFO.

3.16.7.15 Receive FIFO Level (RFL)

Register	Offset	R/W	Description	Reset Value
RFL x = 0, 1	UARTx_BA+0x84	R	Receive FIFO Level	0x0000_0000

Bits	Description	
[31:5]	Reserved	Reserved
[4:0]	RFL	Receive FIFO Level. This indicates the number of data entries in the receive FIFO.

3.16.7.16 Halt TX (HTX)

Register	Offset	R/W	Description	Reset Value
HTX x = 0, 1	UARTx_BA+0XA4	R/W	Halt TX	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved
[0]	HTX	This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are implemented and enabled. 0 = Halt TX disabled 1 = Halt TX enabled **Note**, if FIFOs are implemented and not enabled, the setting of the halt TX register has no effect on operation.

3.16.7.17 DMA Software Acknowledge (DMASA)

Register	Offset	R/W	Description	Reset Value
DMASA x = 0, 1	UARTx_BA+0xA8	W	DMA Software Acknowledge	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved
[0]	DMASA	DMA Software Acknowledge. This register is use to perform a DMA software acknowledge if a transfer needs to be terminated due to an error condition. For example, if the DMA disables the channel, then the UART should clear its request. This causes the TX request, TX single, RX request and RX single signals to de-assert. **Note** that this bit is 'self-clearing'. It is not necessary to clear this bit.

3.16.7.18 Divisor Latch Fraction Register (DLF)

Register	Offset	R/W	Description	Reset Value
DLF x=0,1	UARTx_BA+0xC0	R/W	Divisor Latch Fraction Register	0x0000_0000

Bits	Description	
[31:4]	Reserved	Reserved
[3:0]	DLF	Fractional part of divisor. The fractional value is added to integer value set by DLH, DLL. Fractional value is determined by (Divisor Fraction value)/(2^DLF_SIZE). This table describes the DLF Values to be programmed for DLF_SIZE=4.

DLF Value	Fraction	Fractional Value
0000	0/16	0.0000
0001	1/16	0.0625
0010	2/16	0.125
0011	3/16	0.1875
0100	4/16	0.25
0101	5/16	0.3125
0110	6/16	0.375
0111	7/16	0.4375
1000	8/16	0.5
1001	9/16	0.5625
1010	10/16	0.625
1011	11/16	0.6875
1100	12/16	0.75
1101	13/16	0.8125
1110	14/16	0.875
1111	15/16	0.9375

3.16.7.19 Receive Address Register (RAR)

Register	Offset	R/W	Description	Reset Value
RAR x=0,1	UARTx_BA+0xC4	R/W	Receive Address Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7:0]	RAR	This is an address matching register during receive mode. If the 9-th bit is set in the incoming character then the remaining 8-bits will be checked against this register value. If the match happens then sub-sequent characters with 9-th bit set to 0 will be treated as data byte until the next address byte is received. **Note**: This register is applicable only when 'MATCH_ADDR' (LCR_EXT[1]) and 'DATA_LEN_SEL_EXT' (LCR_EXT[0]) bits are set to 1.

3.16.7.20 Transmit Address Register (TAR)

Register	Offset	R/W	Description	Reset Value
TAR	UARTx_BA+0xC8	R/W	Transmit Address Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7:0]	TAR	This is an address matching register during transmit mode. If DATA_LEN_SEL_EXT (LCR_EXT[0]) bit is enabled, then UART sends the 9-bit character with 9-th bit set to 1 and remaining 8-bit address will be sent from this register provided 'SEND_ADDR_CNTRL' (LCR_EXT[2]) bit is set to 1. **Note**: • This register is used only to send the address. The normal data should be sent by programming THR register. • Once the address is started to send on the UART serial lane, then 'SEND_ADDR_CNTRL' bit will be auto-cleared by the hardware.

3.16.7.21 Line Extended Control Register (LCR_EXT)

Register	Offset	R/W	Description	Reset Value
LCR_EXT x=0,1	UARTx_BA+0xCC	R/W	Line Extended Control Register	0x0000_0000

Bits	Description	
[31:4]	Reserved	Reserved
[3]	TX_MODE_CNTRL	Transmit mode control bit. This bit is used to control the type of transmit mode during 9-bit data transfers. 1 = In this mode of operation, Transmit Holding Register (THR) is 9-bit wide. You must ensure that the THR register is written correctly for address/data. Address: 9th bit is set to 1, Data: 9th bit is set to 0. **Note**: Transmit address register (TAR) is not applicable in this mode of operation. 0 = In this mode of operation, Transmit Holding Register (THR) are 8-bit wide. The user needs to program the address into Transmit Address Register (TAR) and data into the THR register. SEND_ADDR_CNTRL bit is used as a control knob to indicate the UART on when to send the address.
[2]	SEND_ADDR_CNTRL	Send address control bit. This bit is used as a control knob for the user to determine when to send the address during transmit mode. 1 = 9-bit character will be transmitted with 9-th bit set to 1 and the remaining 8-bits will match to what is being programmed in "Transmit Address Register". 0 = 9-bit character will be transmitted with 9-th bit set to 0 and the remaining 8-bits will be taken from the Tx FIFO which is programmed through 8-bit wide THR/STHR register. **Note**: 1. This bit is auto-cleared by the hardware, after sending out the address character. User is not expected to program this bit to 0. 2. This field is applicable only when DATA_LEN_SEL_EXT bit is set to 1 and TX_MODE_CNTRL is set to 0.
[1]	MATCH_ADDR	Address Match Mode. This bit is used to enable the address match feature during receive. 1 = Address match mode; UART will wait until the incoming character with 9-th bit set to 1. and, further checks to see if the address matches with what is programmed in "Receive Address Match Register". If match is found, then sub-sequent characters will be treated as valid data and UART starts receiving data. 0 = Normal mode; UART will start to receive the data and 9-bit character will be formed and written into the receive Rx FIFO. User is responsible to read the data and differentiate b/n address and data. **Note**: This field is applicable only when DATA_LEN_SEL_EXT is set to 1.
[0]	DATA_LEN_SEL_EXT	Extension for DATA_LEN_SEL. This bit is used to enable 9-bit data for transmit and receive transfers. 1 = 9 bits per character 0 = Number of data bits selected by DATA_LEN_SEL

3.17 Timer Controller (TMR1/2)

3.17.1 Overview

The PAN101x chip includes three same timer modules, TMR0, TMR1 and TMR2, allowing user to easily implement a timer control for applications. TMR0 locates in apb1 subsystem, while TMR1 and TMR2 locate in apb2 subsystem. The timer can perform functions, such as frequency measurement, delay timing, clock generation, event counting by external input pins, and interval measurement by external capture pins.

3.17.2 Features

- Each set of timer equipped with 24-bit up counter and one 8-bit prescale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle-output and continuous counting operation modes
- 24-bit up counter value is readable through CNT (CNT[23:0])
- Supports event counting function
- 24-bit capture value is readable through CAPDAT (CAP[23:0])
- Supports interval measurement for external capture pin event or 32KHz clock source
- Supports external capture pin event to reset 24-bit up counter
- Supports chip wake-up when timer generated wake up interrupt signal

3.17.3 Block Diagram

The timer controller block diagram and clock control are shown in Figure 3-119 and Figure 3-120.

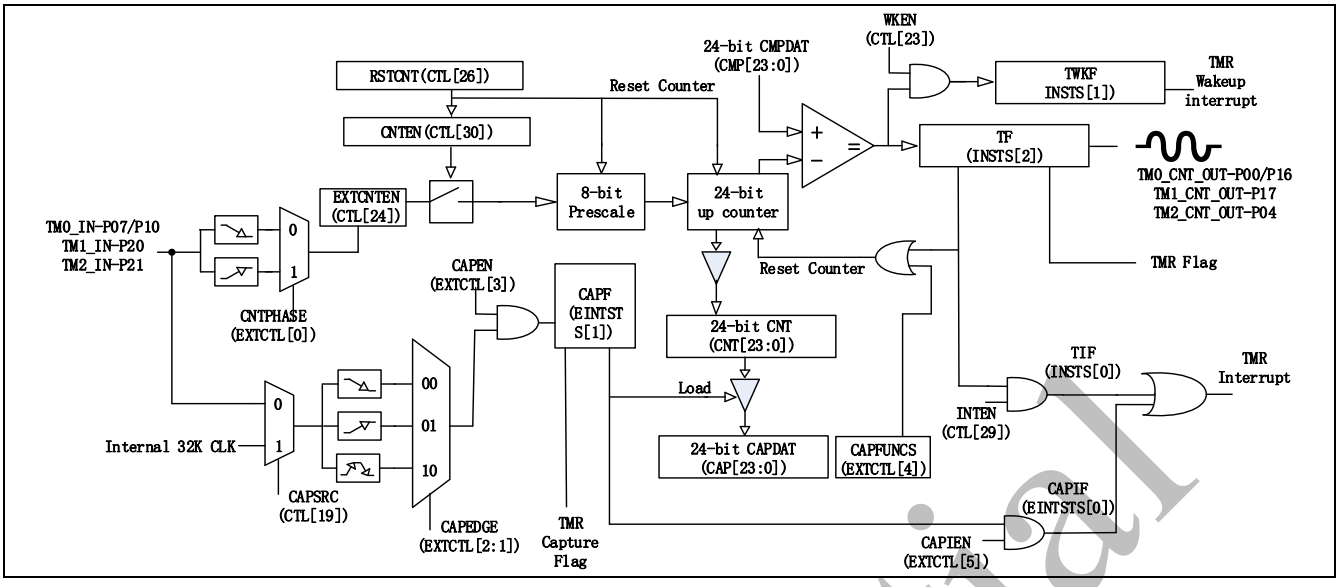


Figure 3-119 Timer Controller Block Diagram

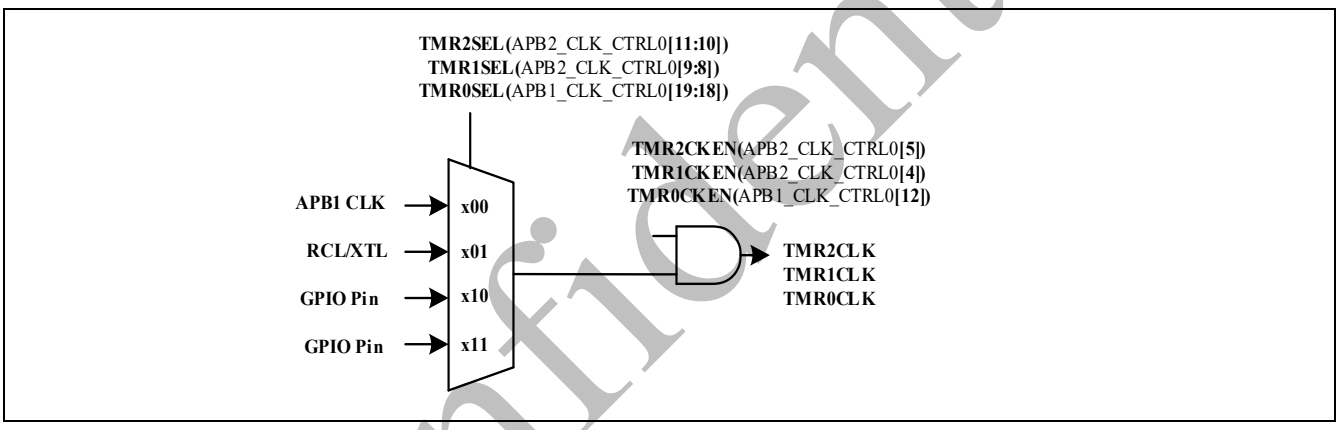


Figure 3-120 Clock Source of Timer Controller

3.17.4 Basic Configuration

3.17.4.1 Clock Source Setting

TMRx clock source can be configured by TMRxSEL field in APB1_CLK_CTRL0 or APB2_CLK_CTRL0 register. You can choose from APB_CLK, RCL/XTL or TMx_IN. APB_CLK is apb subsystem clock source. RCL/XTL refers to 32kHz clock source. TMx_IN is input from GPIO pin, showed in Figure 3-120.

When choose apb subsystem clock as TMRx clock source, you need follow below configuration. The module TMR0 is belong to APB1 subsystem, while TMR1 and TMR2 are belong to APB2 subsystem. Firstly enable TMRx(x refer to 1~2) clock source, you need write 1 to APB1_CLK_EN or APB2_CLK_EN in AHB_CLK_CTRL register respectively.

Secondly, TMR0cken in APB1_CLK_CTRL0 register will enable TMR0 clock, and TMR1cken or TMR2cken in APB2_CLK_CTRL0 register will enable TMR1 or TMR2 clock respectively.

3.17.4.2 Timer Flag

The timer controller supports two flags; one is TF (INTSTS[2]) which is set while timer counter value CNT (CNT[23:0]) matches the timer compared value CMPDAT (CMP[23:0]), and the other is CAPF (EINTSTS[1]) which is set when the transition on the TMx_EXT pin is consistent with CAPEDGE (EXTCTL[2:1]) setting.

3.17.4.3 Timer Interrupt Flag

The timer controller supports two interrupt flags:

One is TIF (INTSTS[0]) which is set while timer counter value CNT (CNT[23:0]) matches the timer compared value CMPDAT (CMP[23:0])and INTEN (CTL[29]) is set to 1.

The other is CAPIF (TIMERx_EINTSTS[0]) which is set when the transition on the TMRx_EXT pin is in accord with CAPEDGE (EXTCTL[2:1]) setting, and CAPIEN (EXTCTL[5]) is set to 1.

The only difference between timer flag and timer interrupt flag is that whether interrupt enable signal is set or not.

3.17.5 Functional Description

3.17.5.1 Timer Counting Operation Mode

The Timer controller provides four timer counting modes: One-shot, Periodic, Toggle-output and Continuous Counting operation modes as described below.

One-shot Mode

If the timer controller is configured at one-shot mode OPMODE (CTL[28:27] is 2’b00) and CNTEN (CTL[30]) is set, the timer counter starts up counting. Once the CNT (CNT[23:0]) value reaches CMPDAT (CMP[23:0]) value, the TF (INTSTS[2]) will be set to 1, CNT value and CNTEN bit is cleared automatically by timer controller then timer counting operation stops. In the meantime, if the INTEN (CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

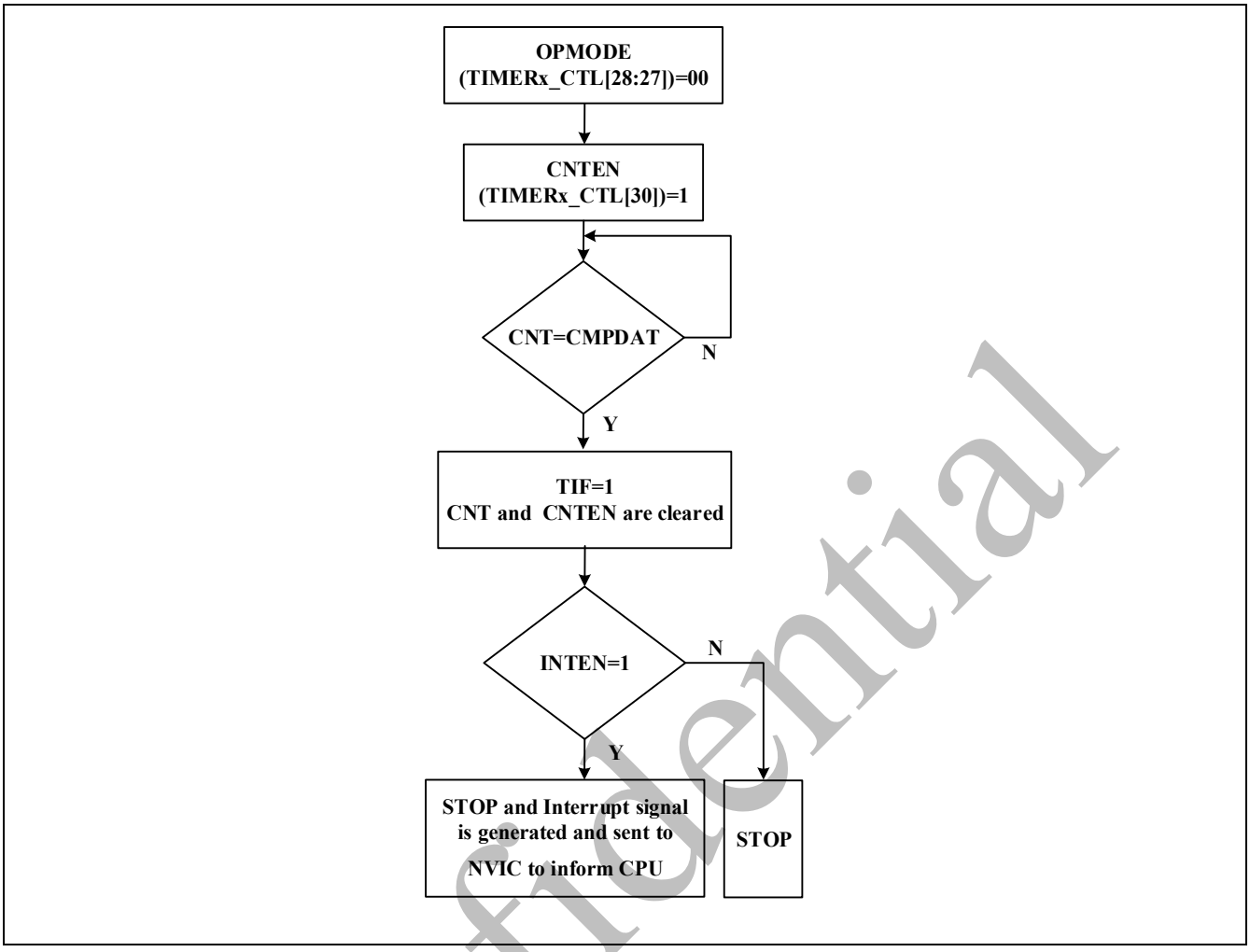


Figure 3-121 One-shot Mode

Periodic Mode

If the timer controller is configured at periodic mode (CTL[28:27] is 2'b01) and CNTEN (CTL[30]) is set, the timer counter starts up counting. Once the CNT (CNT[23:0]) value reaches CMPDAT (CMP[23:0]) value, the TF (INTSTS[2]) will be set to 1, CNT value will be cleared automatically by timer controller and timer counter operates counting again. In the meantime, if the INTEN (CTL[29]) bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. In this mode, the timer controller operates counting and compares with CMPDAT value periodically until the CNTEN bit is cleared by user.

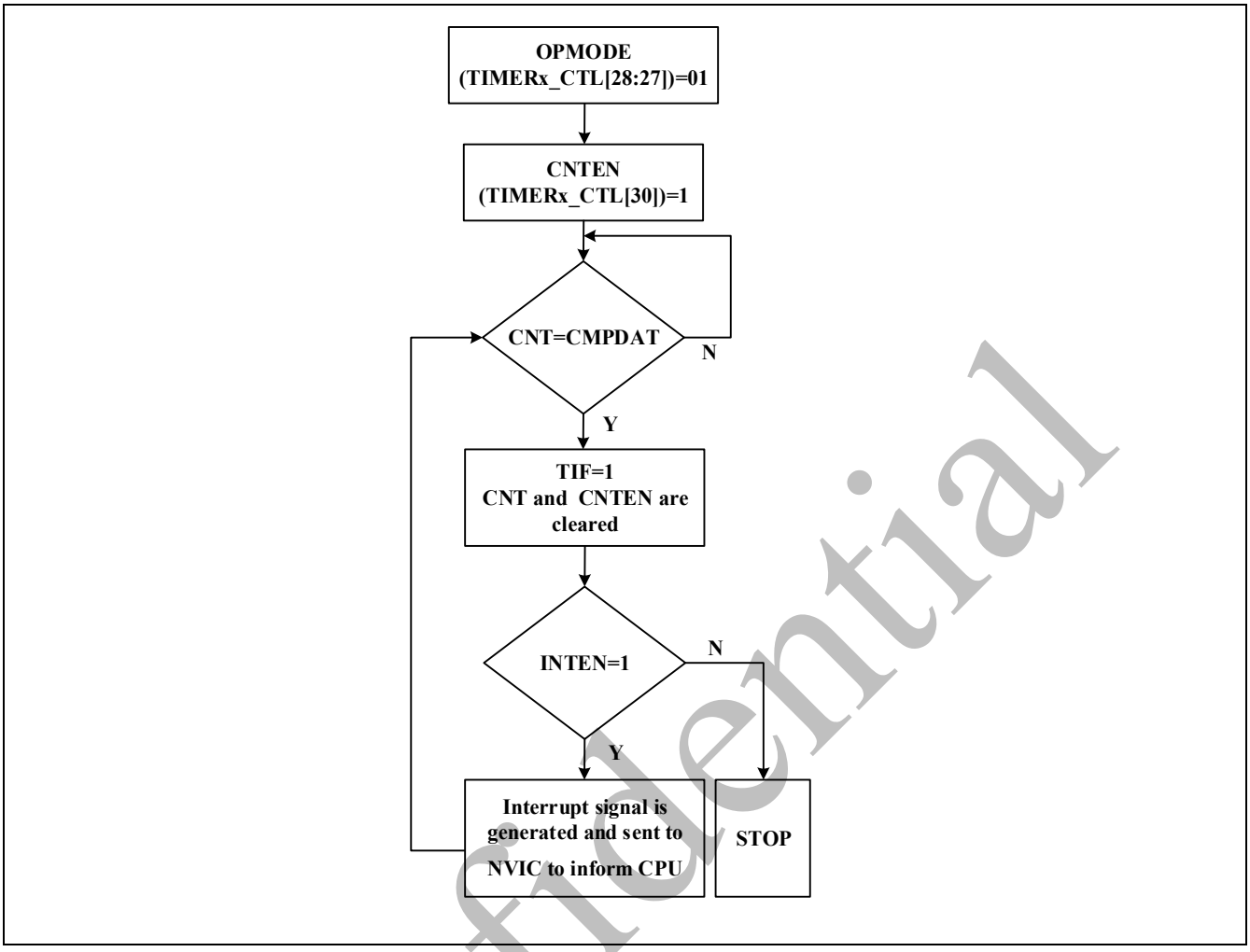


Figure 3-122 Periodic Mode

Toggle-output Mode

If the timer controller is configured at toggle-output mode (CTL[28:27] is 2'b10) and CNTEN (CTL[30]) is set, the timer counter starts up counting. The counting operation of toggle-output mode is almost the same as periodic mode, except toggle-output mode has associated TMR0 ~ TMR2 pin to output signal while specify TIF (INTSTS[0]) is set. So in this mode, INTEN (CTL[29]) must be set. Normally, the toggle-output signal on TMRx_CNT_OUT（x=0,1,2）pin is high and changing back and forth with 50% duty cycle. For other duty cycle signal, firstly, do not clear TIF flag in your ISR, only clear TF flag, secondly count TF triggered times, for example, if you want to get a 75% or 25% duty ratio signal, you need clear TIF flag when TF flag reach 2 times, then clear TIF flag when TF flag reach 1 time. GPIO output pin P00 and P16 are corresponding to TMR0, P17 and P04 are corresponding to TMR1 and TMR2 respectively.

Continuous Counting Mode

If the timer controller is configured at continuous counting mode (CTL[28:27] is 2'b11) and CNTEN (CTL[30]) is set, the timer counter starts up counting. Once the CNT (CNT[23:0]) value reaches the CMPDAT (CMP[23:0]) value, the TF (INTSTS[0]) will be set to 1 and the CNT value keeps up counting until reach 2^24-1. In the meantime, if the INTEN (CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU. User can change different CMPDAT value immediately without disabling timer counting and restarting timer counting in this mode.

For example, CMPDAT value is set as 80, first. The TF will set to 1 when CNT value is equal to 80, the timer counter is kept counting and CNT value will not go back to 0, it continues to count 81, 82, 83,…to 2^24-1, 0, 1, 2, 3, …to 2^24-1 again and again. Next, if user programs the CMPDAT value as 200 and clears TF, the TF will be set to 1 again when CNT value reaches 200. At last, user programs CMPDAT as 500 and clears TF, the TF will set to 1 again when CNT value reaches to 500. In this mode, the timer counting is continuous. You can update CMPDAT continuously when system detected TF or TIF flag or update CMPDAT value in your ISR function.

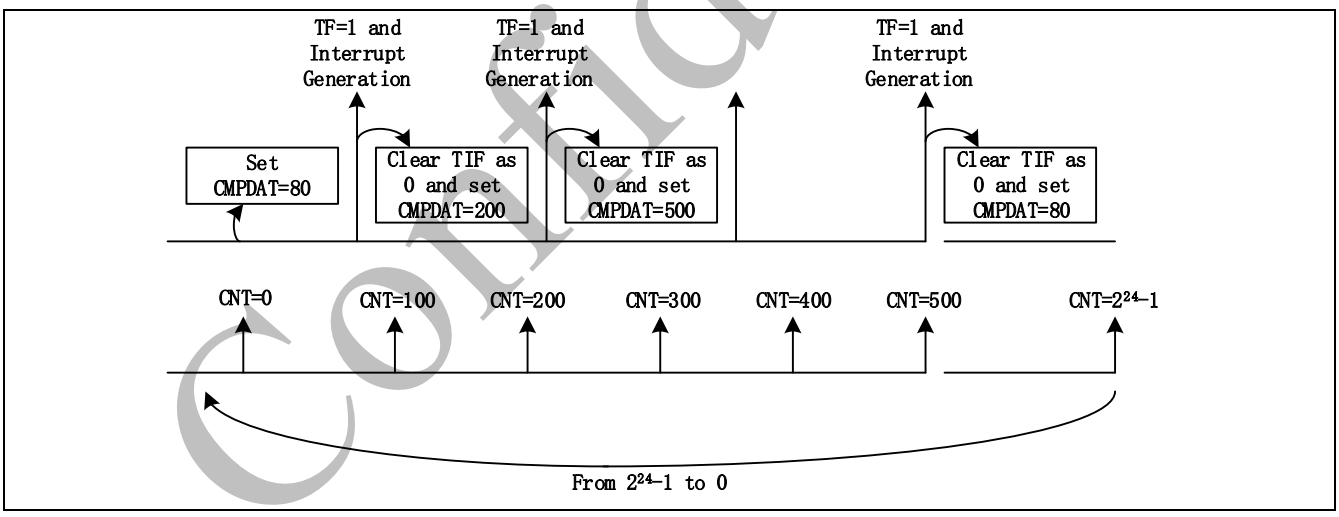


Figure 3-123 Continuous Counting Mode

3.17.5.2 Event Counting Mode

The timer controller also provides an application which can count the input event from GPIO input pin..The number of event will reflect to CNT (CNT[23:0]) value. It is also called as event counting function. In this function, EXTCNTEN (CTL[24]) should be set and the timer peripheral clock source should be set as APB_CLK.

User can enable or disable TMRx pin de-bounce circuit by setting CNTDBEN (EXTCTL[7]).

The input event frequency should be less than 1/3 APB_CLK if TMx pin de-bounce disabled or less than 1/8 APB_CLK if TMRx pin de-bounce enabled to assure the returned CNT value is correct, and user can also select edge detection phase of TMRx pin by setting CNTPHASE (EXTCTL[0]) bit.

In event counting mode, the timer counting operation mode can be selected as one-shot, periodic and continuous counting mode to counts the counter value CNT (CNT[23:0]) from TMRx pin.

3.17.5.3 Input Capture Function

The input capture or reset function is provided to capture or reset timer counter value. The capture function with free-counting capture mode and trigger-counting capture mode are configured by CAPSEL (EXTCTL[8]). The free-counting capture mode, external reset counter mode, trigger-counting capture mode are described as follows. Input capture function can both capture TMRx_EXT GPIO input pin signal and internal 32KHz clock, which can be set in CAPSRC(CTL[19]) register. Following operation modes set TMRx_EXT pin input signal as instruction.

Free-Counting Capture Mode

The event capture function is used to load CNT (CNT[23:0]) value to CAPDAT (CAP[23:0]) value while edge transition detected on TMRx_EXT (x= 1~2) pin. In this mode, CAPSEL (EXTCTL[8]) and CAPFUNCS (EXTCTL[4]) should be as 0. The expected transition on TMRx_EXT pin will trigger event capture function. The timer peripheral clock source should be set as APB_CLK.

User can enable or disable TMRx_EXT pin de-bounce circuit by setting CAPDBEN (EXTCTL[6]). The transition frequency of TMRx_EXT pin should be less than 1/3 APB_CLK if TMRx_EXT pin de-bounce disabled, less than 1/8 APB_CLK if TMRx_EXT pin de-bounce enabled to assure the capture function can be work normally, and user can also select edge transition detection of TMRx_EXT pin by setting CAPEDGE (EXTCTL[2:1]).

In event capture mode, user does not consider what timer counting operation mode is selected, the capture event occurred only if edge transition on TMRx_EXT pin is detected.

Users must consider the Timer will keep register CAP unchanged and drop the new capture value, if the CPU does not clear the CAPIF (EINTSTS[0]) status. The operation method is described in Figure 3-124.

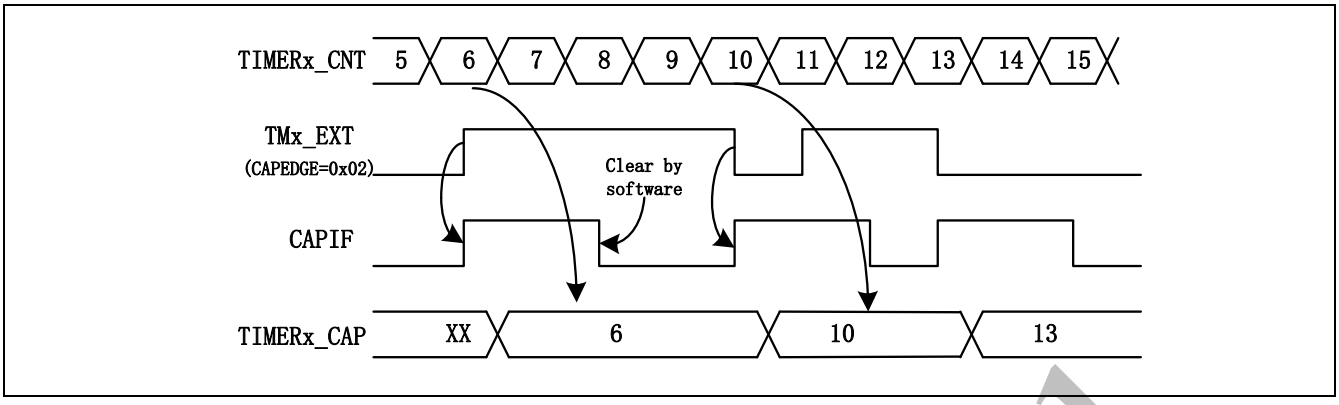


Figure 3-124 Free-Counting Capture Mode

External Reset Counter Mode

The timer controller also provides reset counter function to reset CNT (CNT[23:0]) value while edge transition detected on TMRx_EXT (x= 1~2). In this mode, most the settings are the same as event capture mode except CAPFUNCS (EXTCTL[4]) should be as 1 for select TMRx_EXT transition is using to trigger reset counter value. The operation method is also described in Figure 3-125. The difference between Free-Counting Capture Mode and External Reset Counter Mode is whether the TMR_CNT value will be cleared or not.

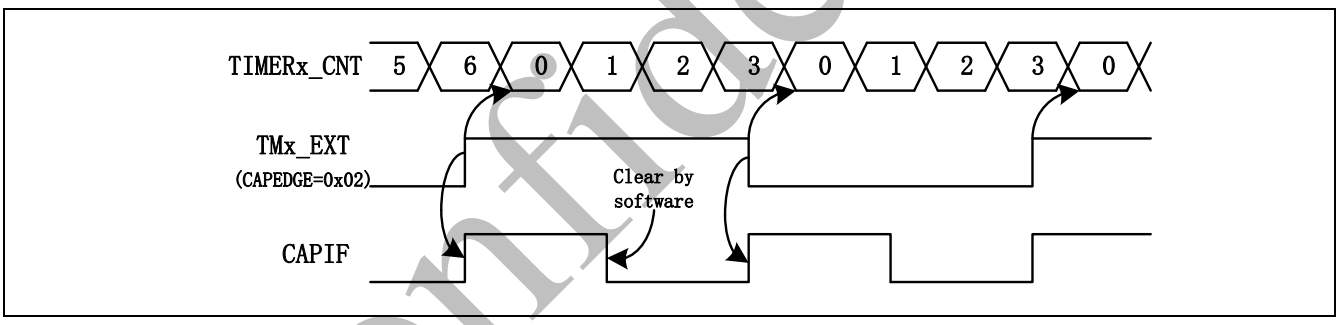


Figure 3-125 External Reset Counter Mode

Trigger-Counting Capture Mode

If CAPSEL (EXTCTL[8]) is set to 1, CAPEN (EXTCTL[3]) is set to 1 and CAPFUNCS (EXTCTL[4]) is set to 0, the CNT will be reset to 0 then captured into CAPDAT register when TMRx_EXT (x= 1~2) pin trigger condition occurred. The TMRx_EXT expected edge can be chosen by CAPEDGE (EXTCTL[2:1]). The detailed operation method is described in Table 3-22.

Firstly when TMRx_EXT expected edge occurr, the TMRx CNT will start count from zero. Secondly TMRx_EXT expected edge reoccur, CAPF (EINTSTS[1]) will set to 1, and the interrupt signal is generated if CAPIEN (EXTCTL[5]) is enabled. Once CAPIF(EINTSTS[0]) is detected, you need disable CAPIEN in ISR function. The TMRx_EXT source operating frequency should be less than 1/3 APB_CLK frequency if disable TMRx_EXT de-bounce or

less than 1/8 APB_CLK frequency if enabling TMRx_EXT de-bounce. It also provides TMRx_EXT enabled or disabled capture de-bounce function by CAPDBEN (EXTCTL[6]).

Table 3-22 Input Capture Mode Operation

Function	CAPSEL (EXTCTL[8])	CAPFUNCS (EXTCTL[4])	CAPEDGE (EXTCTL[2:1])	Operation Description
Free-counting Capture Mode	0	0	00	A 1 to 0 transition on TMRx_EXT (x= 1~2) pin is detected. CNT is captured to CAPDAT.
	0	0	01	A 0 to 1 transition on TMRx_EXT (x= 1~2) pin is detected. CNT is captured to CAPDAT.
	0	0	10	Either 1 to 0 or 0 to 1 transition on TMRx_EXT (x= 1~2) pin is detected. CNT is captured to CAPDAT.
	0	0	11	Reserved
External Reset Counter Mode	0	1	00	A 1 to 0 transition on TMRx_EXT (x= 1~2) pin is detected. CNT is reset to 0.
	0	1	01	A 0 to 1 transition on TMRx_EXT (x= 1~2) pin is detected. CNT is reset to 0.
	0	1	10	Either 1 to 0 or 0 to 1 transition on TMRx_EXT (x= 1~2) pin is detected. CNT is reset to 0.
	0	1	11	Reserved
TriggerCounting Capture Mode	1	0	00	Falling Edge Trigger: The 1st 1 to 0 transition on TMRx_EXT (x= 1~2) pin is detected to reset CNT as 0 and then starts counting, while the 2nd 1 to 0 transition stops counting.
	1	0	01	Rising Edge Trigger: The 1st 0 to 1 transition to on TMRx_EXT (x= 1~2) pin is detected to reset CNT as 0 and then starts counting, while the 2nd 0 to 1 transition stops counting.
	1	0	10	Either edge Trigger: An 1 to 0 transition on TMx_EXT (x= 1~2) pin is detected to reset CNT as 0 and then starts counting, while 0 to 1 transition stops counting.
	1	0	11	Either edge Trigger: A 0 to 1 transition on TMx_EXT (x= 1~2) pin is detected to reset CNT as 0 and then starts counting, while 1 to 0 transition stops counting.

3.17.6 TMR Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
TRM Base Address: TMR1_BA = 0x4001_4000 TMR2_BA = 0x4001_5000				
TIMER1_CTL	TMR1_BA+0x00	R/W	Timer1 Control and Status Register	0x0000_0005
TIMER1_CMP	TMR1_BA+0x04	R/W	Timer1 Compare Register	0x0000_0000
TIMER1_INTSTS	TMR1_BA+0x08	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER1_CNT	TMR1_BA+0x0C	R	Timer1 Data Register	0x0000_0000
TIMER1_CAP	TMR1_BA+0x10	R	Timer1 Capture Data Register	0x0000_0000
TIMER1_EXTCTL	TMR1_BA+0x14	R/W	Timer1 External Control Register	0x0000_0000
TIMER1_EINTSTS	TMR1_BA+0x18	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER2_CTL	TMR2_BA+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TIMER2_CMP	TMR2_BA+0x04	R/W	Timer2 Compare Register	0x0000_0000
TIMER2_INTSTS	TMR2_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER2_CNT	TMR2_BA+0x0C	R	Timer2 Data Register	0x0000_0000
TIMER2_CAP	TMR2_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER2_EXTCTL	TMR2_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER2_EINTSTS	TMR2_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000

3.17.7 TMR Register Description

3.17.7.1 Timer Control Register (CTL)

Register	Offset	R/W	Description	Reset Value
TIMER1_CTL	TMR1_BA+0x00	R/W	Timer1 Control and Status Register	0x0000_0005
TIMER2_CTL	TMR2_BA+0x00	R/W	Timer2 Control and Status Register	0x0000_0005

Bits	Descriptions	
[31]	ICEDEBUG	ICE Debug Mode Acknowledge Disable Bit (Write Protect) 0 = ICE debug mode acknowledgement effects TIMER counting. TIMER counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. TIMER counter will keep going no matter CPU is held by ICE or not. **Note**: This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	CNTEN	Timer Counting Enable Bit 0 = Stops/Suspends counting. 1 = Starts counting. **Note1**: In stop status, and then setting CNTEN to 1 will enable the 24-bit up counter to keep counting from the last stop counting value. **Note2**: This bit is auto-cleared by hardware in one-shot mode (CTL[28:27] = 2'b00) when the timer interrupt flag TIF (INTSTS[0]) is generated.
[29]	INTEN	Timer Interrupt Enable Bit 0 = Timer Interrupt Disabled. 1 = Timer Interrupt Enabled. **Note**: If this bit is enabled, when the timer interrupt flag TIF is set to 1, the timer interrupt signal will be generated and inform CPU.
[28:27]	OPMODE	Timer Counting Mode Selection 00 = The Timer controller is operated in one-shot mode. 01 = The Timer controller is operated in periodic mode. 10 = The Timer controller is operated in toggle-output mode. 11 = The Timer controller is operated in continuous counting mode.
[26]	RSTCNT	Timer Counter Reset Setting this bit will reset the 24-bit up counter value CNT (CNT[23:0]) and also force CNTEN (CTL[30]) to 0 if ACTSTS (CTL[25]) is 1. 0 = No effect. 1 = Reset internal 8-bit prescale counter, 24-bit up counter value and CNTEN bit.
[25]	ACTSTS	Timer Active Status (Read Only) This bit indicates the 24-bit up counter status. 0 = 24-bit up counter is not active. 1 = 24-bit up counter is active.
[24]	EXTCNTEN	Event Counter Mode Enable Bit This bit is for external counting pin function enabled. 0 = Event counter mode Disabled. 1 = Event counter mode Enabled. **Note**: When timer is used as an event counter, this bit should be set to 1 and select APB_CLK as timer clock source

[23]	WKEN	Wake-up Function Enable Bit If this bit is set to 1, while the timer interrupt flag TIF (INTSTS[0]) is 1 and INTEN (CTL[29]) is enabled, the timer interrupt signal will generate a wake-up trigger event to CPU. 0 = Wake-up function Disabled if timer interrupt signal generated. 1 = Wake-up function Enabled if timer interrupt signal generated.
[22:20]	Reserved	Reserved.
[19]	CAPSRC	Capture Pin Source Select Bit 0 = Capture source is from TMRx_EXT (x= 1~2) pin. 1 = Capture source is from internal 32KHz clock source, that is RCL or XTL.
[18:8]	Reserved	Reserved.
[7:0]	PSC	Prescale Counter Timer input clock or event source is divided by (PSC+1) before it is fed to the timer up counter. If this field is 0 (PSC = 0), then there is no scaling.

3.17.7.2 Timer Compare Register (CMP)

Register	Offset	R/W	Description	Reset Value
TIMER1_CMP	TMR1_BA+0x04	R/W	Timer1 Compare Register	0x0000_0000
TIMER2_CMP	TMR2_BA+0x04	R/W	Timer2 Compare Register	0x0000_0000

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[23:0]	CMPDAT	Timer Compared Value CMPDAT is a 24-bit compared value register. When the internal 24-bit up counter value is equal to CMPDAT value, the TF (INTSTS[2]) and TIF (INTSTS[0]) will set to 1. Time-out period = (Period of timer clock input) * (8-bit PSC + 1) * (24-bit CMPDAT). **Note1**: Never write 0x0 or 0x1 in CMPDAT field, or the core will run into unknown state. **Note2**: When timer is operating at continuous counting mode, the 24-bit up counter will keep counting continuously even if user writes a new value into CMPDAT field. But if timer is operating at other modes, the 24-bit up counter will restart counting from 0 and using newest CMPDAT value to be the timer compared value while user writes a new value into the CMPDAT field.

3.17.7.3 Timer Interrupt Status Register (INTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER1_INTSTS	TMR1_BA+0x08	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER2_INTSTS	TMR2_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2]	TF	Timer Flag This bit indicates the interrupt flag status of Timer while 24-bit timer up counter CNT (CNT[23:0]) value reaches to CMPDAT (CMP[23:0]) value. 0 = No effect. 1 = CNT value matches the CMPDAT value. **Note**: This bit is cleared by writing 1 to it.
[1]	TWKF	Timer Wake-up Flag This bit indicates the interrupt wake-up flag status of timer. 0 = Timer does not cause CPU wake-up. 1 = CPU wake-up from Idle or Power-down mode if timer time-out interrupt signal generated. **Note**: This bit is cleared by writing 1 to it.
[0]	TIF	Timer Interrupt Flag This bit indicates the interrupt flag status of Timer while 24-bit timer up counter CNT (CNT[23:0]) value reaches to CMPDAT (CMP[23:0]) value. 0 = No effect. 1 = CNT value matches the CMPDAT value. **Note**: This bit is cleared by writing 1 to it.

3.17.7.4 Timer Data Register (TIMERx_CNT)

Register	Offset	R/W	Description	Reset Value
TIMER1_CNT	TMR1_BA+0x0C	R	Timer1 Data Register	0x0000_0000
TIMER2_CNT	TMR2_BA+0x0C	R	Timer2 Data Register	0x0000_0000

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[23:0]	CNT	Timer Data Register This field reflect the count value from internal 32KHz clock or external event from TMRx (x=0~2) pin which is set in.EXTCNTEN field.

3.17.7.5 Timer Capture Data Register (TIMERx_CAP)

Register	Offset	R/W	Description	Reset Value
TIMER1_CAP	TMR1_BA+0x10	R	Timer1 Capture Data Register	0x0000_0000
TIMER2_CAP	TMR2_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[23:0]	CAPDAT	Timer Capture Data Register When *CAPEN* (EXTCTL[3]) bit is set, and a transition on TMRx_EXT pin or internal 32KHz clock matched the *CAPEDGE* (EXTCTL[2:1]) setting, *CAPIF* (EINTSTS[0]) will set to 1 and the current timer counter value *CNT* (CNT[23:0]) will be auto-loaded into this CAPDAT field.

3.17.7.6 Timer External Control Register (EXTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER1_EXTCTL	TMR1_BA+0x14	R/W	Timer1 External Control Register	0x0000_0000
TIMER2_EXTCTL	TMR2_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000

Bits	Descriptions	
[31:9]	Reserved	Reserved.
[8]	CAPSEL	Capture Mode Select Bit 0 = set for free-counting capture mode or external reset counter mode of timer capture function. 1 = set for trigger-counting mode of timer capture function.
[7]	CNTDBEN	Timer Counter Pin De-bounce Enable Bit 0 = de-bounce disable 1 = de-bounce enable **Note**: If this bit is enabled, the edge detection of TMRx_EXT pin or internal 32KHz clock is detected with de-bounce circuit.
[6]	CAPDBEN	Timer External Capture Pin De-bounce Enable Bit 0 = de-bounce disable 1 = de-bounce enable **Note1**: If this bit is enabled, the edge detection of TMRx_EXT pin or internal 32KHz clock is detected with de-bounce circuit.
[5]	CAPIEN	Timer External Capture Interrupt Enable Bit 0 = disable 1 = enable **Note**: If CAPIEN enabled, timer will generate an interrupt when CAPIF (EINTSTS[0]) is high. For example, while CAPIEN = 1, CAPEN = 1, and CAPEDGE = 00, an 1 to 0 transition on the TMRx_EXT pin or internal 32KHz clock will cause the CAPIF to be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	CAPFUNCS	Capture Function Select Bit 0 = external reset counter mode disabled 1 = external reset counter mode enabled. **Note1**: When CAPFUNCS is 0, transition on TMRx_EXT (x= 1~2) pin is used to save the 24-bit timer counter value to CAPDAT register. **Note2**: When CAPFUNCS is 1, transition on TMx_EXT (x= 1~2) pin is used to save the 24-bit timer counter value to CAPDAT register and then reset the 24-bit timer counter value.
[3]	CAPEN	Timer Capture Enable Bit 0 = disable 1 = enable
[2:1]	CAPEDGE	Timer Capture Edge Detection: 00 = A falling edge on TMRx_EXT pin or internal 32KHz clock will be detected. 01 = A rising edge on TMRx_EXT pin or internal 32KHz clock will be detected. 10 = Either rising or falling edge on TMRx_EXT pin or internal 32KHz clock will be detected. 11 = Reserved.

[0]	CNTPHASE	Event counting mode setting This bit indicates the detection of expected transition of TMRx (x= 1~2) pin 0 = A falling edge of pin will be counted and loaded to CNT value 1 = A rising edge of pin will be counted and loaded to CNT value

3.17.7.7 Timer External Interrupt Status Register (TIMERx_EINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER1_EINTSTS	TMR1_BA+0x18	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER2_EINTSTS	TMR2_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000

Bits	Descriptions	
[31:2]	Reserved	Reserved.
[1]	CAPF	Timer External Capture Flag This bit indicates the timer external capture interrupt flag status. **Note1**: This bit is cleared by writing 1 to it.
[0]	CAPIF	Timer External Capture Interrupt Flag This bit indicates the timer external capture interrupt flag status. **Note1**: Need set CAPIEN to 1. This bit is cleared by writing 1 to it.

3.18 Timer Controller (TMR0)

3.18.1 Overview

The PAN101x chip includes three same timer modules, TMR0, TMR1 and TMR2, allowing user to easily implement a timer control for applications. TMR0 locates in apb1 subsystem, while TMR1 and TMR2 locate in apb2 subsystem. The timer can perform functions, such as frequency measurement, delay timing, clock generation, event counting by external input pins, and interval measurement by external capture pins.

3.18.2 Features

- Each set of timer equipped with 32-bit up counter and one 8-bit prescale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle-output and continuous counting operation modes.
- 32-bit up counter value is readable through CNT
- Supports event counting function
- 32-bit capture value is readable through CAPDAT
- Supports interval measurement for external capture pin event or 32KHz clock source
- Supports external capture pin event to reset 32-bit up counter
- Supports chip wake-up when timer generated wake up interrupt signal
- Supports multiple comparators (only in continuous counting mode)

3.18.3 Block Diagram

The timer controller block diagram and clock control are shown in Figure 3-126, Figure 3-127 and Figure 3-128.

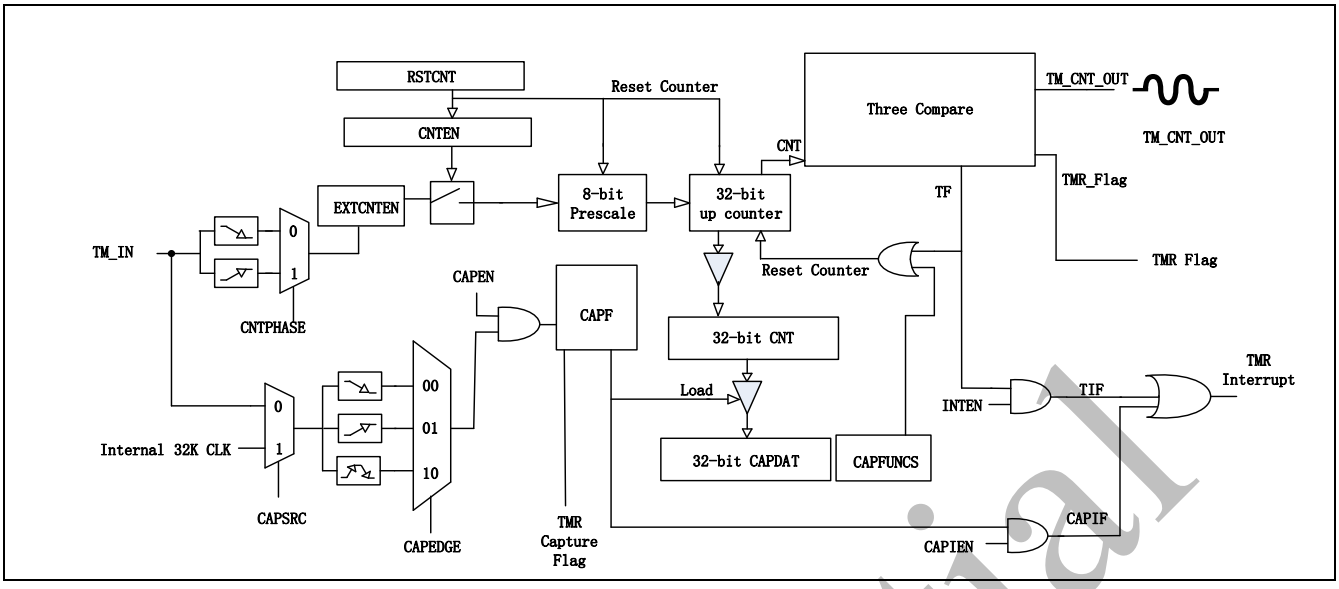


Figure 3-126 Timer Controller Block Diagram

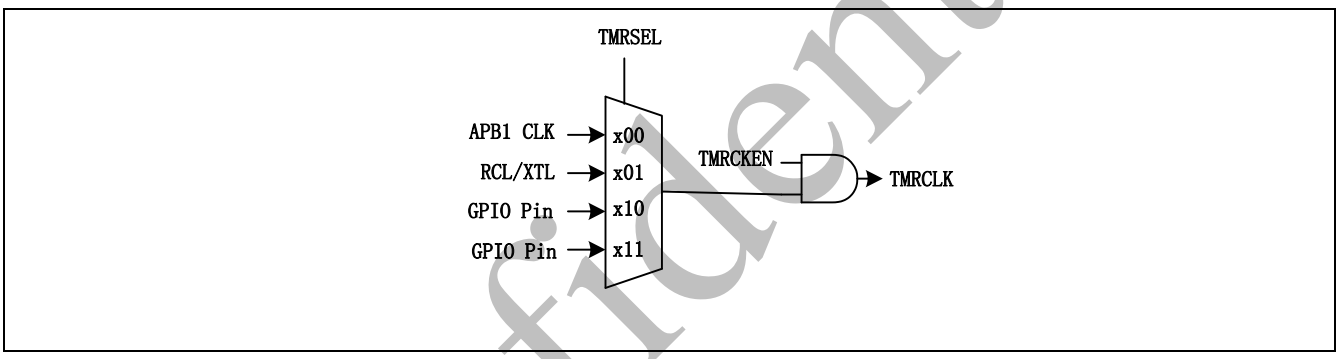


Figure 3-127 Clock Source of Timer Controller

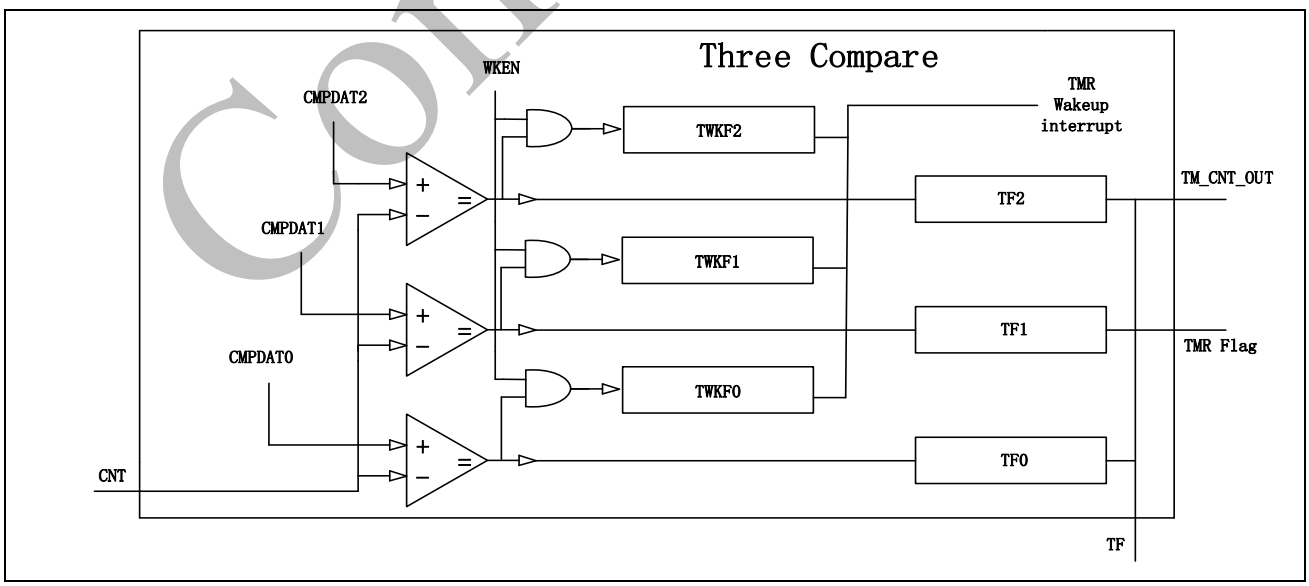


Figure 3-128 Three Compare

3.18.4 Basic Configuration

3.18.4.1 Clock Source Setting

You can choose from APB_CLK, RCL/XTL or TM_I. APB_CLK is APB subsystem clock source. RCL/XTL refers to 32kHz clock source. TM_I is input from GPIO pin, showed in Figure 3-127.

3.18.4.2 Timer Flag

The timer controller supports two flags: one is TF which is set while timer counter value CNT matches the timer compared value CMPDAT, and the other is CAPF which is set when the transition on the TMx_EXT pin is consistent with CAPEDGE setting.

3.18.4.3 Timer Interrupt Flag

The timer controller supports two interrupt flags:

One is TIF which is set while timer counter value CNT (cntmatches the timer compared value CMPDAT and INTEN is set to 1.

The other is CAPIF which is set when the transition on the TMR_EXT pin is in accord with CAPEDGE setting, and CAPIEN is set to 1.

The only difference between timer flag and timer interrupt flag is that whether interrupt enable signal is set or not.

3.18.5 Functional Description

3.18.5.1 Timer Counting Operation Mode

The Timer controller provides four timer counting modes: One-shot, Periodic, Toggle-output and Continuous Counting operation modes as described below.

One-shot Mode

If the timer controller is configured at one-shot mode OPMODE and CNTEN is set, the timer counter starts up counting. Once the CNT value reaches CMPDAT value, the TF will be set to 1, CNT value and CNTEN bit is cleared automatically by timer controller then timer counting operation stops. In the meantime, if the INTEN is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

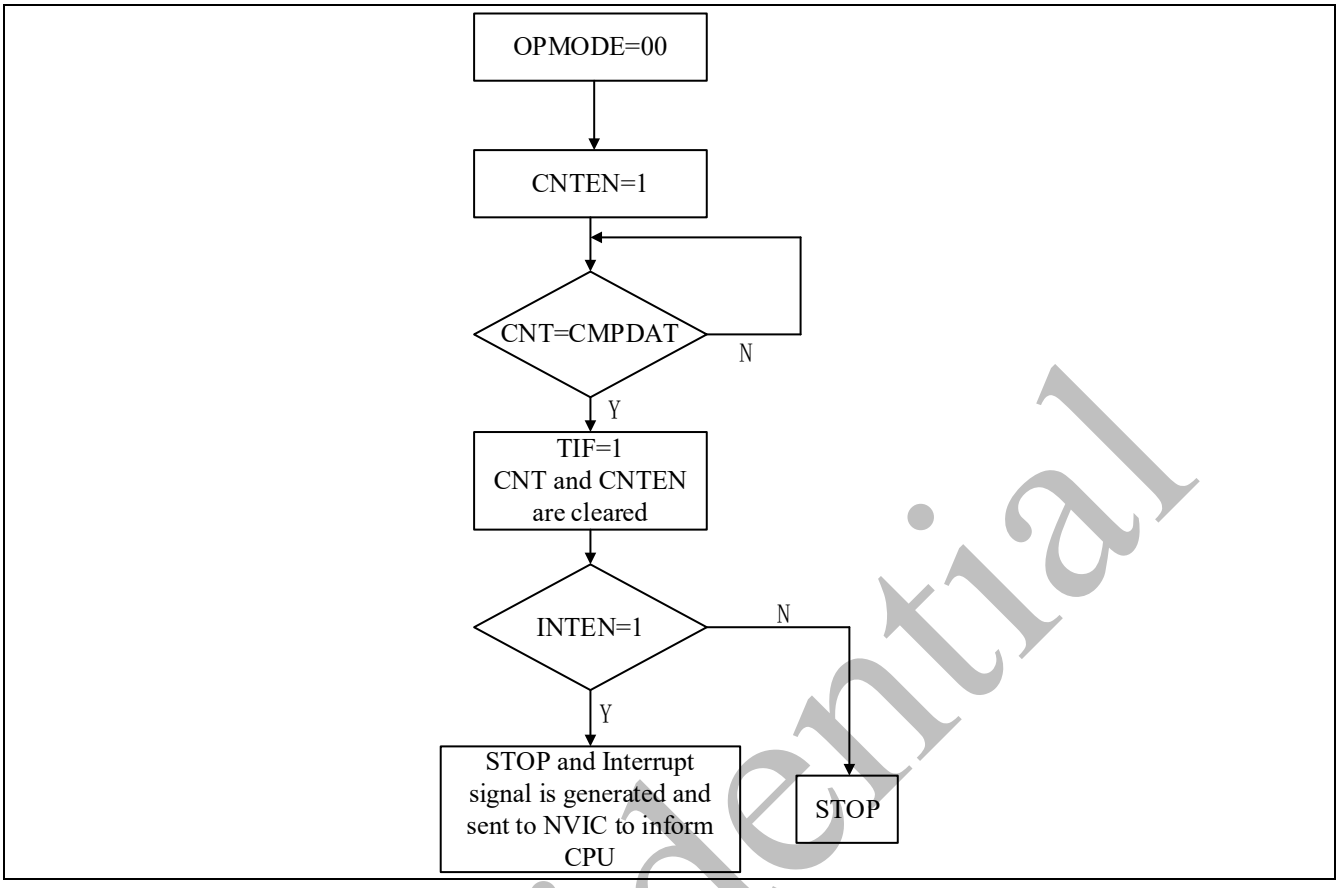


Figure 3-129 One-shot Mode

Periodic Mode

If the timer controller is configured at periodic mode and CNTEN is set, the timer counter starts up counting. Once the CNT value reaches CMPDAT value, the TF will be set to 1, CNT value will be cleared automatically by timer controller and timer counter operates counting again. In the meantime, if the INTEN bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. In this mode, the timer controller operates counting and compares with CMPDAT value periodically until the CNTEN bit is cleared by user.

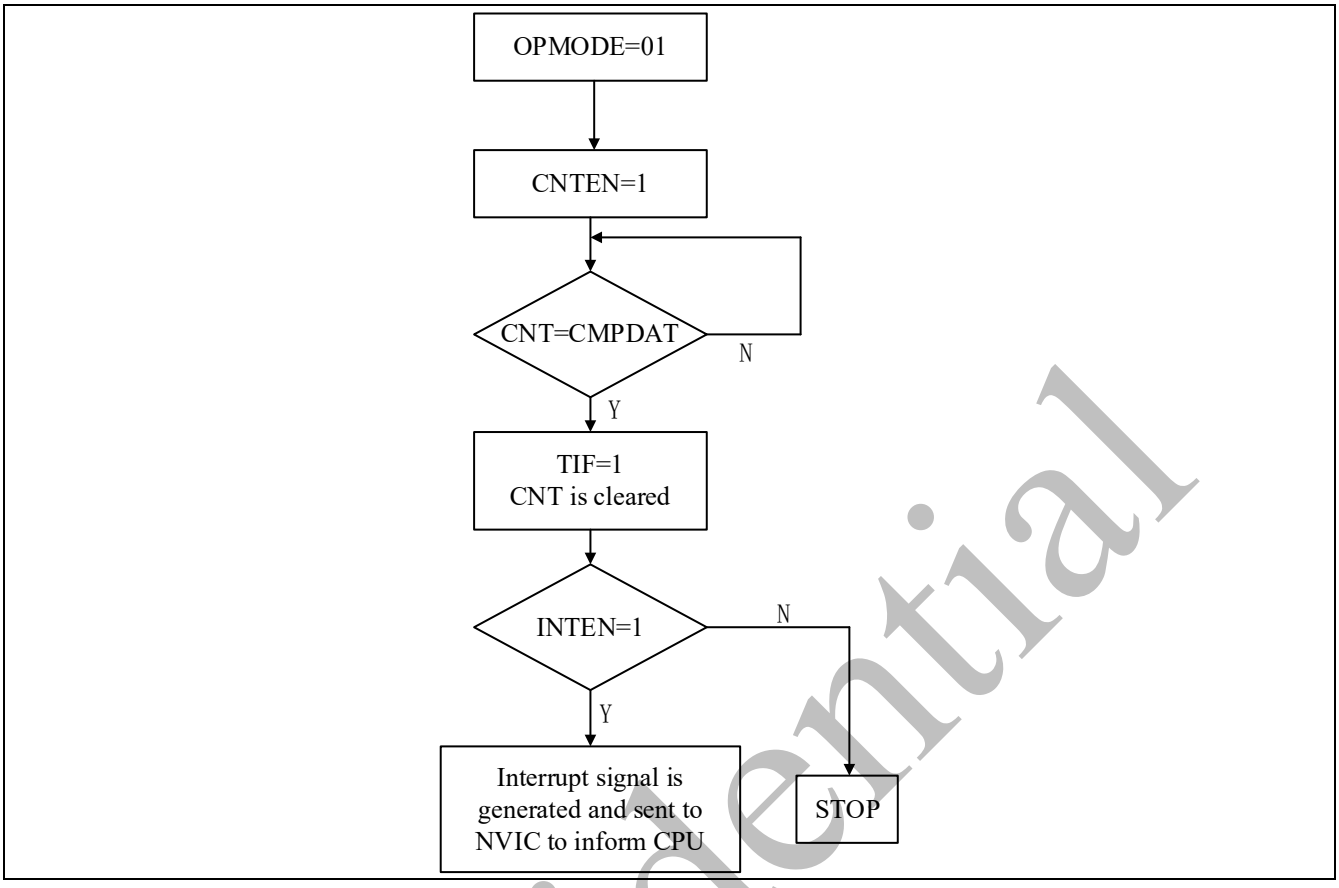


Figure 3-130 Periodic Mode

Toggle-output Mode

If the timer controller is configured at toggle-output mode and CNTEN is set, the timer counter starts up counting. The counting operation of toggle-output mode is almost the same as periodic mode, except toggle-output mode has associated TMR0 ~ TMR2 pin to output signal while specify TIF is set. So in this mode, INTEN must be set. Normally, the toggle-output signal on TMR_CNT_OUT pin is high and changing back and forth with 50% duty cycle. For toggle-out output PULSE mode, waveforms of different duties can be achieved by adjusting the CMPDATA value in the interrupt program.

Continuous Counting Mode

If the timer controller is configured at continuous counting mode and is set, the timer counter starts up counting. Once the CNT value reaches the CMPDAT value, the TF will be set to 1 and the CNT value keeps up counting until reach 2^32-1. In the meantime, if the INTEN is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU. User can change different CMPDAT value immediately without disabling timer counting and restarting timer counting in this mode.

For example, CMPDAT value is set as 80, first. The TF will set to 1 when CNT value is equal

to 80, the timer counter is kept counting and CNT value will not go back to 0, it continues to count 81, 82, 83,…to 2^32-1, 0, 1, 2, 3, …to 2^32-1 again and again. Next, if user programs the CMPDAT value as 200 and clears TF, the TF will be set to 1 again when CNT value reaches 200. At last, user programs CMPDAT as 500 and clears TF, the TF will set to 1 again when CNT value reaches to 500. In this mode, the timer counting is continuous. You can update CMPDAT continuously when system detected TF or TIF flag or update CMPDAT value in your ISR function.

The continuous mode in this TIMER module supports three compare registers (CMP0/1/2) to work at the same time. For TIMER one-shot, periodic and toggle-out counting modes, although there are three CMPDATA, these two counting methods will automatically clear the CNT counter after counting to the target value of CMPDATA, so only one CMPDATA is actually valid. Especially the toggle-out mode requires interrupt enable to output pulse, so the corresponding mask_tifx must be set to 1.

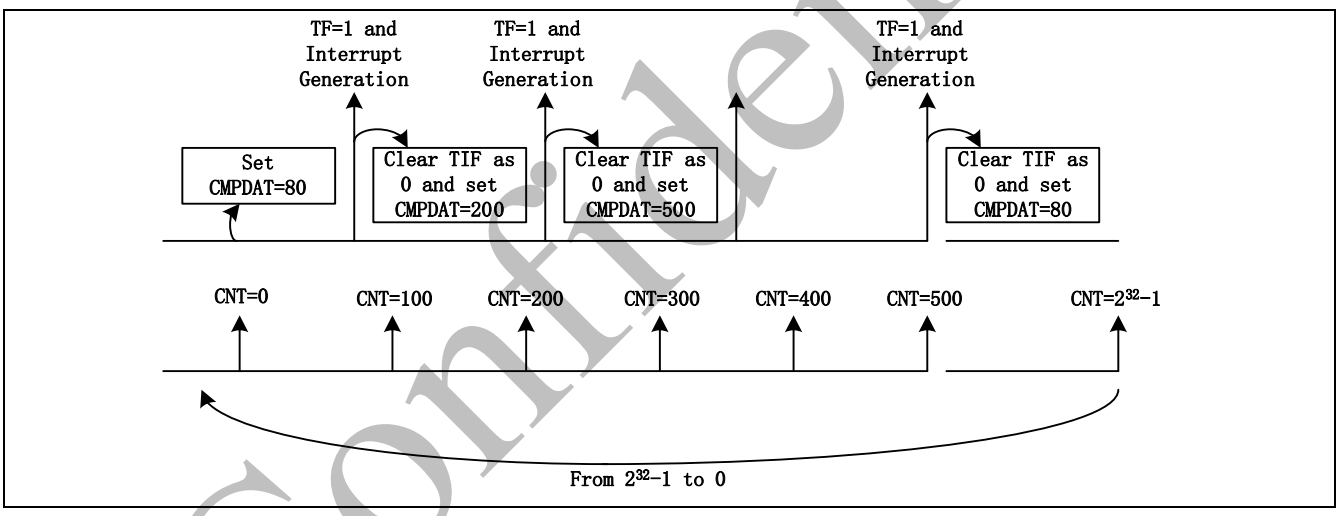


Figure 3-131 Continuous Counting Mode

3.18.5.2 Event Counting Mode

The timer controller also provides an application which can count the input event from GPIO input pin..The number of event will reflect to CNT value. It is also called as event counting function. In this function, EXTCNTEN should be set and the timer peripheral clock source should be set as APB_CLK or 32kHz.

User can enable or disable TMR pin de-bounce circuit by setting CNTDBEN. The input event frequency should be less than 1/3 APB_CLK if TM pin de-bounce disabled or less than 1/8 APB_CLK if TMR pin de-bounce enabled to assure the returned CNT value is correct, and user can also select edge detection phase of TMR pin by setting CNTPHASE bit.

In event counting mode, the timer counting operation mode can be selected as one-shot, periodic and continuous counting mode to counts the counter value CNT from TMR pin..

3.18.5.3 Input Capture Function

The input capture or reset function is provided to capture or reset timer counter value. The capture function with free-counting capture mode and trigger-counting capture mode are configured by CAPSEL. The free-counting capture mode, external reset counter mode, trigger-counting capture mode are described as follows. Input capture function can both capture TMR_EXT GPIO input pin signal and internal 32KHz clock, which can be set in CAPSRC register. Following operation modes set TMR_EXT pin input signal as instruction.

Free-Counting Capture Mode

The event capture function is used to load CNT value to CAPDAT value while edge transition detected on TMR_EXT pin. In this mode, CAPSEL and CAPFUNCS should be as 0. The expected transition on TMR_EXT pin will trigger event capture function. The timer peripheral clock source should be set as APB_CLK.

User can enable or disable TMR_EXT pin de-bounce circuit by setting CAPDBEN. The transition frequency of TMR_EXT pin should be less than 1/3 APB_CLK if TMR_EXT pin de-bounce disabled or less than 1/8 APB_CLK if TMR_EXT pin de-bounce enabled to assure the capture function can be work normally, and user can also select edge transition detection of TMR_EXT pin by setting CAPEDGE.

In event capture mode, user does not consider what timer counting operation mode is selected, the capture event occurred only if edge transition on TMR_EXT pin is detected.

Users must consider the Timer will keep register CAP unchanged and drop the new capture value, if the CPU does not clear the CAPIF status. The operation method is described in Figure 3-132.

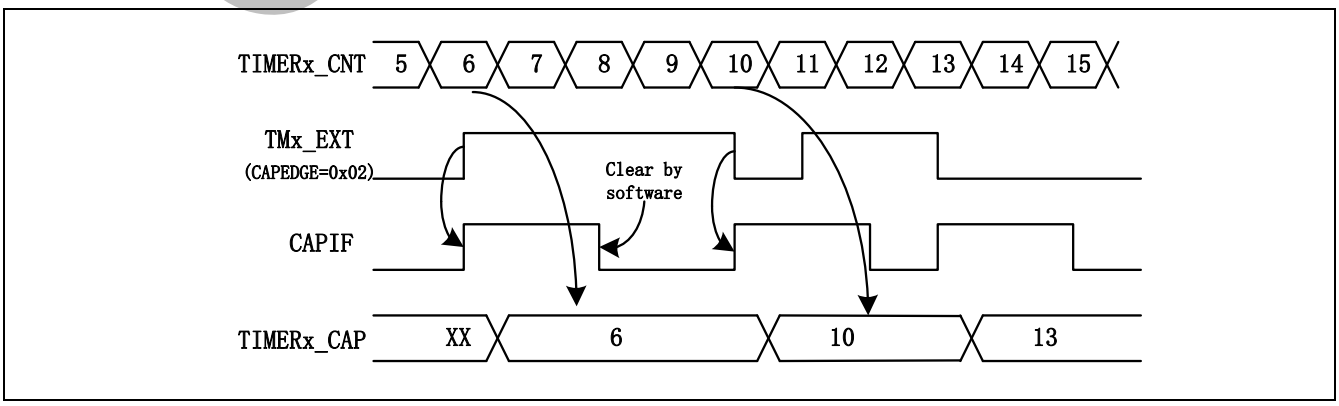


Figure 3-132 Free-Counting Capture Mode

External Reset Counter Mode

The timer controller also provides reset counter function to reset CNT value while edge transition detected on TMR_EXT. In this mode, most the settings are the same as event capture mode except CAPFUNCS should be as 1 for select TMR_EXT transition is using to trigger reset counter value. The operation method is also described in Figure 3-133. The difference between Free-Counting Capture Mode and External Reset Counter Mode is whether the TMR_CNT value will be cleared or not.

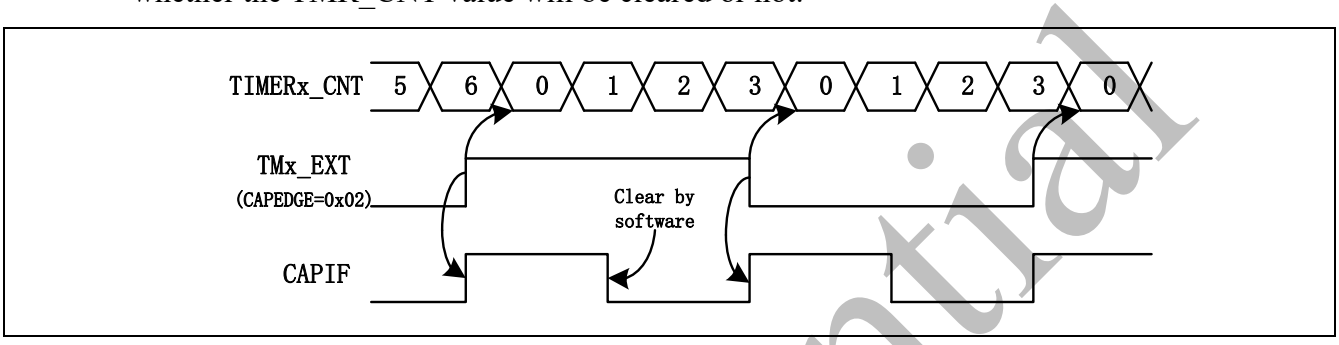


Figure 3-133 External Reset Counter Mode

Trigger-Counting Capture Mode

If CAPSEL is set to 1, CAPEN is set to 1 and CAPFUNCS is set to 0, the CNT will be reset to 0 then captured into CAPDAT register when TMR_EXT pin trigger condition occurred. The TMR_EXT expected edge can be chosen by CAPEDGE. The detailed operation method is described in Table 3-23.

Firstly when TMR_EXT expected edge occurs, the TMR_CNT will start count from zero. Secondly TMR_EXT expected edge reoccur, CAPF will set to 1, and the interrupt signal is generated if CAPIEN is enabled. Once CAPIF is detected, you need disable CAPIEN in ISR function. The TMR_EXT source operating frequency should be less than 1/3 APB_CLK frequency if disable TMR_EXT de-bounce or less than 1/8 APB_CLK frequency if enabling TMR_EXT de-bounce. It also provides TMR_EXT enabled or disabled capture de-bounce function by CAPDBEN.

Table 3-23 Input Capture Mode Operation

Function	CAPSEL	CAPFUNCS	CAPEDGE	Operation Description
Free-countig Capture Mode	0	0	00	A 1 to 0 transition on TMR_EXT pin is detected. CNT is captured to CAPDAT.
	0	0	01	A 0 to 1 transition on TMR_EXT pin is detected. CNT is captured to CAPDAT.
	0	0	10	Either 1 to 0 or 0 to 1 transition on TMR_EXT pin is detected. CNT is captured to CAPDAT.

	0	0	11	Reserved
External Reset Counter Mode	0	1	00	A 1 to 0 transition on TMR_EXT pin is detected. CNT is reset to 0.
	0	1	01	A 0 to 1 transition on TMR_EXT pin is detected. CNT is reset to 0.
	0	1	10	Either 1 to 0 or 0 to 1 transition on TMR_EXT pin is detected. CNT is reset to 0.
	0	1	11	Reserved
Trigger Counting Capture Mode	1	0	00	Falling Edge Trigger: The 1st 1 to 0 transition on TMR_EXT pin is detected to reset CNT as 0 and then starts counting, while the 2nd 1 to 0 transition stops counting.
	1	0	01	Rising Edge Trigger: The 1st 0 to 1 transition to on TMR_EXT pin is detected to reset CNT as 0 and then starts counting, while the 2nd 0 to 1 transition stops counting.
	1	0	10	Either edge Trigger: An 1 to 0 transition on TM_EXT pin is detected to reset CNT as 0 and then starts counting, while 0 to 1 transition stops counting.
	1	0	11	Either edge Trigger: A 0 to 1 transition on TM_EXT pin is detected to reset CNT as 0 and then starts counting, while 1 to 0 transition stops counting.

3.18.6 TMR Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
TRM Base Address: TMR_BA = 0x4008_0000				
TIMER_CTL	TMR_BA+0x00	R/W	Timer Control and Status Register	0x0000_0005
TIMER_CMP0	TMR_BA+0x04	R/W	Timer First Compare Register	0x0000_0000
TIMER_INTSTS	TMR_BA+0x08	R/W	Timer Interrupt Status Register	0x0000_0000
TIMER_CNT	TMR_BA+0x0C	R	Timer Data Register	0x0000_0000
TIMER_CAP	TMR_BA+0x10	R	Timer Capture Data Register	0x0000_0000
TIMER_EXTCTL	TMR_BA+0x14	R/W	Timer External Control Register	0x0000_0000
TIMER_EINTSTS	TMR_BA+0x18	R/W	Timer External Interrupt Status Register	0x0000_0000
TIMER_CMP1	TMR_BA+0x1C	R/W	Timer Second Compare Register	0x0000_0000
TIMER_CMP2	TMR_BA+0x20	R/W	Timer Third Compare Register	0x0000_0000

3.18.7 TMR Register Description

3.18.7.1 Timer Control Register (CTL)

Register	Offset	R/W	Description	Reset Value
TIMER_CTL	TMR0_BA+0x00	R/W	Timer Control and Status Register	0x0000_0005

Bits	Descriptions	
[31]	ICEDEBUG	ICE Debug Mode Acknowledge Disable Bit (Write Protect) 0 = ICE debug mode acknowledgement effects TIMER counting. TIMER counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. TIMER counter will keep going no matter CPU is held by ICE or not. **Note**: This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	CNTEN	Timer Counting Enable Bit 0 = Stops/Suspends counting. 1 = Starts counting. **Note1**: In stop status, and then setting CNTEN to 1 will enable the 32-bit up counter tokeep counting from the last stop counting value. **Note2**: This bit is auto-cleared by hardware in one-shot mode when the timer interruptflag TIF is generated.
[29]	INTEN	Timer Interrupt Enable Bit 0 = Timer Interrupt Disabled. 1 = Timer Interrupt Enabled. **Note**: If this bit is enabled, when the timer interrupt flag TIF is set to 1, the timer interrupt signal will be generated and inform CPU.
[28:27]	OPMODE	Timer Counting Mode Selection 00 = The Timer controller is operated in one-shot mode. 01 = The Timer controller is operated in periodic mode. 10 = The Timer controller is operated in toggle-output mode. 11 = The Timer controller is operated in continuous counting mode.
[26]	RSTCNT	Timer Counter Reset Setting this bit will reset the 32-bit up counter value CNT and also force CNTEN to 0 if ACTSTS is 1. 0 = No effect. 1 = Reset internal 8-bit prescale counter, 32-bit up counter value and CNTEN bit.
[25]	ACTSTS	Timer Active Status (Read Only) This bit indicates the 32-bit up counter status. 0 = 32-bit up counter is not active. 1 = 32-bit up counter is active.
[24]	EXTCNTEN	Event Counter Mode Enable Bit This bit is for external counting pin function enabled. 0 = Event counter mode Disabled. 1 = Event counter mode Enabled. **Note**: When timer is used as an event counter, this bit should be set to 1 and select APB_CLK as timer clock source.
[23]	WKEN	Wake-up Function Enable Bit.

		If this bit is set to 1, while the timer interrupt flag TIF is 1 and INTEN is enabled, the timer interrupt signal will generate a wake-up trigger event to CPU. 0 = Wake-up function Disabled if timer interrupt signal generated. 1 = Wake-up function Enabled if timer interrupt signal generated.
[22:20]	Reserved	Reserved.
[19]	CAPSRC	Capture Pin Source Select Bit 0 = Capture source is from TMR_EXT pin. 1 = no use.
[18:14]	Reserved	Reserved.
[13]	MASK_WK2	Wakeup Flag2 Signal Enable When MASK_WK2 = 1 and the wake-up flag2 = 1, the wake-up flag2 can be output.
[12]	MASK_WK1	Wakeup Flag1 Signal Enable When MASK_WK1 = 1 and the wake-up flag1 = 1, the wake-up flag1 can be output.
[11]	MASK_WK0	Wakeup Flag0 Signal Enable When MASK_WK0 = 1 and the wake-up flag0 = 1, the wake-up flag0 can be output.
[10]	MASK_TIF2	Timer interrupt flag2 signal enable When MASK_TIF2 = 1 and Timer interrupt flag2 = 1, the Timer interrupt flag2 signal can be output.
[9]	MASK_TIF1	Timer interrupt flag1 signal enable When MASK_TIF1 = 1 and Timer interrupt flag1 = 1, the Timer interrupt flag1 signal can be output.
[8]	MASK_TIF0	Timer interrupt flag0 signal enable When MASK_TIF0 = 1 and Timer interrupt flag0 = 1, the Timer interrupt flag0 signal can be output.
[7:0]	PSC	Prescale Counter Timer input clock or event source is divided by (PSC+1) before it is fed to the timer up counter. If this field is 0 (PSC = 0), then there is no scaling.

3.18.7.2 Timer Compare Register (CMP)

Register	Offset	R/W	Description	Reset Value
TIMER_CMP0	TMR_BA+0x04	R/W	Timer First Compare Register	0x0000_0000
TIMER_CMP1	TMR_BA+0x1C	R/W	Timer Second Compare Register	0x0000_0000
TIMER_CMP2	TMR_BA+0x20	R/W	Timer Third Compare Register	0x0000_0000

Bits	Descriptions	
[31:0]	CMPDAT	Timer Compared Value CMPDAT is a 32-bit compared value register. When the internal 32-bit up counter value is equal to CMPDAT value, the TF and TIF will set to 1. Time-out period = (Period of timer clock input) * (8-bit PSC + 1) * (32-bit CMPDAT). **Note1:** Never write 0x0 or 0x1 in CMPDAT field, or the core will run into unknown state. **Note2:** When timer is operating at continuous counting mode, the 32-bit up counter will keep counting continuously even if user writes a new value into CMPDAT field. But if timer is operating at other modes, the 32-bit up counter will restart counting from 0 and using newest CMPDAT value to be the timer compared value while user writes a new value into the CMPDAT field.

3.18.7.3 Timer Interrupt Status Register (INTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER_INTSTS	TMR_BA+0x08	R/W	Timer Interrupt Status Register	0x0000_0000

Bits	Descriptions	
[31:7]	Reserved	Reserved.
[6]	TF2	Timer Flag This bit indicates the interrupt flag status of Timer while 32-bit timer up counter CNT value reaches to CMPDAT2 value. 0 = No effect. 1 = CNT value matches the CMPDAT2 value. **Note**: This bit is cleared by writing 1 to it.
[5]	TF1	Timer Flag This bit indicates the interrupt flag status of Timer while 32-bit timer up counter CNT value reaches to CMPDAT1 value. 0 = No effect. 1 = CNT value matches the CMPDAT 1value. **Note**: This bit is cleared by writing 1 to it.
[4]	TF0	Timer Flag This bit indicates the interrupt flag status of Timer while 32-bit timer up counter CNT value reaches to CMPDAT0 value. 0 = No effect. 1 = CNT value matches the CMPDAT 0value. **Note**: This bit is cleared by writing 1 to it.
[3]	TWKF2	Timer Wake-up Flag2 This bit indicates the interrupt wake-up flag status of timer. 0 = Timer does not cause CPU wake-up. 1 = CPU wake-up from Idle or Power-down mode if timer time-out interrupt signal generated. **Note**: This bit is cleared by writing 1 to it.
[2]	TWKF1	Timer Wake-up Flag1 This bit indicates the interrupt wake-up flag status of timer. 0 = Timer does not cause CPU wake-up. 1 = CPU wake-up from Idle or Power-down mode if timer time-out interrupt signal generated. **Note**: This bit is cleared by writing 1 to it.
[1]	TWKF0	Timer Wake-up Flag0 This bit indicates the interrupt wake-up flag status of timer. 0 = Timer does not cause CPU wake-up. 1 = CPU wake-up from Idle or Power-down mode if timer time-out interrupt signal generated. **Note**: This bit is cleared by writing 1 to it.
[0]	TIF	Timer Interrupt Flag This bit indicates the interrupt flag status of Timer while 32-bit timer up counter CNT value reaches to CMPDAT value.

		0 = No effect. 1 = CNT value matches the CMPDAT value. **Note**: This bit is cleared by writing 1 to it.

3.18.7.4 Timer Data Register (TIMER_CNT)

Register	Offset	R/W	Description	Reset Value
TIMER_CNT	TMR_BA+0x0C	R	Timer Data Register	0x0000_0000

Bits	Descriptions	
[31:0]	CNT	Timer Data Register This field reflect the count value from internal 32kHz clock or external event from TMR pin which is set in EXTCNTEN field.

3.18.7.5 Timer Capture Data Register (TIMER_CAP)

Register	Offset	R/W	Description	Reset Value
TIMER_CAP	TMR_BA+0x10	R	Timer Capture Data Register	0x0000_0000

Bits	Descriptions	
[31:0]	CAPDAT	Timer Capture Data Register When CAPEN bit is set, and a transition on TMR_EXT pin or internal 32kHz clock matched the CAPEDGE setting, CAPIF will set to 1 and the current timer counter value CNT will be auto-loaded into this CAPDAT field.

3.18.7.6 Timer External Control Register (EXTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER_EXTCTL	TMR_BA+0x14	R/W	Timer External Control Register	0x0000_0000

Bits	Descriptions	
[31:9]	Reserved	Reserved.
[8]	CAPSEL	Capture Mode Select Bit 0 = set for free-counting capture mode or external reset counter mode of timer capture function. 1 = set for trigger-counting mode of timer capture function.
[7]	CNTDBEN	Timer Counter Pin De-bounce Enable Bit 0 = de-bounce disable 1 = de-bounce enable **Note**: If this bit is enabled, the edge detection of TMR_EXT pin or internal 32kHz clock is detected with de-bounce circuit.
[6]	CAPDBEN	Timer External Capture Pin De-bounce Enable Bit 0 = de-bounce disable 1 = de-bounce enable **Note**: If this bit is enabled, the edge detection of TMR_EXT pin or internal 32kHz clock is detected with de-bounce circuit.
[5]	CAPIEN	Timer External Capture Interrupt Enable Bit 0 = disable 1 = enable **Note**: If CAPIEN enabled, timer will generate an interrupt when CAPIF is high. For example, while CAPIEN = 1, CAPEN = 1, and CAPEDGE = 00, an 1 to 0 transition on the TMR_EXT pin or internal 32KHz clock will cause the CAPIF to be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	CAPFUNCS	Capture Function Select Bit 0 = external reset counter mode disabled 1 = external reset counter mode enabled. **Note1**: When CAPFUNCS is 0, transition on TMR_EXT pin is used to save the 24-bit timer counter value to CAPDAT register. **Note2**: When CAPFUNCS is 1, transition on TM_EXT pin is used to save the 24-bit timer counter value to CAPDAT register and then reset the 24-bit timer counter value.
[3]	CAPEN	Timer Capture Enable Bit 0 = disable 1 = enable
[2:1]	CAPEDGE	Timer Capture Edge Detection: 00 = A falling edge on TMR_EXT pin or internal 32kHz clock will be detected. 01 = A rising edge on TMR_EXT pin or internal 32kHz clock will be detected. 10 = Either rising or falling edge on TMR_EXT pin or internal 32kHz clock will be detected. 11 = Reserved.
[0]	CNTPHASE	Event counting mode setting This bit indicates the detection of expected transition of TMR pin

		0 = A falling edge of pin will be counted and loaded to CNT value 1 = A rising edge of pin will be counted and loaded to CNT value

3.18.7.7 Timer External Interrupt Status Register (TIMER_EINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER_EINTSTS	TMR_BA+0x18	R/W	Timer External Interrupt Status Register	0x0000_0000

Bits	Descriptions	
[31:2]	Reserved	Reserved.
[1]	CAPF	Timer External Capture Flag This bit indicates the timer external capture interrupt flag status. **Note**: This bit is cleared by writing 1 to it.
[0]	CAPIF	Timer External Capture Interrupt Flag This bit indicates the timer external capture interrupt flag status. **Note**: Need set CAPIEN to 1. This bit is cleared by writing 1 to it.

3.19 CLKTRIM

3.19.1 Overview

The PAN101x series chip has an analog RC oscillator clock generation circuit, including a low-speed RC32K clock. Because the 32k clock will fluctuate over time due to the change of supply voltage or temperature. Therefore, it is necessary to quickly calibrate the RC32K when voltage or temperature changes. Supports hardware-based timing auto-calibration functionality with CLKTRIM.

3.19.2 Features

3.19.2.1 Measurement

- The software triggers the start.
- The measurement period can be configured, and the waiting time for the clock to be stable can be configured.
- Two interrupt flag: measurement completed flag, and the counter overflow flag.
- Polling mode and interrupt mode.
- The clock to be measured can be RC32K, RC32M or external clock.

3.19.2.2 Calibration

- Divided into coarse tuning (3-bit), fine tuning (6-bit), and precision tuning (8-bit).
- The clock to be calibrated is RC32K, and the software triggers to start.
- It can be used to find the optimal configuration or the configuration that meets the accuracy requirements.
- The search range can be configured.
- The measurement period can be configured, the waiting time for the clock to be stable can be configured, and the increase/decrease relationship can be configured.
- three interrupt flag: coarse tuning completed flag, fine tuning completed flag, and counter overflow interruption.
- Polling mode and interrupt mode.

3.19.3 Block Diagram

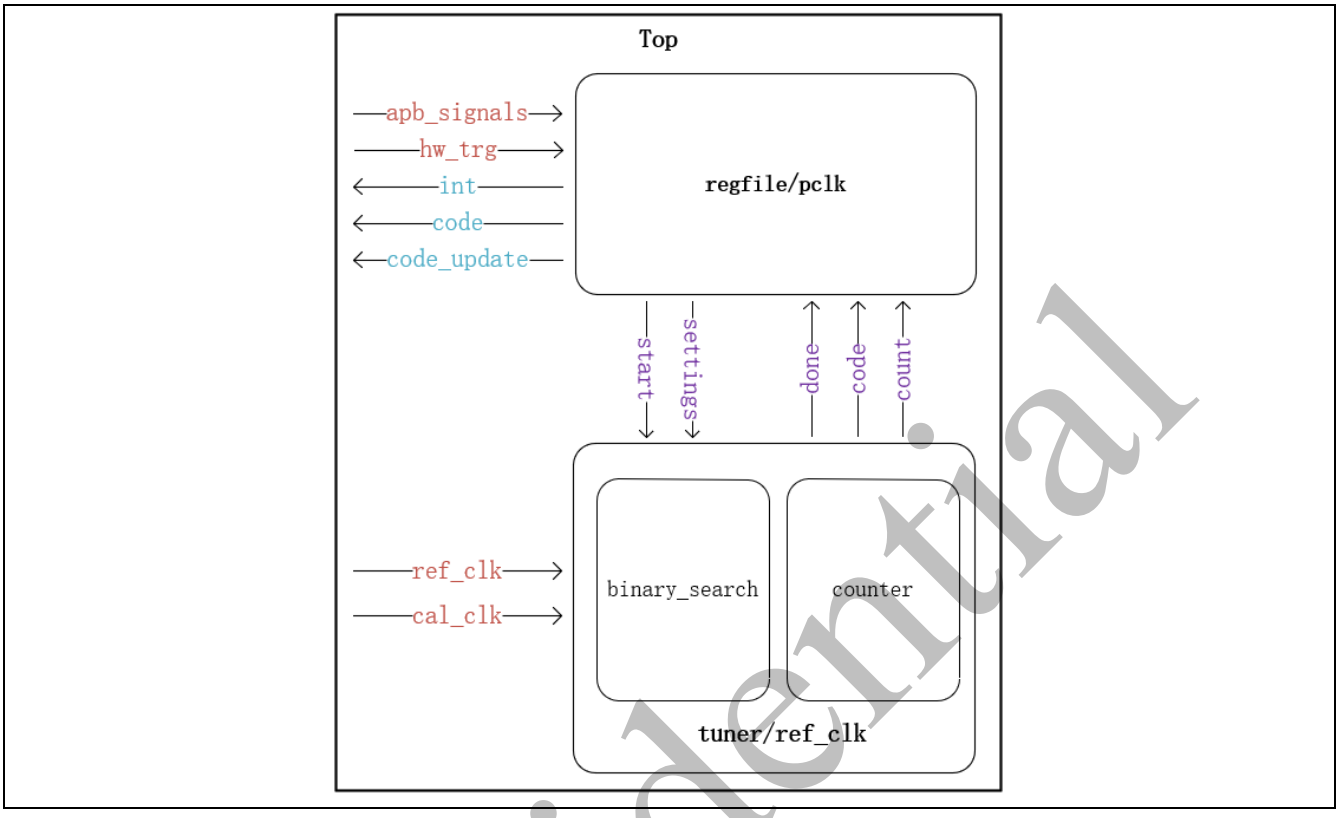


Figure 3-134 CLKTRIM Block Diagram

3.19.4 Functional Description

3.19.4.1 Clock source

The cal_clk is provided by the RCC module. cal_clk selects the rc_32k clock by default. Software can configure cal_clk to rc_32m or external input clock ext_clk. Cal_clk outputs to the clktrim module through a Gate unit. When the clktrim module is not working, the cal_clk is gated. When the software or hardware is ready to start the calibration, you should turn on the Gate uint first, wait for the cal_clk to stabilize, and then start the calibration.

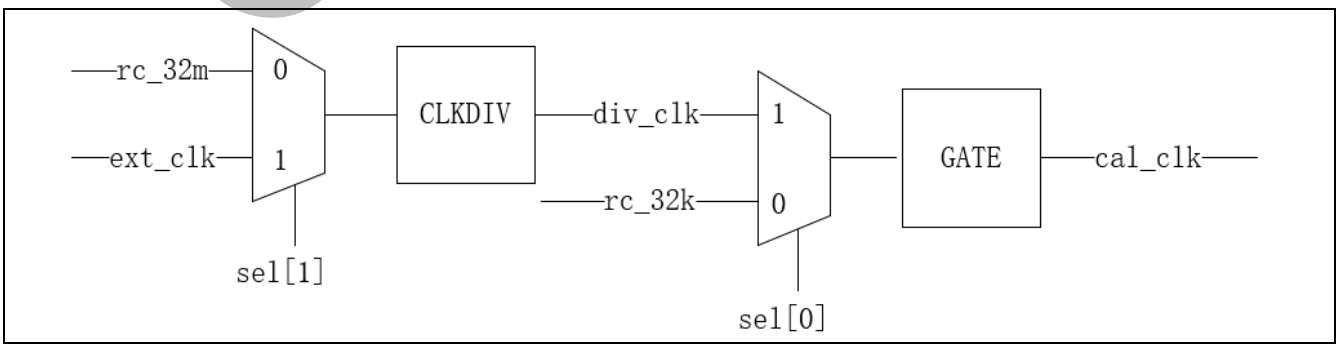


Figure 3-135 Clock Source

3.19.4.2 Measure Principle

The measurement uses a stable clock (REF_CLK) to count the clock to be measured (CAL_CLK). The frequency of the clock to be measured is Fcal, and the frequency of the reference clock is Fref. Within M cycles of the clock to be measured, the reference clock has a total of N cycles. Then the frequency of the clock to be measured is:

$$M / Fcal = N / Fref$$

$$Fcal = Fref * M / N$$

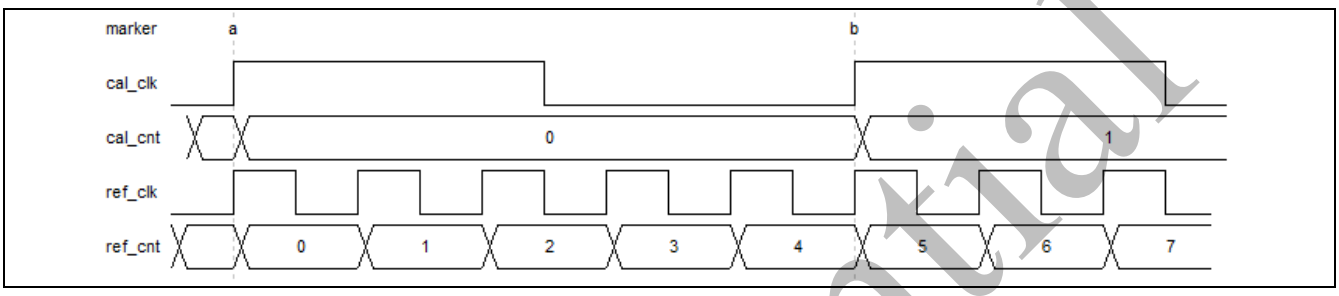


Figure 3-136 Measure Principle

The measurement method introduces two kinds of errors. One is the error caused by the jitter of the reference clock. The more stable the reference clock, the smaller the absolute error of the clock to be measured; the second is the error caused by signal synchronization. After entering the module, the clock to be measured will pass through a two-stage synchronizer. If the clock period to be measured is not an integer multiple of the stable clock period, the measurement will produce an absolute error less than or equal to one stable clock period. The following proves that the measurement error is less than or equal to a stable clock cycle:

- For end a, the resulting error is [-T, 0],
- For terminal b, the resulting error is [0, T].

The total error is the sum of the errors at the a and b ends, and the total error range is [-T, T]. Therefore, the absolute error <=T.

For this solution, the accuracy of using 32M to measure a 32K cycle is 1/1000. The accuracy of measuring ten 32K cycles is 1/10000. To achieve the accuracy requirement of 500ppm, at least 20 clock cycles need to be measured.

As shown in the figure below, the first RC32K clock cycle is synchronized to 4 stable clock cycles, and the second RC32K clock cycle is synchronized to 3 stable clock cycles. The error introduced by the measurement method will not be eliminated.

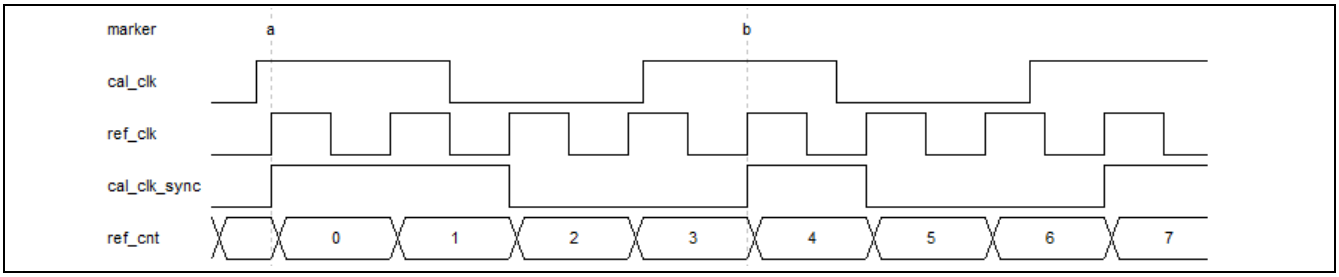


Figure 3-137 Measure Principle

The clock to be measured can also cause errors, which can be reduced. If the clock is unstable, multiple measurements will get different results. The longer the measurement period, the smaller the difference between multiple measurements and the more accurate the results obtained.

3.19.4.3 Calibration Principle

The binary search method is used for calibration. The traditional binary search method to find the target value has the following characteristics: a) Always start the search from the middle value; b) First determine the high bit, and then determine the remaining bits in turn; c) If it is equal to the target value, exit the search.

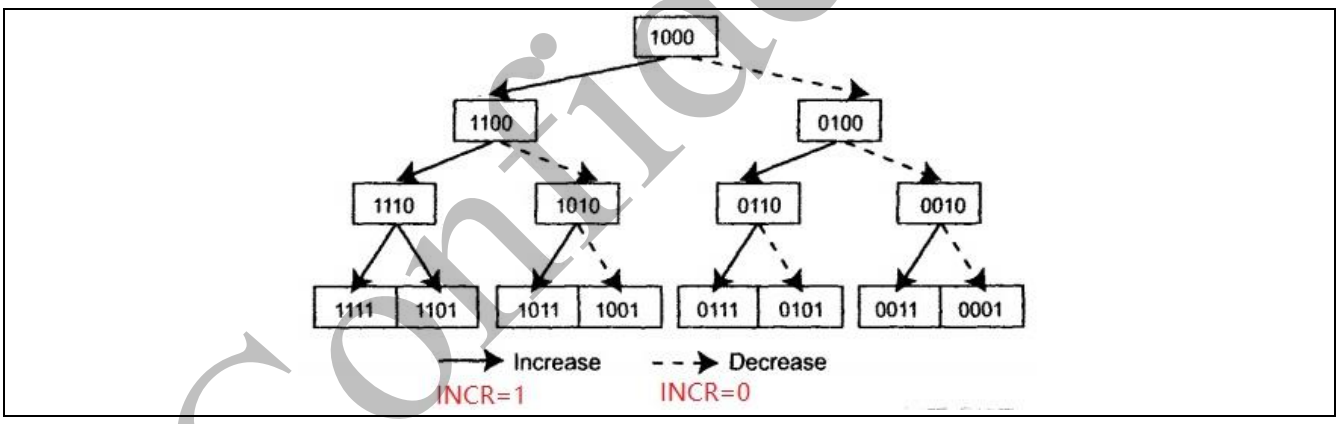


Figure 3-138 Calibration Principle

The above is a 4-bit binary search process. It is assumed that 4-bit CODE has an increasing relationship with the corresponding value. First calculate 1000. If the corresponding value is smaller than the target value, the next calculation is 1100, which corresponds to the branch on the left in the figure; if the corresponding value is too large, the next calculation is 0100, which corresponds to the branch on the right in the figure. And so on, until all bits are confirmed. In the calculation process, if the value corresponding to CODE is exactly equal to the target value, the search ends.

CLKTRIM uses the binary search method to calibrate the RC32K clock frequency, and is divided into three grades, namely coarse tuning, fine tuning and precision tuning. Because the

RC clock itself is unstable and the frequency is a continuously changing quantity, it is difficult to search a code that the corresponding frequency is exactly 32K. Therefore, we only need to look for an optimal value to make the corresponding frequency closest to 32K.

In the process of binary search for the optimal value, two registers are used to store the optimal CODE and the minimum error value respectively. Calculate the absolute error between the frequency value corresponding to the current CODE and 32K. If it is smaller than the value in the register, replace the value in the register with the current CODE and the current error value. After the search is over, the value stored in the register is the optimal CODE and minimum error value.

Currently, we can use the binary search method to determine an optimal value. However, there is a shortcoming in the method: the case when CODE is 0 is not calculated. We make a slight improvement: when the final CODE value falls at 0001, we then calculate the corresponding frequency when CODE=0, if the current error value is smaller, then determine that CODE=0 is the optimal value.

For a 4-bit CODE, it takes at least 4 times to determine the optimal value using the binary search method, and 5 times at most. If the CODE is n-bit, it takes at least n times to determine the optimal value. If n is relatively large, the number of searches will be greater and the time spent will be longer. There are the following methods to shorten the binary search time:

1) Search in a small range

   Take the 4-bit dichotomy as an example. Based on empirical values, assuming we already know that the optimal value will fall between 0-7, then the first search can start from 0100 instead of 1000. If the optimal value is known to fall between 8-15, you can directly start the search from 1100.

2) Add the judgment condition for ending the search early

   Generally, there is an accuracy requirement for the RC clock, such as 32K±500ppm. The accuracy requirement can be used as the judgment condition for exiting the search early. If the error value corresponding to the current CODE value is within the accuracy range, the search is ended early and no longer continues to search for an optimal value.

3) Reduce the measurement period

   A CODE value corresponds to a frequency value, which is measured using the method in section 3.2. In order to reduce the measurement error, the measurement period can be increased, and the time corresponding to the binary search method is longer. Therefore, users need to weigh the relationship between measurement error and time. If the

measurement error requirements are relatively small, the measurement period can be reduced, and an optimal value can be quickly obtained through the dichotomy.

3.19.5 CLKTRIM Register Map

Register	offset	R/W	Description	Reset Value
CLKTRIM_BASE addr：0x4001_6000-0x4001_6FFF				
ClktrimEnReg	CLKTRIM_BASE+0x00	R/W		0x0000_0000
ClktrimCodeReg	CLKTRIM_BASE+0x04	R/W		0x0007_8008
ClktrimCtlReg	CLKTRIM_BASE+0x08	R/W		0x000f_0f02
ClktrimIntReg	CLKTRIM_BASE+0x0C	R/W		0x0000_0001
ClktrimCalCntReg	CLKTRIM_BASE+0x10	R/W		0x0400_001e
ClktrimIdeaCntReg	CLKTRIM_BASE+0x14	R/W		0x0000_752f
ClktrimRefCntReg	CLKTRIM_BASE+0x18	R		0x0000_0000

3.19.6 CLKTRIM Register Description

3.19.6.1 ClktrimEnReg

Register	Offset	R/W	Description	Reset Value
ClktrimEnReg	CLKTRIM_BASE +0x00	R/W	Enable register	0x0000_0000

Bits	Description	
[31:3]	Reserved	Reserved
[2]	FINE_EN	1:Start fine tuning 0:This bit will be cleared automatically after fine tuning finished or REFCNT overflow happens Reset_value:{1b0} Because CLKTRIM retains software calibration functionality, when software wishes to disable hardware timing, it is necessary to first disable hardware timing calibration, and then enable FINE_EN=1. When software-based calibration is successful, FINE_EN will be automatically cleared. When software enables calibration, it needs to disable hardware timing calibration, and setting FINE_EN=1 will affect the interrupt indicating the successful hardware timing calibration。
[1]	COARSE_EN	1:Start coarse tuning 0:This bit will be cleared automatically after coarse tuning finished or REFCNT overflow happens Reset_value:{1b0} Because CLKTRIM retains software calibration functionality, when software wishes to disable hardware timing, it is necessary to first disable hardware timing calibration, and then enable COARSE_EN=1. When software-based calibration is successful, COARSE_EN will be automatically cleared. When software enables calibration, it needs to disable hardware timing calibration, and setting COARSE_EN=1 will affect the interrupt indicating the successful hardware timing calibration.
[0]	MEAS_EN	1: Start clock measure 0: This bit will be cleared automatically if clock measurement is finished or REFCNT overflow happens Reset_value:{1b0}

3.19.6.2 ClktrimCodeReg

Register	offset	R/W	Description	Reset Value
ClktrimCodeReg	CLKTRIM_BASE +0x04	R/W		0x0007_8008

Bits	Description	
[31:19]	Reserved	Reserved
[18:16]	BIT_WIDTH	Decide n-bit binary search, n=BIT_WIDTH+1 Coarse tuning:4-bit Fine tuning:8-bit If BIT_WIDTH = m, /RC32_F/RC32K_C[m:0] should be set to{1'b1, m{1'b0}.The other bits are decided by users. Reset_value:{0x7}
[15:8]	RC32K_F	Fine tuning code Reset_value:{8'b1000_0000}
[7:4]	Reserved	Reserved
[3:0]	RC32K_C	Coarse tuning code Reset_value:{4'b1000}

3.19.6.3 ClktrimCtlReg

Register	offset	R/W	Description	Reset Value
ClktrimCtlReg	CLKTRIM_BASE +0x08	R/W		0x000f_0f02

Bits	Description	
[31:16]	ERR_RANGE	Decide accuracy range. ERROR = abs{REFCNT - IDEA_CNT} If ERROR < ERR_RANGE,binary search will be early terminated. Reset_value:{0x000f}
[15:8]	CODE_STEP	When selecting calibration, it refers to the code that has been added or subtracted from the previous calibration code Reset_value：{8'h0f}
[7:3]	Reserved	Reserved
[2]	HW_FAIL_CALIBRATE	1:hardware calibrate fail flag 0:hardware calibrate not fail This bit is cleared by writing 1 to 0. Reset value:{1'b0}
[1]	EARLY_TERM_EN	1: Early termination binary search is enable 0: disable Reset_value:{1b1}
[0]	DECR	The relation ship between tuning code and frequency. 0:Increase relation 1:Decrease relation Reset_value:{0x0}

3.19.6.4 ClktrimIntReg

Register	offset	R/W	Description	Reset Value
ClktrimIntReg	CLKTRIM_BASE+0x0C	R/W	Interrupt flag register	0x0000_0001

Bits	Description	
[31:11]	Reserved	Reserved
[10]	OVF_FLAG	REFCNT overflow flag. 1: REFCNT is overflow 0: REFCNT is not overflow This bit is cleared by writing 1 to it.
[9]	OVF_INT	REFCNT overflow interrupt 1: REFCNT overflow interrupt is triggered.。 0: No change Always 0 when interrupt is masked. This bit is cleared by writing 1 to it.
[8:7]	Reserved	Reserved
[6]	FTUNE_STOP_FLAG	Fine tuning stop flag 1: Fine tuning is stop 0: No change This bit is cleared by writing 1 to it.
[5]	FTUNE_STOP_INT	Fine tuning stop interrupt 1: Stop interrupt is triggered. 0: No change Always 0 when interrupt is masked. This bit is cleared by writing 1 to it.
[4]	CTUNE_STOP_FLAG	Coarse tuning stop flag 1: Coarse tuning is stop 0: No change This bit is cleared by writing 1 to it.
[3]	CTUNE_STOP_INT	Coarse tuning stop interrupt 1: Stop interrupt is triggered. 0: No change Always 0 when interrupt is masked. This bit is cleared by writing 1 to it.
[2]	MEAS_STOP_FLAG	Clock measurement stop flag 1: Clock measurement is stop 0: No change This bit is cleared by writing 1 to it.
[1]	MEAS_STOP_INT	Clock measurement stop interrupt 1: Stop interrupt is triggered. 0: No change Always 0 when interrupt is masked. This bit is cleared by writing 1 to it. After the measurement is completed, this interrupt is triggered. Once the interrupt is triggered, the measurement enable signal is automatically cleared.。
[0]	INT_EN	1:Interrupt is enabled 0:Interrupt is masked Default value is 1.

3.19.6.5 ClktrimCalCntReg

Register	offset	R/W	Description	Reset Value
ClktrimCalCntReg	CLKTRIM_BASE+0x10	R/W	Measurement period register	0x0400_001e

Bits	Description	
[31:24]	WAIT_CNT	You must wait N calibration clock cycles to make sure RC32K is stable. This register can not be set to zero. N = WAIT_CNT Reset_value:{0x4}
[23:0]	CAL_CNT	Decide the number of calibration clock cycles you want count. This register can not be set to zero. Reset_value:{0x64}

3.19.6.6 ClktrimIdeaCntReg

Register	offset	R/W	Description	Reset Value
ClktrimIdeaCntReg	CLKTRIM_BASE+0x14	R/W	Ideal count register	0x0000_752f

Bits	Description	
[31:0]	IDEA_CNT	Used to compute ERROR.This is a 0-base counter. IDEA_CNT = CAL_CNT * (REF_CLK / CAL_CLK) -1 Reset_value:{0x1869F} //

3.19.6.7 ClktrimRefCntReg

Register	offset	R/W	Description	Reset Value
ClktrimRefCntReg	CLKTRIM_BASE+0x18	R	Reference count register	0x0000_0000

Bits	Description	
[31:0]	REF_CNT	Read-only bits. This is a 0-base counter. If tuning is started, this register indicates the optimal value of reference clock cycles. If measurement is started, this register indicated the number of reference clock cycles. REF_CNT will be updated after tuning or measurement is finished.

3.19.7 Software flow

3.19.7.1 Measurement

1. Select the clock to be tested and the reference clock, and wait for the clock to stabilize;
2. Configure WAIT_CNT and CAL_CNT. Set the waiting time and measurement period;
3. Configure INT_EN, select interrupt mode or polling mode;
4. Enable MEAS_EN;
5. In interrupt mode. After the interrupt is triggered, check the interrupt status. If MEAS_STOP_INT is set to 1, read the value of REFCNT. Calculate the true frequency of CAL_CLK through the formula in 3.19.4.2. If OVF_INT is set to 1, check the clock, reduce the measurement period, and restart the measurement;
6. In the polling mode. After the software detects that MEAS_STOP_FLAG is set to 1, it reads REFCNT and calculates the CAL_CLK frequency. If it is found that OVF_FLAG is set to 1, check the clock, reduce the measurement period, and restart the measurement;
7. Clear status flag, interrupt flag.

3.19.7.2 Calibration

1. Select the clock to be calibrated and the reference clock;
2. Configure RC32K_F/RC32K_C and BIT_WIDTH. Set the initial value of precision tuning /fine tuning /coarse tuning, and set the n-bit binary search method. The initial value must correspond to BIT_WIDTH, for example, BIT_WIDTH=2, which means the 3-bit binary search method, the initial value must be (xx100);
3. Configure DECR. Set the relationship between CODE and frequency, increase/decrease;
4. Configure INT_EN, select interrupt mode or polling mode;
5. Configure WAIT_CNT and CAL_CNT. Set RC stabilization time and measurement period;
6. Configure IDEA_CNT. Set the target value of the binary search method;
7. Configure EARLY_TERM_EN and ERR_RAGNE. Set the early exit search enable signal and accuracy range;
8. Enable COARSE_EN/FINE_EN/PRECISION_EN;
9. After the calibration is over, clear the status flag bit and the interrupt flag bit.

3.20 Electronic Codebook Mode Encryption (ECB)

3.20.1 Overview

The AES electronic codebook mode encryption (ECB) can be used for a range of cryptographic functions like hash generation, digital signatures, and keystream generation for data encryption/decryption. The ECB encryption block supports 128 bit AES encryption (both encryption and decryption).

3.20.2 Features

- 128 bit AES encryption
- Supports standard AES ECB block encryption and decryption
- Memory pointer support

3.20.3 Block Diagram

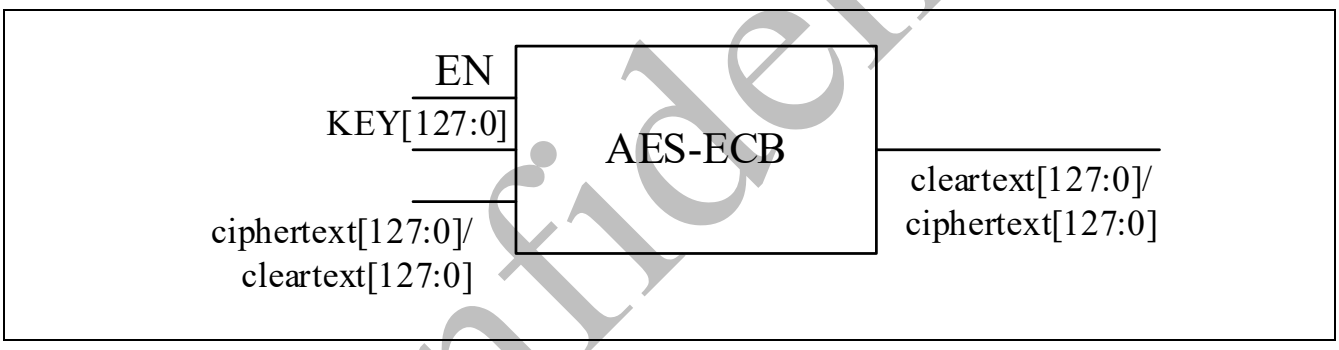


Figure 3-139 AES-ECB Diagram

3.20.4 Functional Description

AES ECB access to Link Layer Data RAM for in-place operations on cleartext and ciphertext during encryption or decryption.

3.20.5 AES-ECB Control Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
AES-ECB Base Address: AES-ECB_BA = 0x50020000				
SECURE1	0x46C	R/W		0x000a_0000
SECURE2	0x470	R/W		0xbadc_ab24
SECURE3	0x474	R/W		0xdeaf_babe
SECURE4	0x478	R/W		0x0000_0000
SECURE5	0x47C	R/W		0x0000_0000
SECURE6	0x480	R/W		0x0000_0000
SECURE7	0x484	R/W		0x0000_0001
SECURE8	0x488	R/W		0x0000_0080
SECURE9	0x48C	R/W		0x0000_0001
SECURE10	0x490	R/W		0x0000_0080

3.20.6 AES-ECB Register Description

3.20.6.1 SECURE1

Register	Offset	R/W	Description	Reset Value
SECURE1	0x46C	R/W		0x000a_0000

Bits	Description	
[31:27]	Reserved	Reserved
[26]	SECURE1_SEC_DECRYPT_EN	Secure decrypt enable by software(used by ECB decrypt) 0: disable 1: enable
[25:19]	SECURE1_SEC_AD_LEN	Length of AD Data
[18:17]	SECURE1_SEC_MIC_LEN	MIC Length 0 => 0Bytes 1 => 4Bytes 2 => 8Bytes 3 => 16Bytes
[16:10]	SECURE1_SEC_VLD_MAC	This field specifies the number of packets that have valid MIC.
[9:3]	SECURE1_SEC_PKT_ENABLES	This field indicates the packets to be processed.
[2:1]	SECURE1_SEC_MODE	Security mode 0 => ECB_Decrypt 1 => Reserved 2 => ECB_Encrypt 3 => Reserved
[0]	SECURE1_SEC_ENABLE	This field specifies the active-high Enable AES. - 1'b0: Enable secure controller - 1'b1: Disable secure controller

3.20.6.2 SECURE2

Register	Offset	R/W	Description	Reset Value
SECURE2	0x470	R/W		0xbadc_ab24

Bits	Description	
[31:24]	SECURE2_SEC_IV3	This field specifies Nonce 8.
[23:16]	SECURE2_SEC_IV2	This field specifies Nonce 7.
[15:8]	SECURE2_SEC_IV1	This field specifies Nonce 6.
[7:0]	SECURE2_SEC_IV0	This field specifies Nonce 5.

3.20.6.3 SECURE3

Register	Offset	R/W	Description	Reset Value
SECURE3	0x474	R/W		0xdeaf_babe

Bits	Description	
[31:24]	SECURE3_SEC_IV7	This field specifies Nonce 12.
[23:16]	SECURE3_SEC_IV6	This field specifies Nonce 11.
[15:8]	SECURE3_SEC_IV5	This field specifies Nonce 10.
[7:0]	SECURE3_SEC_IV4	This field specifies Nonce 9.

3.20.6.4 SECURE4

Register	Offset	R/W	Description	Reset Value
SECURE4	0x478	R/W		0x0000_0000

Bits	Description	
[31:16]	SECURE4_SEC_KEY2_RAR	This field specifies the key pointer for packet 1.
[15:0]	SECURE4_SEC_KEY1_RAR	This field specifies the key pointer for packet 0.

3.20.6.5 SECURE5

Register	Offset	R/W	Description	Reset Value
SECURE5	0x47C	R/W		0x0000_0000

Bits	Description	
[31:16]	SECURE5_SEC_KEY4_RAR	This field specifies the key pointer for packet 3.
[15:0]	SECURE5_SEC_KEY3_RAR	This field specifies the key pointer for packet 2.

3.20.6.6 SECURE6

Register	Offset	R/W	Description	Reset Value
SECURE6	0x480	R/W		0x0000_0000

Bits	Description	
[31:16]	SECURE6_SEC_KEY6_RAR	This field specifies the key pointer for packet 5.
[15:0]	SECURE6_SEC_KEY5_RAR	This field specifies the key pointer for packet 4.

3.20.6.7 SECURE7

Register	Offset	R/W	Description	Reset Value
SECURE7	0x484	R/W		0x0000_0001

Bits	Description	
[31]	Reserved	Reserved
[30:24]	SECURE7_SEC_MD_LEN	Length of m Data
[23:8]	SECURE7_SEC_KEY7_RAR	This field specifies the key pointer for packet 6.
[7:0]	SECURE7_SEC_PKT_CNT_TX_MSB	Counter for the transmitted packets to be processed (Nonce 0:4) (MSB).

3.20.6.8 SECURE8

Register	Offset	R/W	Description	Reset Value
SECURE8	0x488	R/W		0x0000_0080

Bits	Description	
[31:0]	SECURE8_SEC_PKT_CNT_TX_LSB	Counter for the transmitted packets to be processed (Nonce 0:4) (LSB).

3.20.6.9 SECURE9

Register	Offset	R/W	Description	Reset Value
SECURE9	0x48C	R/W		0x0000_0001

Bits	Description	
[31:8]	Reserved	Reserved
[7:0]	SECURE9_SEC_PKT_CNT_RX_MSB	Counter for the received packets to be processed (Nonce 0:4)(MSB).

3.20.6.10 SECURE10

Register	Offset	R/W	Description	Reset Value
SECURE10	0x490	R/W		0x0000_0080

Bits	Description	
[31:0]	SECURE10_SEC_PKT_CNT_RX_LSB	Counter for the received packets to be processed (Nonce 0:4)(LSB).

3.21 Random Number Generators (RNG)

3.21.1 Overview

The Random number generator (RNG) generates true non-deterministic random numbers based on internal thermal noise that are suitable for cryptographic purposes. The RNG does not require a seed value.

3.21.2 Features

- 32-bit random number
- Support scramble function

3.21.3 Block Diagram

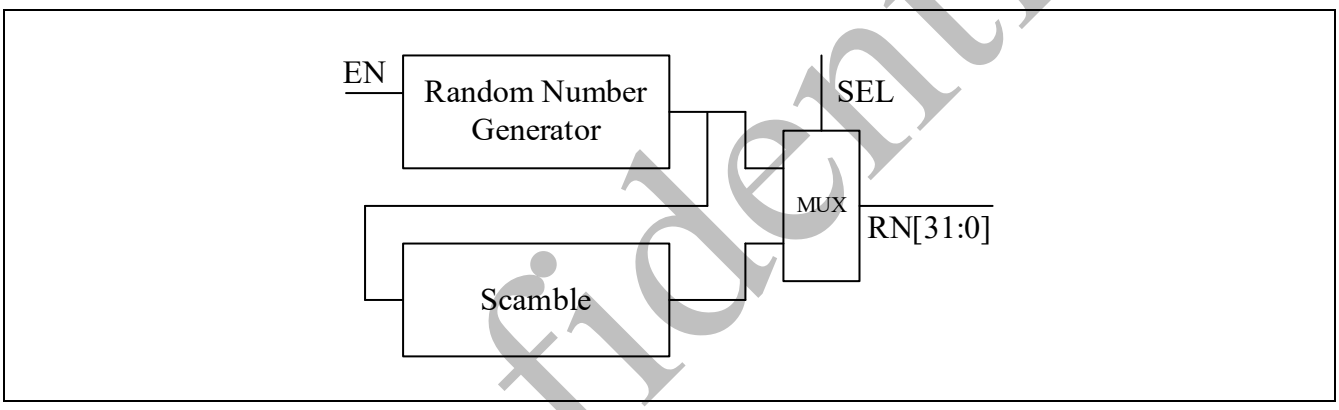


Figure 3-140 RNG Block Diagram

3.21.4 Functional Description

The RNG is started by triggering the EN task. When started, new random numbers are generated continuously and written to the VALUE register.

A simple scramble algorithm is employed on the internal bit stream to avoid long '1' or '0'. The value is then queued into VALUE register. It is possible to select the original random number or the scramble number as the final value.

3.21.5 RNG Control Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
RNG Base Address: RNG_BA = 0x50020000				
INTR1	0x444	R/W	RNG interrupt flag	0x0000_0000
INTCLR	0x448	R/W	RNG interrupt clear	0x0000_0000
INTMSK	0x44C	R/W	RNG interrupt mask	0x0000_0002
RNG1	0x49C	R/W	RNG control	0x0000_0000
RNG2	0x4A0	R	RNG value	0x0000_0000

3.21.6 RNG Register Description

3.21.6.1 RNG Interrupt Flag Register(INTR1)

Register	Offset	R/W	Description	Reset Value
INTR1	0x444	R/W	RNG interrupt flag	0x0000_0000

Bits	Description	
[31:4]	Reserved	Reserved.
[3]	INTR1_IC_RNG_DONE	This field specifies the output status done signal of the TRNG.
[2:0]	Reserved	Reserved.

3.21.6.2 RNG Interrupt Clear Register (INTCLR)

Register	Offset	R/W	Description	Reset Value
INTCLR	0x448	R/W	RNG interrupt clear	0x0000_0000

Bits	Description	
[31:4]	Reserved	Reserved
[3]	INTCLR_IC_RNG_DONE_CLR	This field specifies the active-high clear signal of TRNG done output status.
[2:0]	Reserved	Reserved.

3.21.6.3 RNG Interrupt Mask Register (INTMSK)

Register	Offset	R/W	Description	Reset Value
INTMSK	0x44C	R/W	RNG interrupt mask	0x0000_0002

Bits	Description	
[31:4]	Reserved	Reserved
[3]	INTMSK_IC_RNG_DONE_MASK	This field specifies the mask of the done signal of TRNG done interrupt
[2:0]	Reserved	Reserved.

3.21.6.4 RNG Control Register (RNG1)

Register	Offset	R/W	Description	Reset Value
RNG1	0x49C	R/W	RNG control	0x0000_0000

Bits	Description	
[31:2]	Reserved	Reserved
[1]	RNG1_RNG_RING_SCRMB_SEL	This field enables the RING output only.
[0]	RNG1_RNG_EN	This field specifies the enable signal for the RNG module.

3.21.6.5 RNG Value Register (RNG2)

Register	Offset	R/W	Description	Reset Value
RNG2	0x4A0	R	RNG value	0x0000_0000

Bits	Description	
[31:0]	RNG2_RNG_DATA	This field specifies the 32-bit stream data output of the TRNG module.

3.22 Real Time Counter (RTC)

3.22.1 Overview

The Real time counter (RTC) module provides a generic, low power timer on the low-frequency clock source.

3.22.2 Features

- 32-bit counter
- Support 3-compare
- Support interrupt
- Support 3 clock source 32kHz RC or 32.768kHz XTAL or 32kHz precision clock

3.22.3 Block Diagram

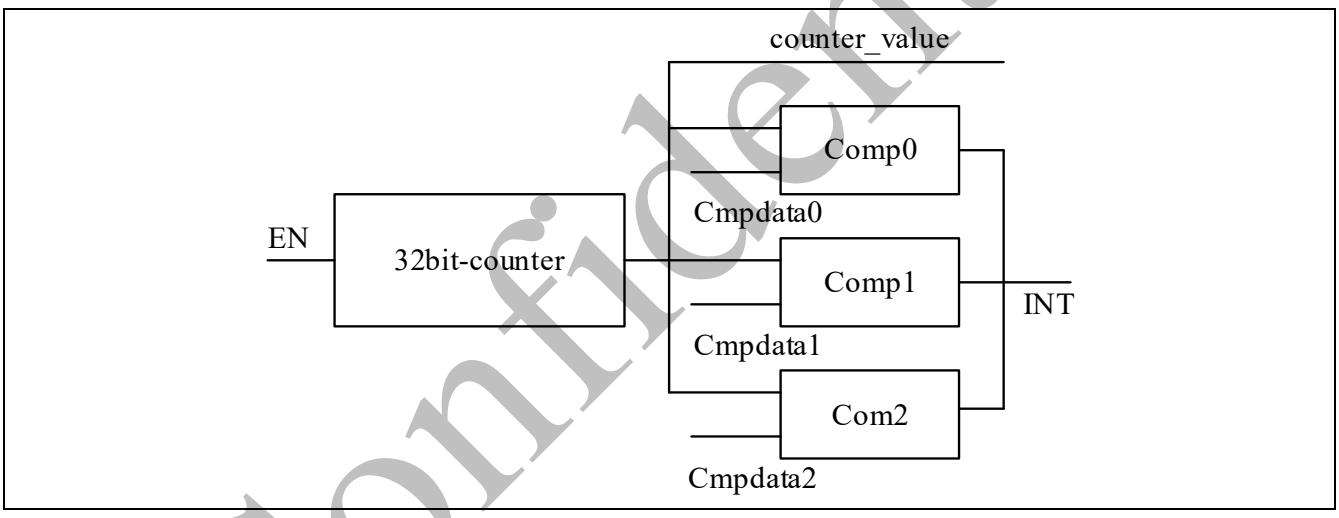


Figure 3-141 RTC Block Diagram

3.22.4 Functional Description

The RTC is started by triggering the EN task. When started, software can read the counter value at any time. The RTC supports 3 compare function, each compare value is set by software, compare value is a spcace time(target time – initial time), when counter value counts to compare value, a interrupt is generated. The RTC is a continue running timer, it will goon running while compare is accur.

3.22.5 RTC Control Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
ANA Base Address: ANA_BA = 0x4007_0000				
LP_FL_CTRL	ANA_BA+0x04	R/W	Low power control	0x0000_0048
LP_SPACING_TIME0	ANA_BA+0x08	R/W	The spcing time0 of slptmr	0x0000_0000
LP_SPACING_TIME1	ANA_BA+0x0C	R/W	The spcing time1 of slptmr	0x0000_0000
LP_SPACING_TIME2	ANA_BA+0x10	R/W	The spcing time2 of slptmr	0x0000_0000
LP_SLPTMR	ANA_BA+0x14	R	SLPTMR value	0xxxxx_xxxx
LP_INT_CTRL	ANA_BA+0x18	R/W	Low power interrupt control	0x0001_0001
ACT_32K_CTRL	RCC_BA+0x54	R/W	active 32K clock control	0x0000_0000
ACT_32K_BASECORR	RCC_BA+0x58	R/W	active 32K clock base correction	0x0000_0000

3.22.6 RTC Register Description

Refer to 3.2.9 Register Description

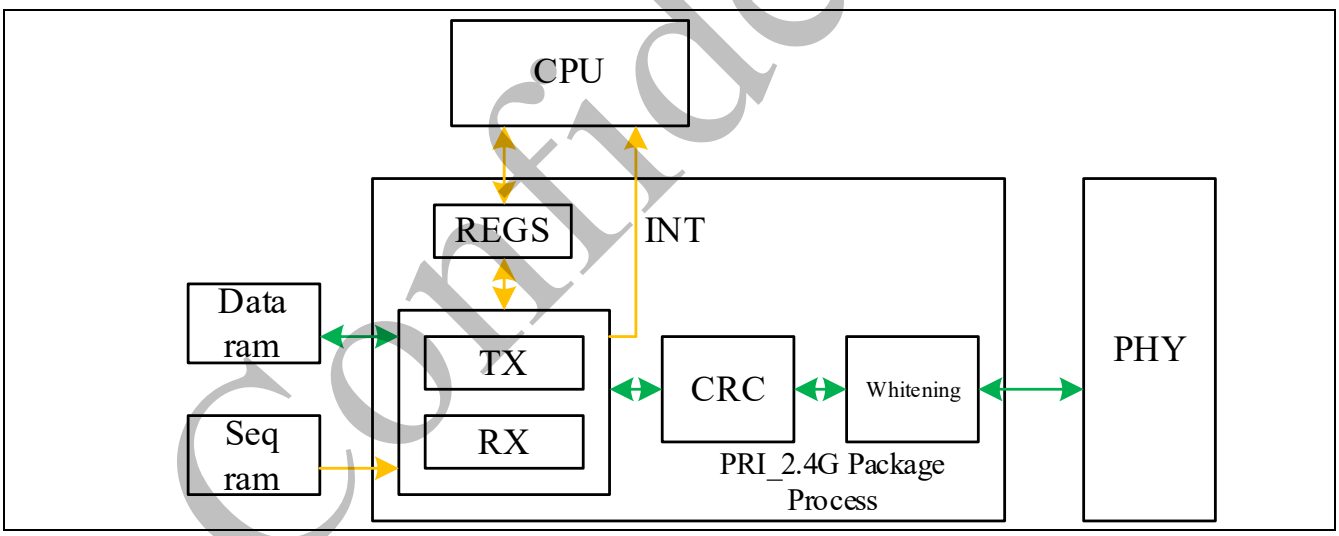
4 2.4 GHz proprietary protocols

4.1 System features

- Support 250K, 1M and 2M PHY
- XN297L, PAN1026, Nordic Transceiver protocol compliant
- Support No Acknowledge, Acknowledge and Acknowledge with payload
- Support CRC8, CRC16 and CRC24
- Support whitening
- Compatible with Bluetooth frame structure, can simulate Bluetooth broadcast and scanning
- Compatible with Bluetooth CODED PHY S2/S8
- Supports 2-byte address
- Support the same spread spectrum function as the BLE protocol

4.2 Block diagram

The overall structure of the proprietary is as follows:



The proprietary protocols implementation is mainly a packet processing module, including register module, TX state control, RX state control, CRC, whitening.

4.3 Frame structure for proprietary protocols

4.3.1 XN297 frame structure

The XN297 frame structure has three modes: Normal, Normal_m1 and Enhanced.

4.3.1.1 Normal mode

The normal mode frame structure consists of Preamble, Addr, Payload and CRC, which the address and length can be configured, and the CRC can choose crc24, crc16 and then crc8.

4.3.1.2 Normal_m1 mode

The frame structure is the same as the normal mode frame structure, in this mode:

After TX ends, it automatically enters RX mode after waiting for the latency (hardware latency).

After RX ends, it automatically enters TX mode after waiting for the latency (hardware latency).

4.3.1.3 Enhanced mode

Where signal includes：

Preamble: 3bytes total, 0x710F55, sent from the high bit.

Addr: 2, 3, 4, 5bytes address length configurable, configured by register, sent from the high bit.

Signal: In enhanced mode, signal is made up of {length[6:0], tx_pid[1:0], tx_no_ack}. It is sent from the high bit, and is not available in normal mode. Pid can be maintained by the software at each transmit.

Payload: Send from the low bit.

Crc24/crc16/crc8: Sent from the high bit.

Whitenning: Addr (Optional), Signal(Normal mode without this segment), Payload and Crc24. Initial values are configurable.

CRC: Addr(Optional), Signal (Normal mode without this segment), Payload.

4.3.2 NRF frame structure

The NRF frame structure has three modes: Normal, Normal_m1 and Enhanced.

4.3.2.1 Normal mode

Preamble (1~24Bytes)	Addr (2-5Bytes)	Header1 (0-1Bytes)	Header0 (0-1Bytes)	Length (0-1Bytes)	Payload (0-255Bytes)	Crc24/crc16/crc8 (3/2/1Bytes)

The normal mode frame structure consists of Preamble, Addr, Header1, Header0, Length, Payload and CRC, which the length of Preamble, the Addr and Length can be configured, and the CRC can choose crc24, crc16 and then crc8. Header1, Header0 and Length are optional fields, determined by register pri_04[8] and register pri_04[10:9] together.

4.3.2.2 Normal_m1 mode

The frame structure is the same as the normal mode frame structure, in this mode:

After TX ends, it automatically enters RX mode after waiting for the interval timing which is consists of hardware timing and setting timing.

After RX ends, it automatically enters TX mode after waiting for the interval timing which is consists of hardware timing and setting timing.

4.3.2.3 Enhanced mode

Preamble (1Byte)	Addr (2-5Bytes)	Signal (9/11bits)	Payload (0-255Bytes)	Crc24/crc16/crc8 (3/2/1Bytes)

There are two types of enhanced mode, the difference is the structure of the signal, the length of the enhanced-1 is 6bits, and the length of the enhanced-2 is 8bits, as follows:

Length (6bits)	PID (2bits)	NOACK (1bit)

Length (8bits)	PID (2bits)	NOACK (1bit)

Preamble: 1byte in total. If the highest bit of the Addr is 1, send 10101010, otherwise send 01010101. It is sent from the high bit. Length is configurable, which supports 1, 2, 3, 4, 6, 8, 12, 16, 24B.

Addr: Configured by the register, variable length (2-5bytes), sent from the high bit.

Signal: In enhanced mode, signal is made up of {length[5/7:0], tx_pid[1:0], tx_no_ack}. It is

not available in normal mode. Pid can be maintained by the software at each transmit.

Payload: Sent from the high bit.

Crc24/crc16/crc8: The value after the calculation is completed, sent from the high bit.

Whitening: Addr (Optional), Signal(Normal mode without this segment), Payload and Crc24. Initial values are configurable.

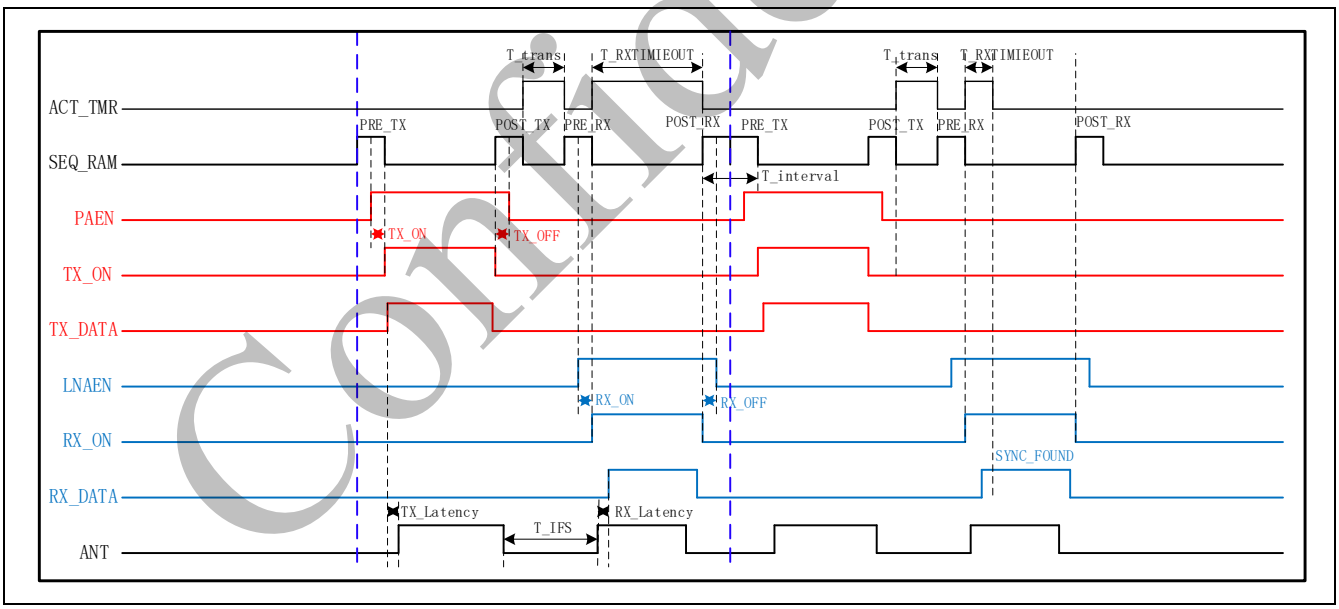
CRC: Addr(Optional), Signal (Normal mode without this segment), Payload.

4.3.3 250k frame structure

The 250k rate mode supports XN297, nordic and CC2510FX/CC2511FX communication protocols. For CC2510FX/CC2511FX frame structure, use nordic normal mode frame structure adaptation, and enable CRC and whitening function.

4.4 Timing Description

The switching timing of PTX is shown in the figure below:



The timing of PRX is similar with PTX. For single TX or single RX timing, see the TX or RX section in the figure above. For conversion time calculations, see the following table:

Name	Description
PRE_TX/POST_TX/ PRE_RX/POST_RX	The timing is set in tx/rx SEQ RAM and is calculated by software .
TX_ON	The starting time of PA which is composed of the latency part of the SEQ RAM and the time of SPI sending data.

TX_OFF	The delay time of disable PA at TX mode, which must be greater than TX_latency.
RX_ON	The starting time of NA which is composed of the latency part of the SEQ RAM and the time of SPI sending data.
RX_OFF	The delay time of disable LNA at RX mode
T_trans	Pri_2.4G interval to complete TX/RX mutual switching TX-->RX: T_IFS=T_trans+ TX_Latency+ RX_Latency+POST_TX+PRE_RX RX-->TX: T_IFS=T_trans+POST_RX+PRE_TX T_trans: Configured by the register TRX_TRANS_WAIT_TIME
TX_Latency	Sending data, latency from MAC to ANT, hardware latency, theoretically a fixed value. The software has a macro definition for TX_Latency.
RX_Latency	Receiving data, latency from ANT to MAC, hardware latency, theoretically a fixed value, The software has a macro definition for RX_Latency.
T_interval	Interval between two consecutive PTX or PRX PTX: Determined by POST_RX, PRE_TX and software latency PRX: Determined by POST_TX, PRE_RX, and software latency
T_RXTIMEOUT	RX timeout time, configured by the register bit RX_WAIT_TIME, this timeout counter stops working when RX receives the sync word (SYNC_FOUND) correctly.

4.5 PID

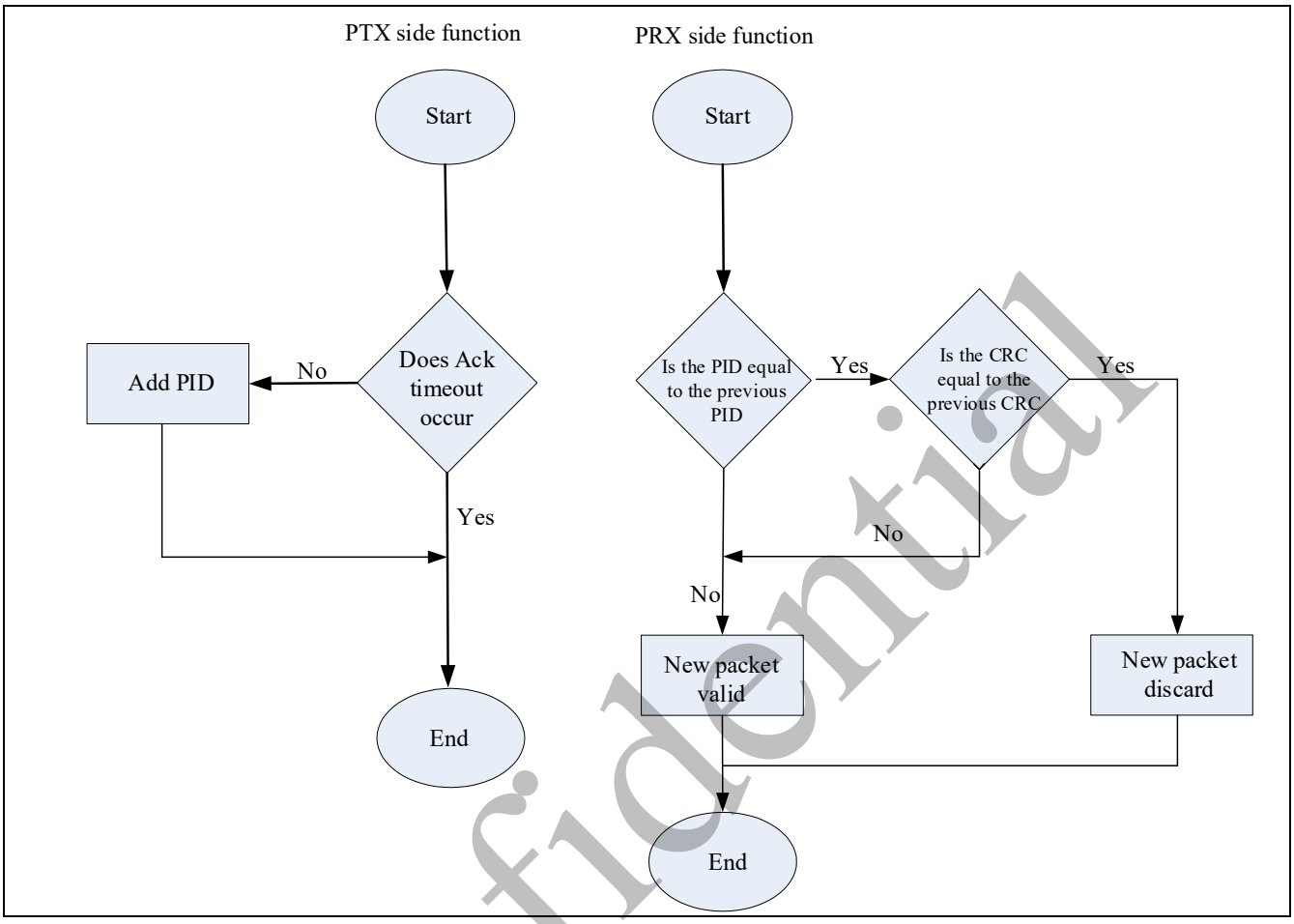


Figure 4-1 PID generation and detection

Each packet of data includes a two-bit PID (Packet Identifier Digital) to help the receiving side identify whether the data is a new packet or a resent packet, preventing multiple deposits of the same packet, and the PID is generated and detected as shown in Figure 4-1. The PID value is incremented by one when no timeout occurs for the ack packet at the sending side.PID can be maintained by the software at each transmit.

4.6 Register Description

4.6.1 PRI_R00

Register	Offset	R/W	Description	Reset Value
pri_r00	0x4c0	R/W		0x0

Bits	Descriptions	
[31:24]	TX_PAYLOAD_LENGTH	Send payload length.
[23:16]	RX_PAYLOAD_LENGTH	Receive payload length.
[15]	ACCADDR_CRC_DIS	0: crc will include access address，header，payload 1: crc will just include header，payload
[14]	CRC_SEL24	1: crc24 0: Determined by the register crc_sel16
[13:12]	ADDR_BYTE_LENGTH	RX/TX address width, if the address width is set below 5 bytes, the address uses a low byte. 00: 2 bytes 01: 3 bytes 10: 4 bytes 11: 5 bytes
[11]	DPY_EN	In enhanced mode: By configuring this bit, the hardware can independently determine the length of the receiving packet, and there is no need to configure the receiving payload length rx_payload_length information on the software side. Normal mode is not supported. Note: Enhanced PTX devices must enable this bit when receiving ack with payload.
[10]	CRC_ENABLE	Crc enable bit 1: crc enable 0: crc disable
[9]	CRC_SEL16	crc_sel24 equal to 0, this bit is valid. 1: crc16 0: crc8
[8]	SCR_ENABLE	Scrambling function is enabled or not. 1: whitening enable 0: whitening disable
[7]	NRF_ENHANCE	0: nrf_enh0 1: nrf_enh1
[6]	ENHANCE	Enhanced Mode Configuration
[5]	BW_MODE	0: 1Mbps 1: 2Mbps
[4:3]	CHIP_MODE	Operating mode selection（In BLE mode, pri module clock gate） 0x: rsv　10: 297　11: nrf
[2]	Reserved	Reserved
[1]	TX_NOACK_EN	If configured to 1, the no-ack bit is 1 in enhanced mode, and no rx reply to ACK is required.
[0]	PRI_RX	Rx/tx control bit

		1: prx 0: ptx

4.6.2 PRI_R01

Register	Offset	R/W	Description	Reset Value
pri_r01	0x4c4	R/W		0x00000000

Bits	Descriptions	
[31]	Reserved	Reserved
[30:29]	RX_PID	Read only The rx pid of current package
[28:27]	TX_PID	Read only The tx pid of current package
[26]	PRI_RX_LNGTH_ERR_FLAG	Read only Received packet length exceeds set value, error flag
[25]	PRI_ACC_ADDR_ERR_FLAG	Read only Acc addr error flag in spread spectrum mode
[24]	Reserved	Reserved
[23]	ACC_ADDR_ERR_MASK	Acc addr err interrupt mask in spread spectrum mode
[22]	PRI_RX_LNGTH_ERR_IRQ_MASK	Maximum received packet length err irq mask
[21]	ENDIAN	Active @nrf mode 0: little endian 1: big endian Note: when nrf_enh0 and xn297 mode should set endian to 1
[20]	TX_DONE_IER	Configured to 0, no tx_done_irq is generated
[19]	RX_DONE_IER	Configured to 0, no rx_done_irq is generated
[18]	RX_GOON	When crc error, rx will go on when this bit is set 0: rx will stop when crc error 1: rx will go on when crc error
[17]	PRI_RST	Pri softwre reset control 0: do not reset pri module 1: reset pri module
[16]	PRI_EXIT_RX	Software exit rx mode control 0: Software does not exit rx mode 1: Software exit rx mode Flag bit: rx_timeout_irq_flag
[15]	TX_DONE_IRQ_FLAG	Tx end flag bit RO (end of master state machine when valid)
[14]	RX_DONE_IRQ_FLAG	Rx end flag bit RO (end of master state machine when valid)
[13]	RX_PID_ERR_IRQ_FLAG	Rx pid error interrupt flag bit RO
[12]	RX_CRC_ERR_IRQ_FLAG	Rx crc error interrupt flag bit RO
[11]	RX_TIMEOUT_IRQ_FLAG	Rx timeout flag bit (including rx mode timeout, ack timeout, software exit rx) RO
[10]	TX_IRQ_FLAG	Tx interrupt flag bit RO (when valid, sub-state machine ends, main state machine does not end)

[9]	RX_IRQ_FLAG	Rx interrupt flag bit RO (when valid, sub-state machine ends, main state machine does not end)
[8]	IRQ_CLEAR_EN	Interrupt clear flag, when configured to 1, clear all current interrupt information
[7]	RX_CRC_ERR_IRQ_MASK	Configured to 1, the total interrupt pri_irq will not have that interrupt message
[6]	RX_PID_ERR_IRQ_MASK	Configure to 1, no rx_pid_err_irq will be generated
[5]	TX_RXACK_OUTTIME_IRQ_MASK	Configured to 1, this interrupt will not be generated
[4]	TX_IRQ_MASK	Tx_irq mask configured to 1, this interrupt will not be generated
[3]	RX_IRQ_MASK	Configured to 1, this interrupt will not be generated
[2:0]	MULTI_RX_ACC_ADDR	The pipe number of current receive packet

4.6.3 PRI_R02

Register	Offset	R/W	Description	Reset Value
pri_r02	0x4c8	R/W		0x0

Bits	Descriptions	
[31]	Reserved	Reserved
[31:16]	RX_WAIT_TIME	The maximum time to wait for ACK after PTX is converted to RX mode, beyond which the transmission is considered to have failed, an interrupt is generated, and the software resends. Normal mode: The waiting time after entering the rx state
[15]	RX_WAIT_TIME_OUT_EN	Configured to 1, normal mode rx wait timeout enable
[14:0]	TRX_TRANS_WAIT_TIME	Enhanced mode: the waiting time of PTX to RX, or RX to TX

4.6.4 PRI_R03

Register	Offset	R/W	Description	Reset Value
pri_r03	0x4cc	R/W		0x0

Bits	Descriptions	
[31:0]	RAR[31:0]	rx_addr LSB 32bits

4.6.5 PRI_R04

Register	Offset	R/W	Description	Reset Value
pri_r04	0x4d0	R/W		0x0

Bits	Descriptions	
[31:24]	RX_HEADER1	The value of rx header, active when hdr_len_exist is 1, read only When Hdr_len_numb: 11: rx_header1 is active Others：no use
[23:16]	RX_HEADER0	The value of rx header, active when hdr_len_exist is 1, read only When Hdr_len_numb: 00: no use 01: no use 10: rx_header0 is active 11: rx_header0 is active
[15]	PRI_RX_FEC	Enable rx spread spectrum
[14]	PRI_TX_FEC	Enable tx spread spectrum
[13:12]	PRI_CI_MODE	Select spread spectrum mode 2'b00-S8 2'b01-S2 2'b1x-reserved
[11]	NORMAL_M1	Normal_m1 mode, cannot be used in conjunction with enhance mode 0: disable 1: enable, no need to judge other flag bits when trx switching, just meet the switching time
[10:9]	HDR_LEN_NUMB	Payload length's locate of frame, active when pld_len_exist is 1 00: no use, must not set this value 01: length is exist and is the first byte of frame; no header 10: length is exist and is the second byte of frame; header is the first byte of frame 11: length is exist and is the third byte of frame; header is the first two bytes of frame
[8]	HDR_LEN_EXIST	Header and Length are exist 0: disable, payload length is decided by pri_r00[23:16] 1: exist, header and length are exist at frame
[7:0]	RAR[39:32]	Rx_addr MSB 8bits

4.6.6 PRI_R05

Register	Offset	R/W	Description	Reset Value
pri_r05	0x4d4	R/W		0x0

Bits	Descriptions	
[31:0]	TAR[31:0]	Tx_addr LSB 32bits

4.6.7 PRI_R06

Register	Offset	R/W	Description	Reset Value
pri_r06	0x4d8	R/W		0x00007f00

Bits	Descriptions	
[31:24]	TX_HEADER1	The value of rx header, active when hdr_len_exist is 1, read only When Hdr_len_numb: 00: no use 01: no use 10: no use 11: Tx_header1 is active
[23:16]	TX_HEADER0	The value of rx header, active when hdr_len_exist is 1, read only When Hdr_len_numb: 00: no use 01: no use 10: Tx_header0 is active 11: Tx_header0 is active
[15]	PRI_RX_MAX_CTRL_EN	The max length of rx paylad control 1: enable 0: disable Valid in dynamic payload length mode. When enabled, the receive packet length threshold is controlled by the rx_payload_length. The received payload length ≤ rx_payload_length: Can be received normally. The received payload length > rx_payload_length: Only rx_payload_length length of data is received and a length_error interrupt is generated.
[14:8]	WHITENING_INIT	The initial value of whitening(default: 0x7f)
[7:0]	TAR[39:32]	Tx_addr MSB 8bits

4.6.8 PRI_R07

Register	Offset	R/W	Description	Reset Value
pri_r07	0x4dc	R/W		0x0

Bits	Descriptions	
[31:28]	Reserved	Reserved
[27:17]	RX_RAM_START_ADDR	Rx data ram start address: 0x400
[16]	RX_RAM_READY	Rx data ram ready,must be 1 before rx receive
[15:12]	Reserved	Reserved
[11:1]	TX_RAM_START_ADDR	Tx data ram start address: 0x000
[0]	TX_RAM_READY	Tx data ram ready,must be 1 before tx send

4.6.9 PRI_R08

Register	Offset	R/W	Description	Reset Value
pri_r08	0x4e0	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR1[31:0]	Multi-channel RX mode addr1[31:0]

4.6.10 PRI_R09

Register	Offset	R/W	Description	Reset Value
pri_r09	0x4e4	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR2[31:0]	Multi-channel RX mode addr2[31:0]

4.6.11 PRI_R0A

Register	Offset	R/W	Description	Reset Value
pri_r0c	0x4e8	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR3[31:0]	Multi-channel RX mode addr3[31:0]

4.6.12 PRI_R0B

Register	Offset	R/W	Description	Reset Value
pri_r0b	0x4ec	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR4[31:0]	Multi-channel RX mode addr4[31:0]

4.6.13 PRI_R0C

Register	Offset	R/W	Description	Reset Value
pri_r0c	0x4f0	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR5[31:0]	Multi-channel RX mode addr5[31:0]

4.6.14 PRI_R0D

Register	Offset	R/W	Description	Reset Value
pri_r0d	0x4f4	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR6[31:0]	Multi-channel RX mode addr6[31:0]

4.6.15 PRI_R0E

Register	Offset	R/W	Description	Reset Value
pri_r0e	0x4f8	R/W		0x0

Bits	Descriptions	
[31:0]	ADDR7[31:0]	Multi-channel RX mode addr7[31:0]

4.6.16 PRI_R0F

Register	Offset	R/W	Description	Reset Value
pri_r0f	0x4fc	R/W		0x0

Bits	Descriptions	
[31:24]	ADDR4[39:32]	Multi-channel RX mode addr4[39:32] MSB
[23:16]	ADDR3[39:32]	Multi-channel RX mode addr3[39:32] MSB
[15:8]	ADDR2[39:32]	Multi-channel RX mode addr2[39:32] MSB
[7:0]	ADDR1[39:32]	Multi-channel RX mode addr1[39:32] MSB

4.6.17 PRI_R10

Register	Offset	R/W	Description	Reset Value
pri_r10	0x500	R/W		0x1000000

Bits	Descriptions	
[31]	ADDR_EN[7]	Enable multi-channel RX mode addr7
[30]	ADDR_EN[6]	Enable multi-channel RX mode addr6
[29]	ADDR_EN[5]	Enable multi-channel RX mode addr5
[28]	ADDR_EN[4]	Enable multi-channel RX mode addr4
[27]	ADDR_EN[3]	Enable multi-channel RX mode addr3
[26]	ADDR_EN[2]	Enable multi-channel RX mode addr2
[25]	ADDR_EN[1]	Enable multi-channel RX mode addr1
[24]	ADDR_EN[0]	Enable multi-channel RX mode addr0, enabled by default
[23:16]	ADDR7[39:32]	Multi-channel RX mode addr7[39:32] MSB
[15:8]	ADDR6[39:32]	Multi-channel RX mode addr6[39:32] MSB
[7:0]	ADDR5[39:32]	Multi-channel RX mode addr5[39:32] MSB

4.6.18 PRI_R11

Register	Offset	R/W	Description	Reset Value
pri_r11	0x504	R/W		0x0

Bits	Descriptions	
[31:26]	Reserved	Reserved
[25]	NDC_PREAMBLE_SEL	Select preamble in nrf_enh1 mode
[24]	RX_DATA_REVERSE_EN	Enable RX data inversion
[23]	PRE_SYNC_12B_EN	Enable 12bit presync
[22]	PRE_SYNC_8B_EN	Enable 8bit presync
[21]	PRE_SYNC_4B_EN	Enable 4bit presync
[20]	PRE_SYNC_ENABLE	Enable Pre sync output

[19]	250K_MODE_EN	Enable 250k rate mode
[18]	BOE_PRE_SEL	Boe preamble switch 0: 0x55 1: 0xAA
[17]	LQI_EN_FOR_SYNC	LQI sync function enable control 0: disable 1: enable Note: just enable when addr is 2 byte
[16:15]	RX_PID_MANUAL	Rx pid configured externally by user
[14:13]	TX_PID_MANUAL	Tx pid configured externally by user
[12]	PRE_2BYTE_MODE	preamble extended control 0: disable 1: enable Note: when enable 297-preamble repeat twice. When 250K mode is used, it must set to 1.
[11]	TX_DATA_REVERSE_EN	Enable TX data bitstream inversion
[10:9]	SCR_INI[8:7]	Scr init initial value [8:7]
[8:6]	PRE_LEN[2:0]	Preamble length control 0:2B 1:3B 2:4B 3:6B 4:8B 5:12B 6:16B 7:24B Need to set 0x504[12] to 1, Pre_2byte_mode
[5]	BOE_MODE_EN	Enable Boe crc scr polynomial
[4:2]	ADDR_ERR_THR[2:0]	The allowed number of error bit during access addr matching 0: exactly matched
[1]	PID_MANUAL_EN	Configured to 1, allow the tx_pid_manual and rx_pid_manual configured externally by user.
[0]	ADDR_SCR_DIS	The whitening of addr control 0: enabled, whitening includes addr, header, payload 1: disabled, whitening includes header, payload

4.7 Instructions for use

4.7.1 Normal mode

4.7.1.1 PTX mode

1. Initialization

a. Clock initialization, select XTH as clock source, enable BLE_32M/BLE_32K clock.

b. Interrupt enabled, configure interrupt service function.

c. Memory initialized, emngr_init()；

d. RF LDO enabled, ana_init()；

e. LL common initialized, llhwc_cmn_init()；

f. Phy configuration：PHY2_PHY_DRV_SEQ_TX_STRT/END_ADDR；
PHY2_PHY_DRV_SEQ_RX_STRT/END_ADDR

2. TX initialization：

a. Prepare tx data , address: 0x50028000+PRI_R07_TX_RAM_START_ADDR

b. Configure PRI_R00_CHIP_MODE: 2, XN297L; 3,NRF

c. Configure PRI_R00_ENHANCE: 0, Normal mode

d. Configure operating bandwidth PRI_R00_BW_MODE: 0, 1M; 1, 2M

e. Configure the device address width PRI_R00_ADDR_BYTE_LEN: 0,2Bytes; 1,3bytes; 2, 4bytes; 3 5bytes

f. Configure the device address high byte PRI_R06_TAR_MSB, only used in 5bytes address width

g. Configure the device address low bytes PRI_R05_TAR_LSB

h. Configure the TX packet start relative address PRI_R07_TX_RAM_START_ADDR, base address 0x50028000

i. Configure the TX packet length PRI_R00_TX_PAYLOAD_LEN, package length refers to the protocol requirements

j. Configure whether do the TX packet whitening PRI_R00_SCR_EN: 0, De-whitening;1, Whitening

k. Configure whether to enable CRC PRI_R00_CRC_EN, if CRC is enabled, configure the length PRI_R00_CRC_SEL16: 0, 8bits; 1,16bits

3. TX starting

a. Configure tx mode PRI_R00_PRI_RX: 0

b. Configure tx active,RAM PRI_R07_TX_RAM_READY

4.7.1.2 PRX mode

1. **Initialization：same as 4.7.1.1**
2. **RX initialization：Note that NRF mode should not configure PRI_R00_SCR_EN**
   a. Configure PRI_R00_CHIP_MODE: 2, XN297L; 3,NRF
   b. Configure PRI_R00_ENHANCE: 0, Normal mode
   c. Configure operating bandwidth PRI_R00_BW_MODE: 0, 1M； 1, 2M
   d. Configure the device address width PRI_R00_ADDR_BYTE_LEN: 0,2Bytes; 1,3bytes;2, 4bytes; 3 5bytes
   e. Configure the device address high byte PRI_R04_RAR_MSB, only used in 5bytes address width
   f. Configure the device address low bytes PRI_R03_RAR_LSB
   g. Configure the TX packet start relative address PRI_R07_RX_RAM_START_ADDR, base address 0x50028000
   h. Configure the TX packet length PRI_R00_RX_PAYLOAD_LEN, the packet length same as the PTX
   i. Configure whether do the TX packet whitening PRI_R00_SCR_EN: same as the PTX
   j. Configure whether to enable CRC PRI_R00_CRC_EN, if CRC is enabled, configure the length PRI_R00_CRC_SEL16: 0, 8bits; 1,16bits
3. **RX starting**
   a. Configure rx mode, PRI_R00_PRI_RX: 1
   b. Packet receive timeout enable PRI_R02_RX_WAIT_TIME_OUT_EN: 0, Enable; 1, Disable. If enabled, PRI_R02_RX_WAIT_TIME configures the timeout(Unit: us).
   c. Configure rx active, RAM PRI_R07_RX_RAM_READY

4.7.2 Enhanced mode

4.7.2.1 PTX mode

1. **Initialization: same as 4.7.1.1**
2. **TX initialization：Note that NRF mode should not configure PRI_R00_SCR_EN**

   a~k: Configured as 4.7.1.1(except for step c PRI_R00_ENHANCE configured to 1)

   l. NRF Mode Model Selection Register PRI_R00[7]: 0,nrf_enh0;1,nrf_enh1

   XN297L mode does not need this configuration.

m. Configure the peer device address high byte PRI_R04_RAR_MSB, only used in 5bytes address.

n. Configure the peer device address low bytes PRI_R03_RAR_LSB.

o. Configure whether to receive ACK, PRI_R00_TX_NOACK_EN: 0, Do not receive； 1, Receive. If received, perform the following steps p~t.

p. Configure RX packet SRAM ready PRI_R07_RX_RAM_READY.

q. Configure dynamic parsing of RX packet length PRI_R00_DPY_EN, must be enabled.

r. Configure the RX packet start address PRI_R07_RX_RAM_START_ADDR

s. Configure the TRX transition time PRI_R02_TRX_TRANS_WAIT_TIME

t. Configure timeout enable PRI_R02_RX_WAIT_TIME_OUT_EN and RX packet timeout PRI_R02_RX_WAIT_TIME

3. TX starting

a. Configure tx mode PRI_R00_PRI_RX: 0

b. Configure tx active,RAM PRI_R07_TX_RAM_READY

4.7.2.2 PRX mode

1. Initialization：same as 4.7.1.1

2. RX initialization：Note that NRF mode should not configure PRI_R00_SCR_EN

a~j: Configured as 4.7.1.2 (except for step b PRI_R00[7] configured to 1)

k. NRF enhanced mode selection PRI_R00[7], 0,nrf_enh0; 1,nrf_enh1
XN297L mode does not need this configuration.

l. Configure the TRX transition time PRI_R02_TRX_TRANS_WAIT_TIME, the sum of this time and the PHY startup shutdown time is the actual TRX transition time.

m. Configure the peer device address high byte PRI_R06_TAR_MSB, only used in 5bytes address.

n. Configure the peer device address low bytes PRI_R05_TAR_LSB

o. If ACK with payload, enable PRI_R00_RX_ACK_PAYLOAD_EN, and perform the following step p. Additional note: Step p can be configured before receiving the packet or in the Interrupt Service Program after receiving the packet.

p. Fill TX SRAM TX packet data, configure the start address PRI_R07_TX_RAM_START_ADDR and the length PRI_R00_TX_PAYLOAD_LEN.

3. RX starting

a. Configure rx mode, PRI_R00_PRI_RX: 1

b. Packet receive timeout enable PRI_R02_RX_WAIT_TIME_OUT_EN: 0, Enable; 1, Disable. If enabled, PRI_R02_RX_WAIT_TIME configures the timeout(Unit: us).

c. Configure rx active, RAM PRI_R07_RX_RAM_READY

4.7.3 Normal_m1 mode

4.7.3.1 PTX mode

1. **Initialization: same as 4.7.1.1**
2. **TX initialization**

a~k: Configured as 4.7.1.1(except for step c PRI_R01_NORMAL_M1 configured to 1)

l. Configure the peer device address high byte PRI_R04_RAR_MSB, only used in 5bytes address

m. Configure the peer device address low bytes PRI_R03_RAR_LSB

n. Configure RX packet SRAM ready PRI_R07_RX_RAM_READY

o. Configure the RX packet start address PRI_R07_RX_RAM_START_ADDR

p. Configure the TRX transition time PRI_R02_TRX_TRANS_WAIT_TIME

q. Configure timeout enable PRI_R02_RX_WAIT_TIME_OUT_EN and RX packet timeout PRI_R02_RX_WAIT_TIME

3. **TX starting**

a. Configure tx mode PRI_R00_PRI_RX: 0

b. Configure tx active, RAM PRI_R07_TX_RAM_READY

4.7.3.2 PRX mode

1. **Initialization：same as 4.7.1.1**

a~j. Configured as 4.7.1.2 (except for step b PRI_R01_NORMAL_M1 configured to 1 also)

k. Configure the TRX transition time PRI_R02_TRX_TRANS_WAIT_TIME, the sum of this time and the PHY startup shutdown time is the actual TRX transition time.

l. Configure the peer device address high byte PRI_R06_TAR_MSB, only used in 5bytes address

m. Configure the peer device address low bytes PRI_R05_TAR_LSB

n. Fill TX SRAM TX packet data, configure the start address PRI_R07_TX_RAM_START_ADDR and the length PRI_R00_TX_PAYLOAD_LEN

2. **RX starting**

a. Configure rx mode, PRI_R00_PRI_RX: 1

b. Packet receive timeout enable PRI_R02_RX_WAIT_TIME_OUT_EN: 0, Enable; 1, Disable. If enabled, PRI_R02_RX_WAIT_TIME configures the timeout(Unit: us).

c. Configure rx active, RAM PRI_R07_RX_RAM_READY

4.7.4 250k mode

4.7.4.1 PTX mode

1. **Initialization: same as 4.7.1.1**
2. **TX initialization**

a. Prepare tx data , address: 0x50028000+PRI_R07_TX_RAM_START_ADDR

b. Configure PRI_R00_CHIP_MODE: 3,NRF

c. Configure PRI_R00_ENHANCE: 0, Normal mode

d. Configure operating bandwidth PRI_R00_BW_MODE：0, 1M

e. SRAM populate data, address: 0x50028000+PRI_R07_TX_RAM_START_ADDR

f. Configure PRI_R00_CHIP_MODE: 3,NRF

g. Configure PRI_R00_ENHANCE: 0, Normal mode

h. Configure operating bandwidth PRI_R00_BW_MODE：0, 1M；

i. Configure the device address width PRI_R00_ADDR_BYTE_LEN：0,2Bytes, 1,3bytes;2, 4bytes; 3 5bytes

j. Configure the device address high byte PRI_R06_TAR_MSB, only used in 5bytes address width

k. Configure the device address low bytes PRI_R05_TAR_LSB

l. Configure the rate mode PRI_R11_250K_MODE_EN: 1, enable 250k rate mode

m. Configure the TX packet start relative address PRI_R07_TX_RAM_START_ADDR, base address 0x50028000

n. Configure the TX packet length PRI_R00_TX_PAYLOAD_LEN, package length refers to the protocol requirements

o. Configure whether do the TX packet whitening PRI_R00_SCR_EN：0, De-whitening;1, Whitening

p. Configure whether to enable CRC PRI_R00_CRC_EN, if CRC is enabled, configure the length PRI_R00_CRC_SEL16: 0, 8bits; 1,16bits

4.7.4.2 PRX mode

1. **Initialization: same as 4.7.1.1**

2. RX initialization：

a. Configure PRI_R00_CHIP_MODE: 3, NRF

b. Configure PRI_R00_ENHANCE: 0, Normal mode

c. Configure operating bandwidth PRI_R00_BW_MODE: 0, 1M；

d. Configure the device address width PRI_R00_ADDR_BYTE_LEN: 0,2Bytes, 1,3bytes;2, 4bytes; 3 5bytes

e. Configure the device address high byte PRI_R04_RAR_MSB, only used in 5bytes address width

f. Configure the device address low bytes PRI_R03_RAR_LSB

g. Configure the rate mode PRI_R11_250K_MODE_EN: 1, enable 250k rate mode

h. Configure the TX packet start relative address PRI_R07_RX_RAM_START_ADDR, base address 0x50028000

i. Configure the TX packet length PRI_R00_RX_PAYLOAD_LEN, the packet length same as the PTX

j. Configure whether do the TX packet whitening PRI_R00_SCR_EN: same as the PTX

k. Configure whether to enable CRC PRI_R00_CRC_EN, if CRC is enabled, configure the length PRI_R00_CRC_SEL16: 0, 8bits; 1,16bits

3. RX starting

a. Configure rx mode, PRI_R00_PRI_RX: 1

b. Packet receive timeout enable PRI_R02_RX_WAIT_TIME_OUT_EN: 0, Enable; 1, Disable. If enabled, PRI_R02_RX_WAIT_TIME configures the timeout(Unit: us).

c. Configure rx active, RAM PRI_R07_RX_RAM_READY

4.7.5 Interrupt Service Program

The time point of interrupt reporting comes at the same time as the hardware post_tx and post_rx, so after entering the interrupt, the software is prohibited from reading or writing the registers of the PHY module during Tpost_txs, Tpost_rx. The above time can be calculated by software.

4.7.5.1 PTX mode

After clearing the corresponding interrupt, you need to pull down PRI_R07_TX_RAM_READY, For Enhanced mode, you also need to pull down PRI_R07_RX_RAM_READY.

4.7.5.2 PRX mode

After clearing the corresponding interrupt, you need to pull down PRI_R07_RX_RAM_READY.

Abbreviation

A	
APB	Advanced Peripheral Bus
ADC	Analog-to-Digital Converter
ALT	Alternate Function Select
ANAC	Analog Control
APROM	Application ROM
ATT	Attribute Protocol
B	
BAUDR	Baud Rate Select Register
BLDC	Brushless Direct Current Motor
Bluetooth LE	Bluetooth Low Energy
BOD	Brown-out Detector
BOM	Bill of Materials
C	
CFGx	Configuration Register for Channel x
ChEnReg	DMA Channel Enable Register
CMPDAT	Compare value
CPU	Central Processing Unit
CRC-32	Cyclic Redundancy Check
CTLx	Control Register for Channel x
CTRLR0	Control Register 0
CTRLR1	Control Register 1
D	
DARx	Destination Address Register for Channel x
Desiccant	A material for adsorbing moisture while remaining dry
DFBA	Data Flash Base Address Register
DLF	Divisor Latch Fraction Register
DLH	Divisor Latch High
DLL	Divisor Latch Low
DmaCfgReg	DMA Configuration Register
DMACR	DMA Control Register
DmaIdReg	DMA ID Register
DMARDLR	DMA Receive Data Level Register
DMASA	DMA Software Acknowledge
DMATDLR	DMA Transmit Data Level Register
DR	Data Register
DSTATARx	Destination Status Address Register for Channel x
DSTATx	Destination Status Register for Channel x
E	
ESD	Electro-Static discharge
F	
FCR	FIFO Control Register
FIFO	First Input First Output
FMC	Flash Memory Controller
Floor life	The longest time that the HiSilicon product is allowed to remain in the workshop (environ-ment <30 °C / 60% RH, before unpacking the moisture-proof packaging to reflow).
G	
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPIO	General-purpose I/O
H	
HCLK	Tstem Clock
HIC	Humidity Indicator Card
HID	Human Interface Device
HIRC	32MHz internal high speed osc-illator
HTX	Halt TX
HXT	32MHz external high speed crystal oscillator
I	
I2C	Inter-Integrated Circuit
IAP	In-Application-Programming
ICE	In-Circuit-Emulator
ICP	In-Circuit Programming
ICR	Interrupt Clear Register
IDR	Identification Register
IER	Interrupt Enable Register
IIR	Interrupt Identity Register
IMR	Interrupt Mask Register
IRSR	Interrupt Raw Status Registers
ISB	Instruction Synchronization Barrier
ISP	In-System Programming
ISR	Interrupt Service Routine
L	
L2CAP	Logical Link Control and Adaptation Protocol
LCR	Line Control Register
LCR_EXT	Line Extended Control Register
LDO	Low dropout regulator
LDROM	Loader ROM
LIRC	32 kHz internal low speed RC oscillator
LNA	Low Noise Amplifier

LSB	Least significant bit
LSR	Line Status Register
LstDstReg	Last Destination Transaction Request Register
LstSrcReg	Last Source Transaction Request Register
LVR	Low Voltage Reset
LXT	32.768 kHz external low speed crystal oscillator
M	
MBB	Moisture Barrier Bag
MCR	Modem Control Register
MCU	Micro Control Unit
MDM	Mobile Device Management
MFP	Multiple Function Port
MISO	Master input slave output
MOSI	Master output slave input
MSB	Most Significant Bit
MSL	Moisture sensitivity level, this product is on level 3
MSR	Modem Status Register
MSTICR	Multi-Master Interrupt Clear Register
N	
NMI	Non Maskable Interrupt
NVIC	Nested Vectored Interrupt Controller
P	
PA	Power Amplifier
PLL	Phase Locked Loop
POR	Power-on Reset
PWM	Pulse Width Modulation
R	
RAR	Receive Address Register
RBR	Receive Buffe Register
ReqDstReg	Destination Software Transaction Request Register
ReqSrcReg	Source Software Transaction Request Register
RF	Radio frequency
RFL	Receive FIFO Level
RISR	Raw Interrupt Status Register
ROM	Read-Only Memory
RSSI	Received Signal Strength Indication
RTOS	Real Time Operating System
RXFLR	Receive FIFO Level Register
RXFTLR	Receive FIFO Threshold Level Register
RXOICR	Receive FIFO Overflow Interrupt Clear Register
RXUICR	Receive FIFO Underflow Interrupt Clear Register
S	
SARx	Source Address Register for Channel x Register
SCB	System Control Block Registers
SCB	System Control Block Registers
SCR	Scratchpad Register
SER	Slave Enable Register
SglReqDstReg	Single Destination Transaction Request Register
SglReqSrcReg	Single Source Transaction Request Register
Shelf Life	Normal storage time after moisture-proof packaging
SM	Security Manager
SoC	System on chip
SPI	Serial Peripheral Interface
SPROM	Security protection ROM
SR	Status Register
SRAM	Static random access memory
SSTATARx	Source Status Address Register for Channel x
SSTATx	Source Status Register for Channel x
Statusint	Combined Interrupt Status Register
SWD	Serial Wire Debug
SysTick	System Timer
T	
TAR	Transmit Address Register
TFL	Transmit FIFO Level
THR	Transmit Holding Register
THRE	Transmitter Holding Register Empty
TMR	Timer Controller
TXFLR	Transmit FIFO Level Register
TXFTLR	Transmit FIFO Threshold Level Register
TXOICR	Transmit FIFO Overflow Interrupt Clear Register
U	
UART	Universal Asynchronous Receiver/Transmitters
USR	UART Status Register
W	
WDT	Watchdog Timer
WWDT	Window Watchdog Timer

Revision History

Version	Date	Content
1.0	Jun. 2024	Initial
1.1	Aug. 2024	Add the MSOP10.
1.2	May. 2025	Update the register's name of UART in UART Register Map. Update the Key Features. Update the 3.4.7.2. Add Timing Parameters. Update the Figure 3-59, Figure 3-62, Figure 3-92, Figure 3-93, Figure 3-94, Figure 3-95, Figure 3-96, Figure 3-97. Update the USB Register Map and USB Register Description.

USING THIS DOCUMENT

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure the accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

TRADEMARKS

Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

DISCLAIMER

All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

Contact Us

Shanghai Panchip Microelectronics Co., Ltd.
The 302 Room of Building D, No. 666 Shengxia Road
Zhangjiang Hi-Tech Park, Shanghai
People's Republic of China

021-50802371

http://www.panchip.com