PANCHIP

PAN108 series

User Manual

V1.9 Apr. 2024

Panchip Microelectronics Co., Ltd.

BLE SoC Transceiver

General Description

PAN108 series integrates BLE5.3 and 2.4GHz dual-mode wireless SoC transceiver. The wireless transceiver circuit works in the 2.400-2.483GHz universal ISM frequency band. There is an external 512KB/1MB Flash program memory and a built-in 64KB SRAM memory. In addition, PAN108 series is equipped with a wealth of peripherals, including up to 48 GPIOs, 24-channel PWM, three 32-bit timers, 1 I2C, 2 UARTs, 2 SPIs, 8 external channels ADC, WDT, WWDT, I2S master, I2S slave, USB2.0(Full Speed), 32K RC automatic calibration, QDEC and automatic key-scan, etc. PAN108 series is suitable for wireless mouse and keyboard, smart home and elec-tronic shelf label, BLE-AOA indoor locating.

Key Features

- **MCU**
  - 32-bit MCU core running up to 64MHz
- **Memory**
  - Build-in 512KB/1MB flash supporting deep sleep mode
  - 64KB SRAM
  - 256B eFuse
  - 4 KB cache
- **Low Power**
  - Active mode RX: 5.6mA(DCDC)
  - Active mode TX at 0dBm: 6.1mA(DCDC)
  - Standby mode : 0.34uA
  - Standby mode(SRAM retention): 2uA(GPIO, XTL, RCL can wake up)
  - Deep sleep mode: 8uA(All Logic Retention, GPIO, XTL, RCL can wake up)
- **Clock**
  - 32MHz RC
  - 32MHz XTAL
  - 32kHz RC
  - 32.768kHz XTAL
  - DPLL(Two channels: 64MHz/48MHz and 48MHz (USB 2.0))
- **RF**
  - Mode
    - BLE5.3 modes: 1Mbps, 2Mbps, 500kbps, 125kbps
    - 2.4G private protocol: 1Mbps / 2Mbps, supporting hardware ACK
  - Output power: -45dBm~7dBm
  - Receiver
    - -100dBm@125Kbps
    - -99dBm@500Kbps
    - -96dBm@1Mbps
    - -93dBm@2Mbps
  - RSSI
    - Resolution: 0.25dB
    - Accuracy: ±2dB
    - Range: - 90dbm ~ -15dBm
  - Positioning: AoA/AoD supported
  - Single antenna supported
  - Safety regulations: BQB / ETSI / FCC
- **Peripheral**
  - Up to 48 GPIOs (there are two power supply voltages)
  - 24-channel PWM
  - Three 32-bit timer
  - One I2C
  - Two UARTs
  - Two SPIs
  - 3-channel DMA
  - 11-channel ADC(8 ext, bandgap, VDD/4, temp)
  - Two I2S(one I2S master and one I2S slave)
  - 3-channel QDEC
  - WDT / WWDT
  - ECC accelerator
  - Automatic key-scan
  - 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℃ ~ 85℃
  - Accuracy: ±2℃ (With calibration)
- **Power Management**
  - Integrated voltage regulator
  - Operating voltage: 1.8V to 3.7V (Support DCDC)
- **Package**
  - LQFP64（7×7mm）
  - QFN48（6×6mm） / QFN32（5×5mm）
- **Operating Condition**
  - Operating temperature: -40℃ ~ 85℃ / -40℃ ~ 125℃ / -40℃ ~ 105℃
  - Storage temperature: -60℃~150℃
  - ESD
    - HBM: ±2.5kV(LQFP64) / ±5kV(QFN48)/ ±4kV(QFN32)
    - MM: ±250V(LQFP64) / ±250V(QFN48)/ ±300V(QFN32)
    - CDM: ±500V
  - Latch-up: ±500mA

Typical Applications

- High-precision BLE-AOA in-door location system
- Electronic Shelf Label
- Wireless mouse and keyboard
- LED light control

Bluetooth Features

Bluetooth Low Energy Controller

The PAN108 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, Scanning, Initiating and Connection (both of Central and Periph-eral) role**
- **Up to 10 Link Layer state machines concurrently:**
  - 1 * Passive Scanning
  - 1 * Non-connectable advertising
  - 8 * Any other combinations (Legacy/Extended/Periodic Adverstising, Scanning and Connection)
- **Support LE Features:**
  - LL Encryption
  - LE Data Packet Length Extension
  - LL Privacy
  - Extended Scanner Filter Policies
  - LE Extended and Periodic Adverstising
  - Channel Selection Algorithm #2
  - Constant Tone Extension
- **Support Update Channel Statistics**

Bluetooth Host

- **Generic Access Profile (GAP) with all possible LE roles**
  - Peripheral & Central
  - Observer & 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

Bluetooth Mesh

- **Compatible with Bluetooth SIG Mesh Profile 1.0.1**
- **Support Mesh Provisioning**
- **Provisioner: PB-ADV**
- **Provisionee: PB-ADV, PB-GATT and PB-Remote**
- **Support Mesh Node Feature : Relay, Proxy, Friend, LPN**
- **Support Mesh Models**
  - SIG Models: Config Model, Health Model and Generic Models (Onoff and Light Con-trol Models)
  - SIG Developing Models: PB-Remote Model and SIG OTA Model
- **Support multiple smart speakers control concurrently for BaiDu Xiaodu, Alibaba Ali-genie and Amazon Echo**
- **Support network control: HeartBeat, Subnet, Secure Beacon and Group Control**
- **Support switch control for over 256 nodes without delay**
- **Mesh Security**
  - Provisioning: FIPS P-256 Elliptic Curve
  - Message: AES-CCM Encryption
  - Network: SEQ Control, IV Index and Key Fresh

Proprietary Radio 2.4G Features

- Support 1M and 2M PHY
- XN297L,PAN1026 Transceiver protocol compliant
- Support No Acknowledge, Acknowledge and Acknowledge with payload
- Support CRC8, CRC16 and CRC24
- Support whitening

Content

List of Tables

List of Figures

1 Block Diagram

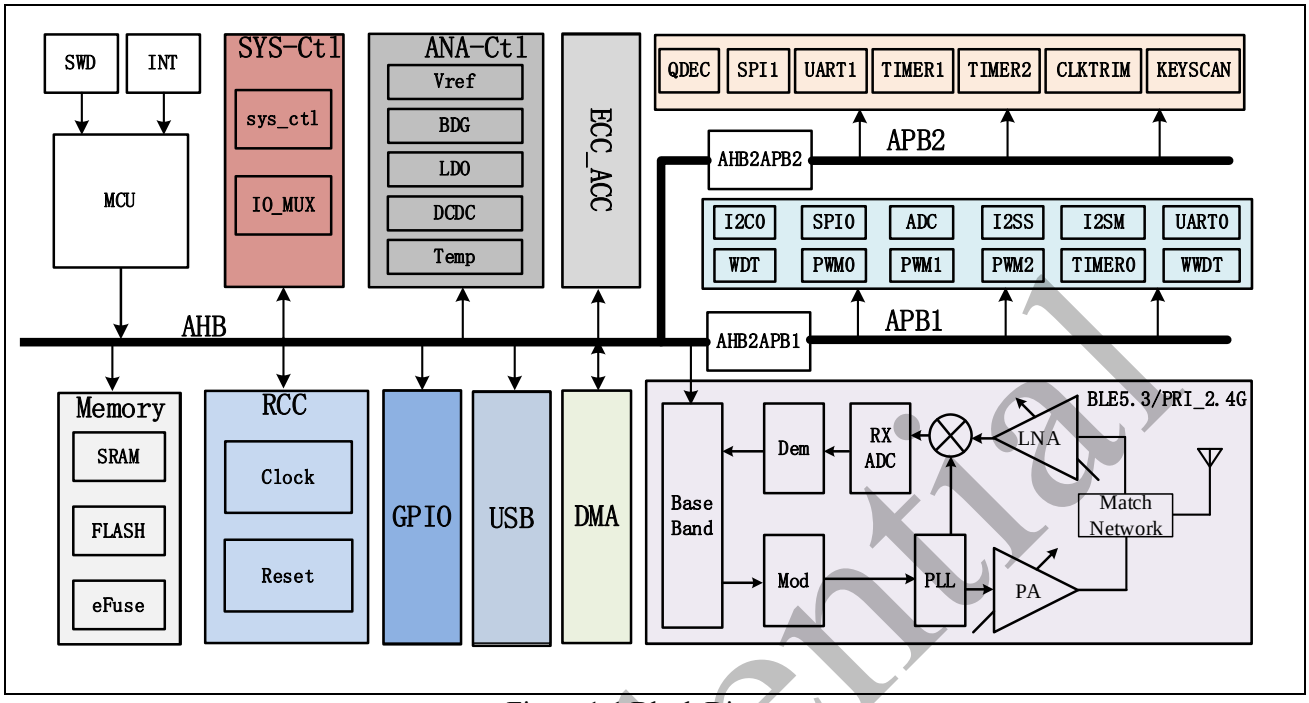


Figure 1-1 Block Diagram

2 Pin Information

2.1 QFN 32-PIN Diagram

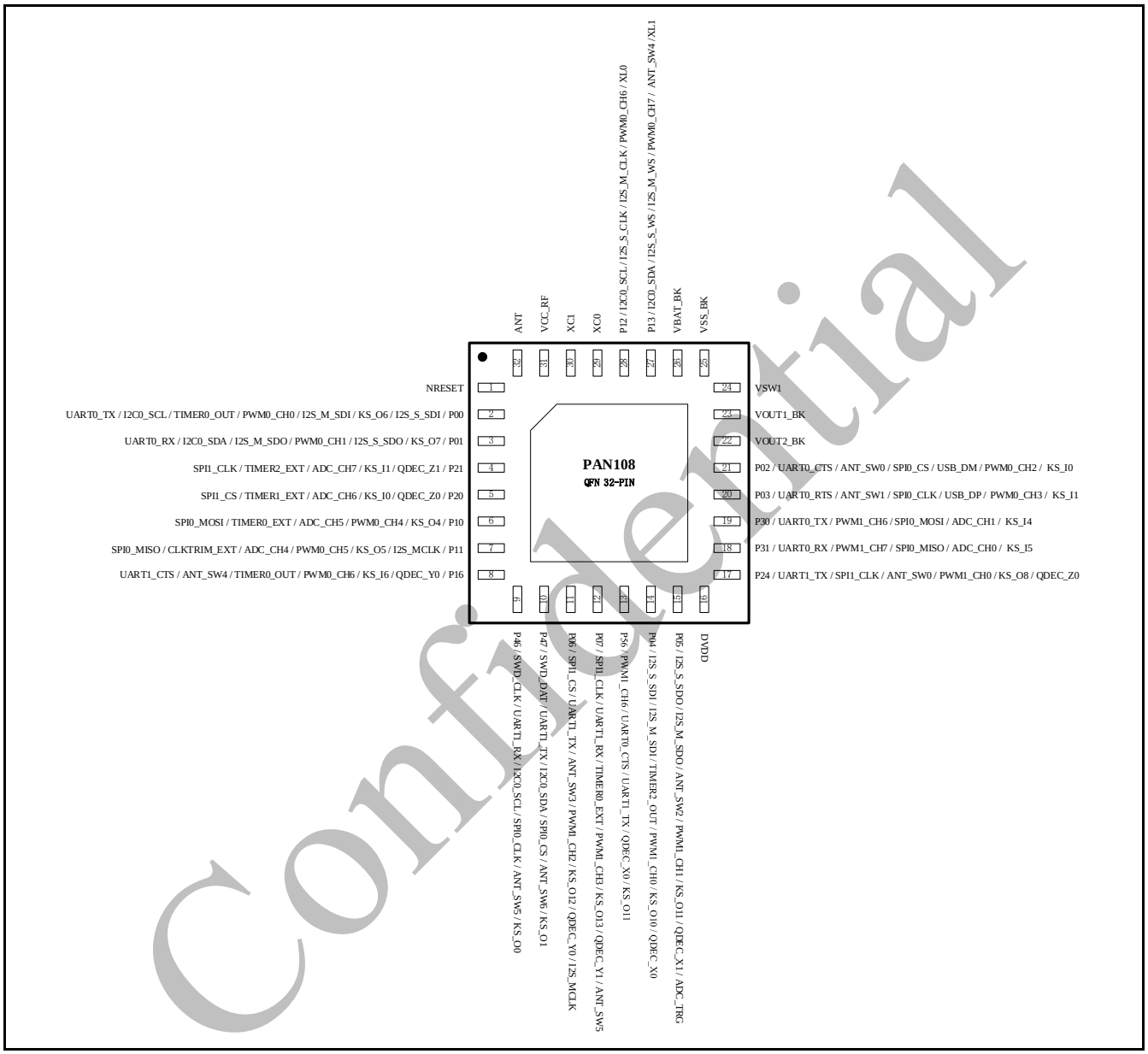


Figure 2-1 QFN 32-PIN Diagram

2.2 QFN 48-PIN Diagram

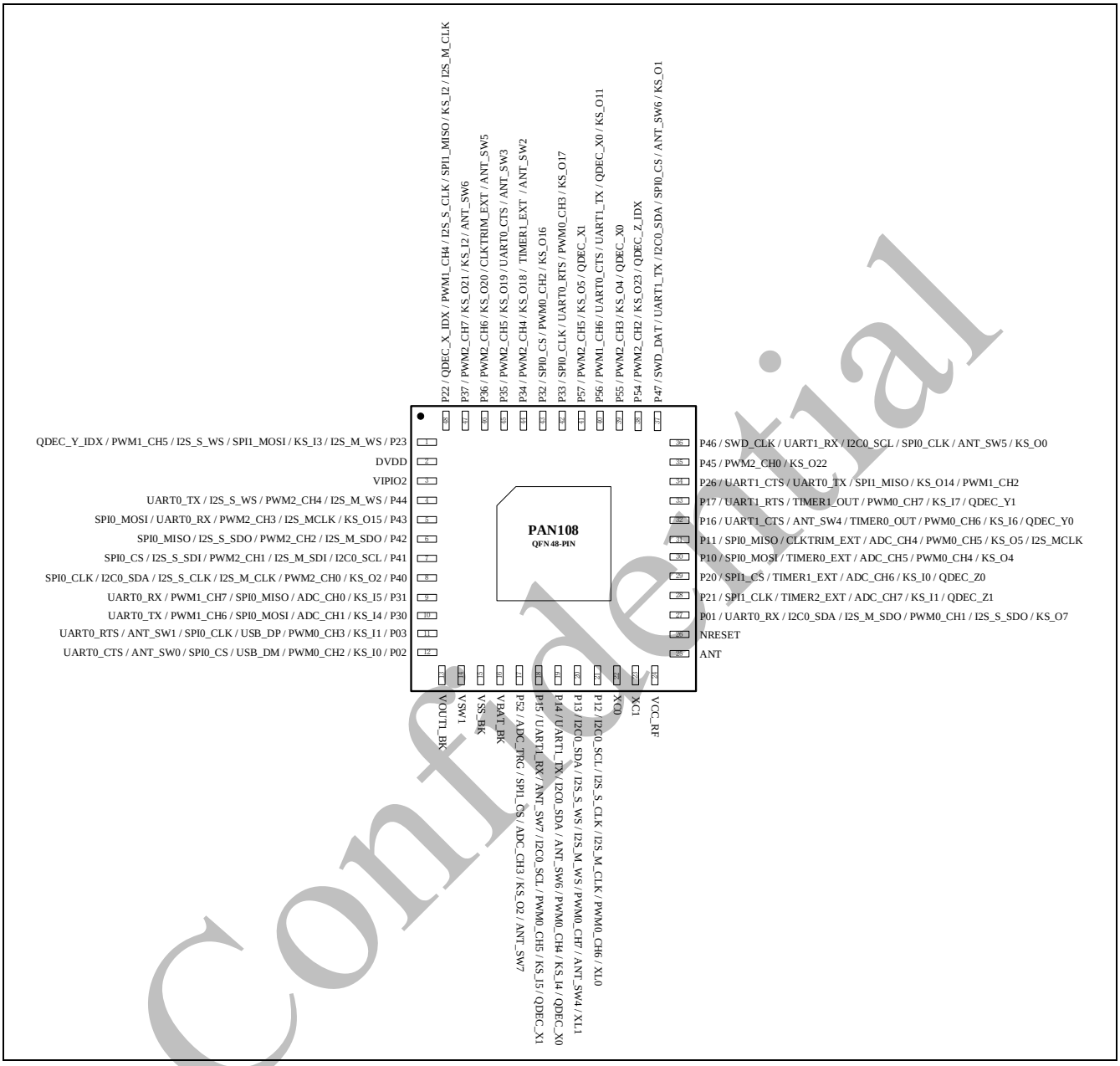


Figure 2-2 QFN 48-PIN Diagram

2.3 LQFP 64-PIN Diagram

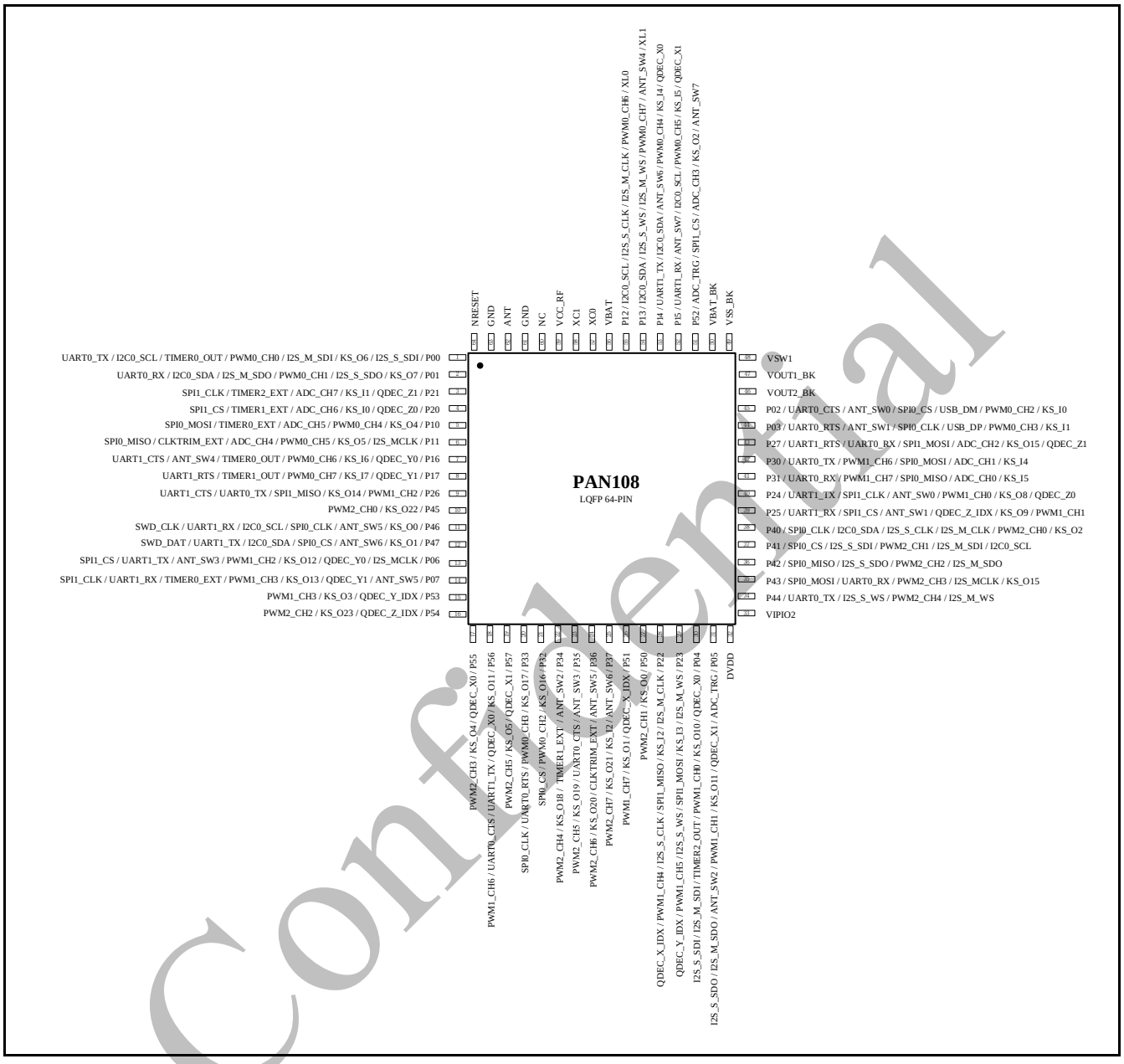


Figure 2-3 LQFP 64-PIN Diagram

2.4 Pin Descriptions

Detail pin descriptions see Table 2-1.

Table 2-1 Pin Descriptions

Package			Pin Name	Pin Type	Description
QFN 32	QFN48	LQFP64			
2	-	1	P00	I/O	General-purpose digital input and output pin
			UART0_TX	O	Uart0 tx pin
			I2C0_SCL	I/O	I2c0 clock pin
			TIMER0_ OUT	O	Timer0 output pin
			PWM0_CH0	O	Pwm0 channel 0 output pin
			I2S_M_SDI	I	I2s master data input pin
			KS_O6	O	Keyscan output channel 6 pin
			I2S_S_SDI	I	I2s slave data input pin
3	27	2	P01	I/O	General-purpose digital input and output pin
			UART0_RX	I	Uart0 rx pin
			I2C0_SDA	I/O	I2c0 data pin
			I2S_M_SDO	O	I2s master data output pin
			PWM0_CH1	O	Pwm0 channel 1 output pin
			I2S_S_SDO	O	I2s slave data output pin
			KS_O7	O	Keyscan output channel 7 pin
4	28	3	P21	I/O	General-purpose digital input and output pin
			SPI1_CLK	I/O	Spi1 clock pin
			TIMER2_EXT	I	Timer2 external input pin
			ADC_CH7	AI	Adc channel 7 pin
			KS_I1	I	Keyscan input channel 1 pin
			QDEC_Z1	I	Qdec z input channel 1 pin
5	29	4	P20	I/O	General-purpose digital input and output pin
			SPI1_CS	I/O	Spi1 cs pin
			TIMER1_EXT	I	Timer1 input pin
			ADC_CH6	AI	Adc channel 6 pin
			KS_I0	I	Keyscan input channel 0 pin
			QDEC_Z0	I	Qdec z input channel 0 pin
6	30	5	P10	I/O	General-purpose digital input and output pin

			SPI0_MOSI	I/O	Spi0 mosi pin
			TIMER0_EXT	I	Timer0 external input pin
			ADC_CH5	AI	Adc channel 5 pin
			PWM0_CH4	O	Pwm0 channel 4 output pin
			KS_O4	O	Keyscan output channel 4 pin
7	31	6	P11	I/O	General-purpose digital input and output pin
			SPI0_MISO	I/O	Spi0 miso pin
			CLKTRIM_EXT	I	CLKTRIM external clock measurement input pin
			ADC_CH4	AI	Adc channel 4 pin
			PWM0_CH5	O	Pwm0 channel 5 output pin
			KS_O5	O	Keyscan output channel 5 pin
			I2S_MCLK	O	I2s output sample clock
8	32	7	P16	I/O	General-purpose digital input and output pin
			UART1_CTS	I	Uart1 cts pin
			ANT_SW4	O	Antenna switch 4 pin
			TIMER0_OUT	O	Timer0 output pin
			PWM0_CH6	O	Pwm0 channel 6 output pin
			KS_I6	I	Keyscan input channel 6 pin
			QDEC_Y0	I	Qdec y input channel 0 pin
-	33	8	P17	I/O	General-purpose digital input and output pin
			UART1_RTS	O	Uart1 rts pin
			TIMER1_OUT	O	Timer1 output pin
			PWM0_CH7	O	Pwm0 channel 7 output pin
			KS_I7	I	Keyscan input channel 7 pin
			QDEC_Y1	I	Qdec y input channel 1 pin
-	34	9	P26	I/O	General-purpose digital input and output pin
			UART1_CTS	I	Uart1 cts pin
			UART0_TX	O	Uart0 tx pin
			SPI1_MISO	I/O	Spi1 miso pin
			KS_O14	O	Keyscan output channel 14 pin
			PWM1_CH2	O	Pwm1 channel 2 output pin
-	35	10	P45	I/O	General-purpose digital input and output pin
			PWM2_CH0	O	Pwm2 channel 0 output pin

			KS_O22	O	Keyscan output channel 22 pin
9	36	11	P46	I/O	General-purpose digital input and output pin
			SWD_CLK	I	Swd clock input pin
			UART1_RX	I	Uart1 tx pin
			I2C0_SCL	I/O	I2c0 scl pin
			SPI0_CLK	I/O	Spi0 clock pin
			ANT_SW5	O	Antenna switch 5 pin
			KS_O0	O	Keyscan output channel 0 pin
10	37	12	P47	I/O	General-purpose digital input and output pin
			SWD_DAT	I/O	Swd data input output pin
			UART1_TX	O	Uart1 tx pin
			I2C0_SDA	I/O	I2c0 sda pin
			SPI0_CS	I/O	Spi0 cs pin
			ANT_SW6	O	Antenna switch 6 pin
			KS_O1	O	Keyscan output channel 1 pin
11	-	13	P06	I/O	General-purpose digital input and output pin
			SPI1_CS	I/O	Spi1 cs pin
			UART1_TX	O	Uart1 tx pin
			ANT_SW3	O	Antenna switch 3 pin
			PWM1_CH2	O	Pwm1 channel 2 output pin
			KS_O12	O	Keyscan output channel 12 pin
			QDEC_Y0	I	Qdec y input channel 0 pin
			I2S_MCLK	O	I2s output sample clock
12	-	14	P07	I/O	General-purpose digital input and output pin
			SPI1_CLK	I/O	Spi1 clock pin
			UART1_RX	I	Uart1 rx pin
			TIMER0_EXT	I	Timer0 external input pin
			PWM1_CH3	O	Pwm1 channel 3 output pin
			KS_O13	O	Keyscan output channel 13 pin
			QDEC_Y1	I	Qdec y input channel 1 pin
			ANT_SW5	O	Antenna switch 5 pin
-	-	15	P53	I/O	General-purpose digital input and output pin
			PWM1_CH3	O	Pwm1 channel 3 output pin

			KS_O3	O	Keyscan output channel 3 pin
			QDEC_Y_IDX	I	Qdec y_idx input pin
-	38	16	P54	I/O	General-purpose digital input and output pin
			PWM2_CH2	O	Pwm2 channel 2 output pin
			KS_O23	O	Keyscan output channel 23 pin
			QDEC_Z_IDX	I	Qdec z_idx input pin
-	39	17	P55	I/O	General-purpose digital input and output pin
			PWM2_CH3	O	Pwm2 channel 3 output pin
			KS_O4	O	Keyscan output channel 4 pin
			QDEC_X0	I	Qdec x input channel 0 pin
13	40	18	P56	I/O	General-purpose digital input and output pin
			PWM1_CH6	O	Pwm1 channel 6 output pin
			UART0_CTS	I	Uart0 cts pin
			UART1_TX	O	Uart1 tx pin
			QDEC_X0	I	Qdec x input channel 0 pin
			KS_O11	O	Keyscan output channel 11 pin
-	41	19	P57	I/O	General-purpose digital input and output pin
			PWM2_CH5	O	Pwm2 channel 5 output pin
			KS_O5	O	Keyscan output channel 5 pin
			QDEC_X1	I	Qdec x input channel 1 pin
-	42	20	P33	I/O	General-purpose digital input and output pin
			SPI0_CLK	I/O	Spi0 clock pin
			UART0_RTS	O	Uart0 rts pin
			PWM0_CH3	O	Pwm0 channel 3 output pin
			KS_O17	O	Keyscan output channel 17 pin
-	43	21	P32	I/O	General-purpose digital input and output pin
			SPI0_CS	I/O	Spi0 cs pin
			PWM0_CH2	O	Pwm0 channel 2 output pin
			KS_O16	O	Keyscan output channel 16 pin
-	44	22	P34	I/O	General-purpose digital input and output pin
			PWM2_CH4	O	Pwm2 channel 4 output pin
			KS_O18	O	Keyscan output channel 18 pin
			TIMER1_EXT	I	Timer1 external input pin

			ANT_SW2	O	Antenna switch 2 pin
-	45	23	P35	I/O	General-purpose digital input and output pin
			PWM2_CH5	O	Pwm2 channel 5 output pin
			KS_O19	O	Keyscan output channel 19 pin
			UART0_CTS	I	Uart0 cts pin
			ANT_SW3	O	Antenna switch 3 pin
-	46	24	P36	I/O	General-purpose digital input and output pin
			PWM2_CH6	O	Pwm2 channel 6 output pin
			KS_O20	O	Keyscan output channel 20 pin
			CLKTRIM_EXT	I	CLKTRIM external clock measurement input pin
			ANT_SW5	O	Antenna switch 5 pin
-	47	25	P37	I/O	General-purpose digital input and output pin
			PWM2_CH7	O	Pwm2 channel 7 output pin
			KS_O21	O	Keyscan output channel 21 pin
			KS_I2	I	Keyscan input channel 2 pin
			ANT_SW6	O	Antenna switch 6 pin
-	-	26	P51	I/O	General-purpose digital input and output pin
			PWM1_CH7	O	Pwm1 channel 7 output pin
			KS_O1	O	Keyscan output channel 1 pin
			QDEC_X_IDX	I	Qdec x_idx input pin
-	-	27	P50	I/O	General-purpose digital input and output pin
			PWM2_CH1	O	Pwm2 channel 1 output pin
			KS_O0	O	Keyscan output channel 0 pin
-	48	28	P22	I/O	General-purpose digital input and output pin
			QDEC_X_IDX	I	Qdec x_idx input pin
			PWM1_CH4	O	Pwm1 channel 4 output pin
			I2S_S_CLK	I	I2s slave data input pin
			SPI1_MISO	I/O	Spi1 miso pin
			KS_I2	I	Keyscan input channel 2 pin
			I2S_M_CLK	O	I2s master clock output pin
-	1	29	P23	I/O	General-purpose digital input and output pin
			QDEC_Y_IDX	I	Qdec y_idx input pin
			PWM1_CH5	O	Pwm1 channel 5 output pin

			I2S_S_WS	I	I2s slave chip select input pin
			SPI1_MOSI	I/O	Spi1 mosi pin
			KS_I3	I	Keyscan input channel 3 pin
			I2S_M_WS	O	I2s master chip select output pin
14	-	30	P04	I/O	General-purpose digital input and output pin
			I2S_S_SDI	I	I2s slave data input pin
			I2S_M_SDI	I	I2s master data input pin
			TIMER2_OUT	O	Timer2 output pin
			PWM1_CH0	O	Pwm1 channel 0 output pin
			KS_O10	O	Keyscan output channel 10 pin
			QDEC_X0	I	Qdec x input channel 0 pin
15	-	31	P05	I/O	General-purpose digital input and output pin
			I2S_S_SDO	O	I2s slave data output pin
			I2S_M_SDO	O	I2s master data output pin
			ANT_SW2	O	Antenna switch 2 pin
			PWM1_CH1	O	Pwm1 channel 1 output pin
			KS_O11	O	Keyscan output channel 11 pin
			QDEC_X1	I	Qdec x input channel 1 pin
			ADC_TRG	I	Adc external trg input pin
16	2	32	DVDD	P	Hldo output pin, typical value 1.2V
-	3	33	VIPIO2	P	Special io power supply pin（P40~P44）
-	4	34	P44	I/O	General-purpose digital input and output pin
			UART0_TX	O	Uart0 tx pin
			I2S_S_WS	I	I2s slave chip select input pin
			PWM2_CH4	O	Pwm2 channel 4 output pin
			I2S_M_WS	O	I2s master chip select input pin
-	5	35	P43	I/O	General-purpose digital input and output pin
			SPI0_MOSI	I/O	Spi0 mosi pin
			UART0_RX	I	Uart0 rx pin
			PWM2_CH3	O	Pwm2 channel 3 output pin
			I2S_MCLK	O	I2s output sample clock
			KS_O15	O	Keyscan output channel 15 pin
-	6	36	P42	I/O	General-purpose digital input and output pin

			SPI0_MISO	I/O	Spi0 miso pin
			I2S_S_SDO	O	I2s slave clock output pin
			PWM2_CH2	O	Pwm2 channel 2 output pin
			I2S_M_SDO	O	I2s master data output pin
-	7	37	P41	I/O	General-purpose digital input and output pin
			SPI0_CS	I/O	Spi0 cs pin
			I2S_S_SDI	I	I2s slave data input pin
			PWM2_CH1	O	Pwm2 channel 1 output pin
			I2S_M_SDI	I	I2s master data input pin
			I2C0_SCL	I/O	I2c0 scl pin
-	8	38	P40	I/O	General-purpose digital input and output pin
			SPI0_CLK	I/O	Spi0 clock pin
			I2C0_SDA	I/O	I2c0 sda pin
			I2S_S_CLK	I	I2s slave clock input pin
			I2S_M_CLK	O	I2s master clock output pin
			PWM2_CH0	O	Pwm2 channel 0 output pin
			KS_O2	O	Keyscan output channel 2 pin
-	-	39	P25	I/O	General-purpose digital input and output pin
			UART1_RX	I	Uart1 rx pin
			SPI1_CS	I/O	Spi1 cs pin
			ANT_SW1	O	Antenna switch 1 pin
			QDEC_Z_IDX	I	Qdec z_idx input pin
			KS_O9	O	Keyscan output channel 9 pin
			PWM1_CH1	O	Pwm1 channel 1 output pin
17	-	40	P24	I/O	General-purpose digital input and output pin
			UART1_TX	O	Uart1 tx pin
			SPI1_CLK	I/O	Spi1 clock pin
			ANT_SW0	O	Antenna switch 0 pin
			PWM1_CH0	O	Pwm1 channel 0 output pin
			KS_O8	O	Keyscan output channel 8 pin
			QDEC_Z0	I	Qdec z input channel 0 pin
18	9	41	P31	I/O	General-purpose digital input and output pin
			UART0_RX	I	Uart0 rx pin

			PWM1_CH7	O	Pwm1 channel 7 output pin
			SPI0_MISO	I/O	Spi0 miso pin
			ADC_CH0	AI	Adc input channel 0 pin
			KS_I5	I	Keyscan input channel 5 pin
19	10	42	P30	I/O	General-purpose digital input and output pin
			UART0_TX	O	Uart0 tx pin
			PWM1_CH6	O	Pwm1 channel 6 output pin
			SPI0_MOSI	I/O	Spi0 mosi pin
			ADC_CH1	AI	Adc input channel 1 pin
			KS_I4	I	Keyscan input channel 4 pin
-	-	43	P27	I/O	General-purpose digital input and output pin
			UART1_RTS	O	Uart1 rts pin
			UART0_RX	I	Uart0 rx pin
			SPI1_MOSI	I/O	Spi1 mosi pin
			ADC_CH2	AI	Adc input channel 2 pin
			KS_O15	O	Keyscan output channel 15 pin
			QDEC_Z1	I	Qdec z input channel 1 pin
20	11	44	P03	I/O	General-purpose digital input and output pin
			UART0_RTS	O	Uart0 rts pin
			ANT_SW1	O	Antenna switch 1 pin
			SPI0_CLK	I/O	Spi0 clock pin
			USB_DP	AI/AO	Usb dp pin
			PWM0_CH3	O	Pwm0 channel 3 output pin
			KS_I1	I	Keyscan input channel 1 pin
21	12	45	P02	I/O	General-purpose digital input and output pin
			UART0_CTS	I	Uart0 cts pin
			ANT_SW0	O	Antenna switch 0 pin
			SPI0_CS	I/O	Spi0 cs pin
			USB_DM	AI/AO	Usb dm pin
			PWM0_CH2	O	Pwm0 channel 2 output pin
			KS_I0	I	Keyscan input channel 0 pin
22	-	46	VOUT2_BK	P	DCDC-2 voltage output pin, power supply to internal Flash
23	13	47	VOUT1_BK	P	DCDC-1 voltage output pin, can be directly

					connected to VCC_RF pin
24	14	48	VSW1	P	DCDC internal power switch (switching frequency is about 650KHz), an external inductor is required when using
25	15	49	VSS_BK	P	Common ground terminal of DCDC power supply, independent power ground
26	16	50	VBAT_BK	P	(The power input pin of the chip, only in the QFN package) The power input pin of the DCDC provides power for the internal DCDC
-	17	51	P52	I/O	General-purpose digital input and output pin
			ADC_TRG	I	Adc external trg input pin
			SPI1_CS	I/O	Spi1 chip select pin
			ADC_CH3	AI	Adc input channel 3 pin
			KS_O2	O	Keyscan output channel 2 pin
			ANT_SW7	O	Antenna switch 7 pin
-	18	52	P15	I/O	General-purpose digital input and output pin
			UART1_RX	I	Uart1 rx pin
			ANT_SW7	O	Antenna switch 7 pin
			I2C0_SCL	I/O	I2c0 scl pin
			PWM0_CH5	O	Pwm0 channel 5 output pin
			KS_I5	I	Keyscan input channel 5 pin
			QDEC_X1	I	Qdec x input channel 1 pin
-	19	53	P14	I/O	General-purpose digital input and output pin
			UART1_TX	O	Uart1 tx pin
			I2C0_SDA	I/O	I2c0 sda pin
			ANT_SW6	O	Antenna switch 6 pin
			PWM0_CH4	O	Pwm0 channel 4 output pin
			KS_I4	I	Keyscan input channel 4 pin
			QDEC_X0	I	Qdec x input channel 0 pin
27	20	54	P13	I/O	General-purpose digital input and output pin
			I2C0_SDA	I/O	I2c0 sda pin
			I2S_S_WS	I	I2s slave chip select input pin
			I2S_M_WS	O	I2s master chip select output pin
			PWM0_CH7	O	Pwm0 channel 7 output pin
			ANT_SW4	O	Antenna switch 4 pin
			XL1	AO	External 32.768KHz clock source output

28	21	55	P12	I/O	General-purpose digital input and output pin
			I2C0_SCL	I/O	I2c0 clk pin
			I2S_S_CLK	I	I2s slave clock input pin
			I2S_M_CLK	O	I2s master clock output pin
			PWM0_CH6	O	Pwm0 channel 6 output pin
			XL0	AI	External 32.768KHz clock source input
-	-	56	VBAT	P	The power input pin of the chip
29	22	57	XC0	AI	External 32MHz clock source input
30	23	58	XC1	AO	External 32MHz clock source output
31	24	59	VCC_RF	P	RF power supply port, can be directly connected to VOUT1_BK pin
-	-	60	NC		
-	-	61	GND	P	Ground
32	25	62	ANT	AI/AO	RF antenna pin, an external antenna is required for use
-	-	63	GND	P	Ground
1	26	64	NRESET	I	Reset pin
33	49	-	E-PAD	P	Chip bottom pad, common ground (QFN package only)

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 PAN108 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 PAN108 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.

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

Addressing Space	Token	Modules
System Control Space (0xE000_E000 – 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA+ 0×010	System Timer Control Registers
0xE000_E100 – 0xE000_ECFF	SCS_BA+ 0×100	Nested Vectored Interrupt Control Registers
0xE000_ED00 – 0xE000_ED8F	SCS_BA+ 0×D00	System Control Block Registers
AHB Modules Space (0x4002_0000 – 0x5003_FFFF)		
0x5002_0000-0x5003_FFFF	BLE5.3	BLE5.3(private 2.4G) Control Registers
0x400A_0000-0x400A_FFFF	USB	USB Control Registers
0x4009_0000-0x4009_FFFF	ECC	ECC Control Registers
0x4008_0000-0x4008_FFFF	eFuse(control)	eFuse Control Registers
0x4007_0000-0x4007_FFFF	ANACTL	ANACTL Control Registers
0x4006_0000-0x4006_FFFF	DMA	DMA Control Registers
0x4005_0000-0x4005_FFFF	FLASH(control, accelerator)	FLASH(control, accelerator) Control Registers
0x4004_0000-0x4004_FFFF	RCC	RCC Control Registers
0x4003_0000-0x4003_FFFF	SYSTEM(IOMUX)	SYSTEM(IOMUX) Control Registers
0x4002_0000-0x4002_FFFF	GPIO	GPIO Control Registers
APB2 Modules Space (0x4001_0000 – 0x4001_AFFF)		

0x4001_A000-0x4001_AFFF	QDEC	QDEC Control Registers
0x4001_9000-0x4001_9FFF	Keyscan	Keyscan Control Registers
0x4001_7000-0x4001_8FFF	Reserved	Reserved
0x4001_6000-0x4001_6FFF	CLKTRIM	CLKTRIM Control Registers
0x4001_5000-0x4001_5FFF	TIMER2	Timer2 Control Registers
0x4001_4000-0x4001_4FFF	TIMER1	Timer1 Control Registers
0x4001_3000-0x4001_3FFF	UART1	UART1 Control Registers
0x4001_2000-0x4001_2FFF	Reserved	Reserved
0x4001_1000-0x4001_1FFF	SPI1	SPI1 Control Registers
0x4001_0000-0x4001_0FFF	Reserved	Reserved
APB1 Modules Space (0x4000_0000 – 0x4000_CFFF)		
0x4000_C000-0x4000_CFFF	PWM2	PWM2 Control Registers
0x4000_B000-0x4000_BFFF	PWM1	PWM1 Control Registers
0x4000_A000-0x4000_AFFF	I2S(Slave)	I2S(Slave) Control Registers
0x4000_9000-0x4000_9FFF	I2S(Master)	I2S(Master) Control Registers
0x4000_8000-0x4000_8FFF	TIMER0	Timer0 Control Registers
0x4000_7000-0x4000_7FFF	WWDT	Window Watchdog Timer Control Registers
0x4000_6000-0x4000_6FFF	WDT	Watchdog Timer Control Registers
0x4000_5000-0x4000_5FFF	ADC	ADC Control Registers
0x4000_4000-0x4000_4FFF	PWM0	PWM0 Control Registers
0x4000_3000-0x4000_3FFF	UART0	UART0 Control Registers
0x4000_2000-0x4000_2FFF	Reserved	Reserved
0x4000_1000-0x4000_1FFF	SPI0	SPI0 Control Registers
0x4000_0000-0x4000_0FFF	I2C0	I2C0 Control Registers
Flash and SRAM Memory Space(0x0000_0000 – 0x2000_FFFF)		
0x2000_0000-0x2000_FFFF	SRAM	SRAM Memory Space
0x0000_0000-0x007F_FFFF	FLASH	Flash Memory Space

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_0000
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_P3_MFP	SYS_BA+0x0C	R/W	P3 Multiple Function and Input Type Control Register	0x0000_0000
SYS_P4_MFP	SYS_BA+0x10	R/W	P4 Multiple Function and Input Type Control Register	0x0000_00C0
SYS_P5_MFP	SYS_BA+0x14	R/W	P5 Multiple Function and Input Type Control Register	0x0000_0000
SYS_REGLCTL	SYS_BA+0x40	R/W	Register Write-Protection Control Register	0x0000_0000
SYS_CNTRL0	SYS_BA+0x48	RW	DMA Channel Select Control Register	0x3F3F_FFC0

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_0000

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]	MFP [7]	P0.7 Alternate Function Select Bit Bits EXT[7] (SYS_P0_MFP[23]), ALT[7] (SYS_P0_MFP[15]), and MFP[7] (SYS_P0_MFP[7]) determine the P0.7 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_clk function is selected. (0, 1, 0) = uart1_rx function is selected. (0, 1, 1) = tm0_ext function is selected. (1, 0, 0) = pwm1_ch3 function is selected. (1, 0, 1) = ks_o13 function is selected. (1, 1, 0) = qdec_y1 function is selected. (1, 1, 1) = ant_sw5 function is selected.
[6]	MFP [6]	P0.6 Alternate Function Select Bit Bits EXT[6] (SYS_P0_MFP[22]), ALT[6] (SYS_P0_MFP[14]), and MFP[6] (SYS_P0_MFP[6]) determine the P0.6 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi1_cs function is selected. (0, 1, 0) = uart1_tx function is selected. (0, 1, 1) = ant_sw3 function is selected. (1, 0, 0) = pwm1_ch2 function is selected. (1, 0, 1) = ks_o12 function is selected. (1, 1, 0) = qdec_y0 function is selected. (1, 1, 1) = i2s_mclk function is selected.
[5]	MFP [5]	P0.5 Alternate Function Select Bit Bits EXT[5] (SYS_P0_MFP[21]), ALT[5] (SYS_P0_MFP[13]), and MFP[5] (SYS_P0_MFP[5]) determine the P0.5 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = i2s_s_sd_o function is selected. (0, 1, 0) = i2s_m_sd_o function is selected. (0, 1, 1) = ant_sw2 function is selected. (1, 0, 0) = pwm1_ch1 function is selected. (1, 0, 1) = ks_o11 function is selected. (1, 1, 0) = qdec_x1 function is selected. (1, 1, 1) = adc_trg function is selected.

[4]	MFP [4]	P0.4 Alternate Function Select Bit Bits EXT[4] (SYS_P0_MFP[20]), ALT[4] (SYS_P0_MFP[12]), and MFP[4] (SYS_P0_MFP[4]) determine the P0.4 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = i2s_s_sd_i function is selected. (0, 1, 0) = i2s_m_sd_i function is selected. (0, 1, 1) = timer2_out function is selected. (1, 0, 0) = pwm1_ch0 function is selected. (1, 0, 1) = ks_o10 function is selected. (1, 1, 0) = qdec_x0 function is selected.
[3]	MFP [3]	P0.3 Alternate Function Select Bit Bits EXT[3] (SYS_P0_MFP[19]), ALT[3] (SYS_P0_MFP[11]), and MFP[3] (SYS_P0_MFP[3]) determine the P0.3 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_rts function is selected. (0, 1, 0) = ant_sw1 function is selected. (0, 1, 1) = spi0_clk function is selected. (1, 0, 0) = usb_dp function is selected. (1, 0, 1) = pwm0_ch3 function is selected. (1, 1, 0) = tadc_vld function is selected. (1, 1, 1) = ks_i1 function is selected.
[2]	MFP [2]	P0.2 Alternate Function Select Bit Bits EXT[2] (SYS_P0_MFP[18]), ALT[5] (SYS_P0_MFP[10]), and MFP[2] (SYS_P0_MFP[2]) determine the P0.2 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_cts function is selected. (0, 1, 0) = ant_sw0 function is selected. (0, 1, 1) = spi0_cs function is selected. (1, 0, 0) = usb_dm function is selected. (1, 0, 1) = pwm0_ch2 function is selected. (1, 1, 0) = tadc_datl function is selected. (1, 1, 1) = ks_i0 function is selected.
[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) = uart0_rx function is selected. (0, 1, 0) = i2c0_sda function is selected. (0, 1, 1) = i2s_m_sd_o function is selected. (1, 0, 0) = pwm0_ch1 function is selected. (1, 0, 1) = i2s_s_sd_o function is selected. (1, 1, 0) = ks_o7 function is selected.
[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) = uart0_tx function is selected. (0, 1, 0) = i2c0_scl function is selected.

		(0, 1, 1) = timer0_out function is selected. (1, 0, 0) = pwm0_ch0 function is selected. (1, 0, 1) = i2s_m_sd_i function is selected. (1, 1, 0) = ks_o6 function is selected. (1, 1, 1) = i2s_s_sd_i function is selected.

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]	MFP [7]	P1.7 Alternate Function Select Bit Bits EXT[7] (SYS_P1_MFP[23]), ALT[7] (SYS_P1_MFP[15]), and MFP[7] (SYS_P1_MFP[7]) determine the P1.7 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart1_rts function is selected. (0, 1, 1) = timer1_out function is selected. (1, 0, 0) = pwm0_ch7 function is selected. (1, 0, 1) = ks_i7 function is selected. (1, 1, 0) = qdec_y1 function is selected.
[6]	MFP [6]	P1.6 Alternate Function Select Bit Bits EXT[6] (SYS_P1_MFP[22]), ALT[6] (SYS_P1_MFP[14]), and MFP[6] (SYS_P1_MFP[6]) determine the P1.6 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart1_cts function is selected. (0, 1, 0) = ant_sw4 function is selected. (0, 1, 1) = timer0_out function is selected. (1, 0, 0) = pwm0_ch6 function is selected. (1, 0, 1) = ks_i6 function is selected. (1, 1, 0) = qdec_y0 function is selected.
[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) = uart1_rx function is selected. (0, 1, 0) = ant_sw7 function is selected. (0, 1, 1) = i2c0_scl function is selected. (1, 0, 0) = pwm0_ch5 function is selected. (1, 0, 1) = ks_i5 function is selected. (1, 1, 0) = qdec_x1 function is selected.
[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) = uart1_tx function is selected. (0, 1, 0) = i2c0_sda function is selected.

		(0, 1, 1) = ant_sw6 function is selected. (1, 0, 0) = pwm0_ch4 function is selected. (1, 0, 1) = ks_i4 function is selected. (1, 1, 0) = qdec_x0 function is selected.
[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) = i2c0_sda function is selected. (0, 1, 0) = i2s_s_ws function is selected. (0, 1, 1) = i2s_m_ws function is selected. (1, 0, 0) = pwm0_ch7 function is selected. (1, 0, 1) = tadc_dath function is selected. (1, 1, 0) = RCH function is selected. (1, 1, 1) = ant_sw4 function is selected. (xtl_quick_en=1, xtl_32k_n 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) = i2c0_scl function is selected. (0, 1, 0) = i2s_s_clk function is selected. (0, 1, 1) = i2s_m_clk function is selected. (1, 0, 0) = pwm0_ch6 function is selected. (1, 0, 1) = tadc_clk function is selected. (1, 1, 0) = dpll 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) = spi0_miso function is selected. (0, 1, 0) = clktrim_ext function is selected. (0, 1, 1) = adc_ch4 function is selected. (1, 0, 0) = pwm0_ch5 function is selected. (1, 0, 1) = ks_o5 function is selected. (1, 1, 0) = ahb_clk function is selected. (1, 1, 1) = i2s_mclk function is selected.
[0]	MFP [0]	P1.0 Alternate Function Select Bit Bits EXT[0] (SYS_P1_MFP[16]), ALT[0] (SYS_P1_MFP[8]), and MFP[0] (SYS_P1_MFP[0]) determine the P1.0 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_mosi function is selected. (0, 1, 0) = tm0_ext function is selected. (0, 1, 1) = adc_ch5 function is selected. (1, 0, 0) = pwm0_ch4 function is selected. (1, 0, 1) = ks_o4 function is selected. (1, 1, 0) = 32k_clk function is selected.

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]	MFP [7]	P2.7 Alternate Function Select Bit Bits EXT[7] (SYS_P2_MFP[23]), ALT[7] (SYS_P2_MFP[15]), and MFP[7] (SYS_P2_MFP[7]) determine the P2.7 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart1_rts function is selected. (0, 1, 0) = uart0_rx function is selected. (0, 1, 1) = spi1_mosi function is selected. (1, 0, 0) = adc_ch2 function is selected. (1, 0, 1) = ks_o15 function is selected. (1, 1, 0) = qdec_z1 function is selected.
[6]	MFP [6]	P2.6 Alternate Function Select Bit Bits EXT[6] (SYS_P2_MFP[22]), ALT[6] (SYS_P2_MFP[14]), and MFP[6] (SYS_P2_MFP[6]) determine the P2.6 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart1_cts function is selected. (0, 1, 0) = uart0_tx function is selected. (0, 1, 1) = spi1_miso function is selected. (1, 0, 1) = ks_o14 function is selected. (1, 1, 0) = pwm1_ch2 function is selected.
[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) = uart1_rx function is selected. (0, 1, 0) = spi1_cs function is selected. (0, 1, 1) = ant_sw1 function is selected. (1, 0, 0) = qdec_z_idx function is selected. (1, 0, 1) = ks_o9 function is selected. (1, 1, 0) = pwm1_ch1 function is selected.
[4]	MFP [4]	P2.4 Alternate Function Select Bit Bits EXT[4] (SYS_P2_MFP[20]), ALT[4] (SYS_P2_MFP[12]), and MFP[4] (SYS_P2_MFP[4]) determine the P2.4 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart1_tx function is selected. (0, 1, 0) = spi1_clk function is selected.

		(0, 1, 1) = ant_sw0 function is selected. (1, 0, 0) = pwm1_ch0 function is selected. (1, 0, 1) = ks_o8 function is selected. (1, 1, 0) = qdec_z0 function is selected.
[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) = qdec_y_idx function is selected. (0, 1, 0) = pwm1_ch5 function is selected. (0, 1, 1) = i2s_s_ws function is selected. (1, 0, 0) = spi1_mosi function is selected. (1, 0, 1) = ks_i3 function is selected. (1, 1, 0) = i2s_m_ws function is selected.
[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) = qdec_x_idx function is selected. (0, 1, 0) = pwm1_ch4 function is selected. (0, 1, 1) = i2s_s_clk function is selected. (1, 0, 0) = spi1_miso function is selected. (1, 0, 1) = ks_i2 function is selected. (1, 1, 0) = i2s_m_clk function is selected.
[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_clk function is selected. (0, 1, 0) = tm2_ext function is selected. (1, 0, 0) = adc_ch7 function is selected. (1, 0, 1) = ks_i1 function is selected. (1, 1, 0) = qdec_z1 function is selected.
[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_cs function is selected. (0, 1, 0) = tm1_ext function is selected. (0, 1, 1) = int2 function is selected. (1, 0, 0) = adc_ch6 function is selected. (1, 0, 1) = ks_i0 function is selected. (1, 1, 0) = qdec_z0 function is selected.

3.1.4.4 Multiple Function Port3 Control Register (SYS_P3_MFP)

Register	Offset	R/W	Description	Reset Value
SYS_P3_MFP	SYS_BA+0x0C	R/W	P3 Multiple Function and Input Type Control Register	0x0000_0000

Bits	Description	
[31:24]	Reserved	Reserved.
[23:16]	EXT[7:0]	P3[7:0] Alternate Function Selection Extension The pin function of P3 depends on EXT, MFP and ALT.
[15:8]	ALT[7:0]	P3[7:0] Alternate Function Select Bit The pin function of P3 depends on EXT, MFP and ALT.
[7]	MFP [7]	P3.7 Alternate Function Select Bit Bits EXT[7] (SYS_P3_MFP[23]), ALT[7] (SYS_P3_MFP[15]), and MFP[7] (SYS_P3_MFP[7]) determine the P3.7 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch7 function is selected. (0, 1, 0) = ks_o21 function is selected. (0, 1, 1) = ks_i2 function is selected. (1, 0, 0) = mdm_access_match function is selected. (1, 1, 0) = USB_DOP function is selected. (1, 1, 1) = ant_sw6 function is selected.
[6]	MFP [6]	P3.6 Alternate Function Select Bit Bits EXT[6] (SYS_P3_MFP[22]), ALT[6] (SYS_P3_MFP[14]), and MFP[6] (SYS_P3_MFP[6]) determine the P3.6 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch6 function is selected. (0, 1, 0) = ks_o20 function is selected. (1, 0, 0) = mdm_rx_on function is selected. (1, 0, 1) = clktrim_ext function is selected. (1, 1, 0) = USB_DIDIF function is selected. (1, 1, 1) = ant_sw5 function is selected.
[5]	MFP [5]	P3.5 Alternate Function Select Bit Bits EXT[5] (SYS_P3_MFP[21]), ALT[5] (SYS_P3_MFP[13]), and MFP[5] (SYS_P3_MFP[5]) determine the P3.5 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch5 function is selected. (0, 1, 0) = ks_o19 function is selected. (1, 0, 0) = mdm_tx_on function is selected. (1, 0, 1) = uart0_cts function is selected. (1, 1, 0) = USB_DIM function is selected. (1, 1, 1) = ant_sw3 function is selected.
[4]	MFP [4]	P3.4 Alternate Function Select Bit Bits EXT[4] (SYS_P3_MFP[20]), ALT[4] (SYS_P3_MFP[12]), and MFP[4] (SYS_P3_MFP[4]) determine the P3.4 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch4 function is selected.

		(0, 1, 0) = ks_o18 function is selected. (1, 0, 0) = mdm_tx_data function is selected. (1, 0, 1) = tm1_ext function is selected. (1, 1, 0) = USB_DIP function is selected. (1, 1, 1) = ant_sw2 function is selected.
[3]	MFP [3]	P3.3 Alternate Function Select Bit Bits EXT[3] (SYS_P3_MFP[19]), ALT[3] (SYS_P3_MFP[11]), and MFP[3] (SYS_P3_MFP[3]) determine the P3.3 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_clk function is selected. (0, 1, 0) = lvr_out function is selected. (0, 1, 1) = uart0_rts function is selected. (1, 0, 0) = pwm0_ch3 function is selected. (1, 0, 1) = ks_o17 function is selected. (1, 1, 1) = mdm_spi_miso function is selected.
[2]	MFP [2]	P3.2 Alternate Function Select Bit Bits EXT[2] (SYS_P3_MFP[18]), ALT[5] (SYS_P3_MFP[10]), and MFP[2] (SYS_P3_MFP[2]) determine the P3.2 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_cs function is selected. (0, 1, 0) = bod_out function is selected. (0, 1, 1) = int0 function is selected. (1, 0, 0) = pwm0_ch2 function is selected. (1, 0, 1) = ks_o16 function is selected. (1, 1, 1) = mdm_spi_mosi function is selected.
[1]	MFP [1]	P3.1 Alternate Function Select Bit Bits EXT[1] (SYS_P3_MFP[17]), ALT[1] (SYS_P3_MFP[9]), and MFP[1] (SYS_P3_MFP[1]) determine the P3.1 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_rx function is selected. (0, 1, 0) = pwm1_ch7 function is selected. (0, 1, 1) = spi0_miso function is selected. (1, 0, 0) = adc_ch0 function is selected. (1, 0, 1) = ks_i5 function is selected. (1, 1, 1) = mdm_spi_csn function is selected.
[0]	MFP [0]	P3.0 Alternate Function Select Bit Bits EXT[0] (SYS_P3_MFP[16]), ALT[0] (SYS_P3_MFP[8]), and MFP[0] (SYS_P3_MFP[0]) determine the P3.0 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = uart0_tx function is selected. (0, 1, 0) = pwm1_ch6 function is selected. (0, 1, 1) = spi0_mosi function is selected. (1, 0, 0) = adc_ch1 function is selected. (1, 0, 1) = ks_i4 function is selected. (1, 1, 1) = mdm_spi_clk function is selected.

3.1.4.5 Multiple Function Port4 Control Register (SYS_P4_MFP)

Register	Offset	R/W	Description	Reset Value
SYS_P4_MFP	SYS_BA+0x10	R/W	P4 Multiple Function and Input Type Control Register	0x0000_00C0

Bits	Description	
[31:24]	Reserved	Reserved.
[23:16]	EXT[7:0]	P4[7:0] Alternate Function Selection Extension The pin function of P4 depends on EXT, MFP and ALT.
[15:8]	ALT[7:0]	P4[7:0] Alternate Function Select Bit The pin function of P4 depends on EXT, MFP and ALT.
[7]	MFP [7]	P4.7 Alternate Function Select Bit Bits EXT[7] (SYS_P4_MFP[23]), ALT[7] (SYS_P4_MFP[15]), and MFP[7] (SYS_P4_MFP[7]) determine the P4.7 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, 0, 1) = ant_sw6 function is selected. (1, 1, 0) = ks_o1 function is selected.
[6]	MFP [6]	P4.6 Alternate Function Select Bit Bits EXT[6] (SYS_P4_MFP[22]), ALT[6] (SYS_P4_MFP[14]), and MFP[6] (SYS_P4_MFP[6]) determine the P4.6 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, 0, 1) = ant_sw5 function is selected. (1, 1, 0) = ks_o0 function is selected.
[5]	MFP [5]	P4.5 Alternate Function Select Bit Bits EXT[5] (SYS_P4_MFP[21]), ALT[5] (SYS_P4_MFP[13]), and MFP[5] (SYS_P4_MFP[5]) determine the P4.5 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch0 function is selected. (0, 1, 0) = ks_o22 function is selected. (0, 1, 1) = mdm_offset_est_done function is selected. (1, 0, 0) = USB_NDOE function is selected. (1, 0, 1) = fsynsd_div[0] function is selected. (1, 1, 0) = Reserved. (1, 1, 1) = Reserved.
[4]	MFP [4]	P4.4 Alternate Function Select Bit Bits EXT[4] (SYS_P4_MFP[20]), ALT[4] (SYS_P4_MFP[12]), and MFP[4] (SYS_P4_MFP[4]) determine the P4.4 function. (0, 0, 0) = GPIO function is selected.

		(0, 0, 1) = uart0_tx function is selected. (0, 1, 0) = i2s_s_ws function is selected. (0, 1, 1) = pwm2_ch4 function is selected. (1, 0, 0) = i2s_m_ws function is selected. (1, 0, 1) = mdm_io_rstn function is selected. (1, 1, 1) = USB_DOM function is selected.
[3]	MFP [3]	P4.3 Alternate Function Select Bit Bits EXT[3] (SYS_P4_MFP[19]), ALT[3] (SYS_P4_MFP[11]), and MFP[3] (SYS_P4_MFP[3]) determine the P4.3 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_mosi function is selected. (0, 1, 0) = uart0_rx function is selected. (0, 1, 1) = pwm2_ch3 function is selected. (1, 0, 0) = i2s_mclk function is selected. (1, 0, 1) = mdm_rx_data_soft[1] function is selected.（uart0_tx_en=1, uart0_tx is avtive） (1, 1, 1) = ks_o15 function is selected.
[2]	MFP [2]	P4.2 Alternate Function Select Bit Bits EXT[2] (SYS_P4_MFP[18]), ALT[5] (SYS_P4_MFP[10]), and MFP[2] (SYS_P4_MFP[2]) determine the P4.2 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_miso function is selected. (0, 1, 0) = i2s_s_sd_o function is selected. (0, 1, 1) = pwm2_ch2 function is selected. (1, 0, 0) = i2s_m_sd_o function is selected. (1, 0, 1) = mdm_rx_data_soft[0] function is selected.
[1]	MFP [1]	P4.1 Alternate Function Select Bit Bits EXT[1] (SYS_P4_MFP[17]), ALT[1] (SYS_P4_MFP[9]), and MFP[1] (SYS_P4_MFP[1]) determine the P4.1 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_cs function is selected. (0, 1, 0) = i2s_s_sd_i function is selected. (0, 1, 1) = pwm2_ch1 function is selected. (1, 0, 0) = i2s_m_sd_i function is selected. (1, 0, 1) = mdm_rx_clk function is selected. (1, 1, 1) = i2c0_scl function is selected.
[0]	MFP [0]	P4.0 Alternate Function Select Bit Bits EXT[0] (SYS_P4_MFP[16]), ALT[0] (SYS_P4_MFP[8]), and MFP[0] (SYS_P4_MFP[0]) determine the P4.0 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = spi0_clk function is selected. (0, 1, 0) = i2c0_sda function is selected. (0, 1, 1) = i2s_s_clk function is selected. (1, 0, 0) = i2s_m_clk function is selected. (1, 0, 1) = mdm_rx_data function is selected. (1, 1, 0) = pwm2_ch0 function is selected. (1, 1, 1) = ks_o2 function is selected.

3.1.4.6 Multiple Function Port5 Control Register (SYS_P5_MFP)

Register	Offset	R/W	Description	Reset Value
SYS_P5_MFP	SYS_BA+0x14	R/W	P5 Multiple Function and Input Type Control Register	0x0000_0000

Bits	Description	
[31:24]	Reserved	Reserved.
[23:16]	EXT[7:0]	P5[7:0] Alternate Function Selection Extension The pin function of P5 depends on EXT, MFP and ALT.
[15:8]	ALT[7:0]	P5[7:0] Alternate Function Select Bit The pin function of P5 depends on EXT, MFP and ALT.
[7]	MFP [7]	P5.7 Alternate Function Select Bit Bits EXT[7] (SYS_P5_MFP[23]), ALT[7] (SYS_P5_MFP[15]), and MFP[7] (SYS_P5_MFP[7]) determine the P5.7 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch5 function is selected. (0, 1, 0) = ks_o5 function is selected. (0, 1, 1) = qdec_x1 function is selected. (1, 0, 0) = fsynsd_clk. (1, 0, 1) = Reserved. (1, 1, 0) = Reserved. (1, 1, 1) = Reserved.
[6]	MFP [6]	P5.6 Alternate Function Select Bit Bits EXT[6] (SYS_P5_MFP[22]), ALT[6] (SYS_P5_MFP[14]), and MFP[6] (SYS_P5_MFP[6]) determine the P5.6 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm1_ch6 function is selected. (0, 1, 0) = uart0_cts function is selected. (0, 1, 1) = uart1_tx function is selected. (1, 0, 0) = qdec_x0 function is selected. (1, 0, 1) = ks_o11 function is selected.
[5]	MFP [5]	P5.5 Alternate Function Select Bit Bits EXT[5] (SYS_P5_MFP[21]), ALT[5] (SYS_P5_MFP[13]), and MFP[5] (SYS_P5_MFP[5]) determine the P5.5 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch3 function is selected. (0, 1, 0) = ks_o4 function is selected. (0, 1, 1) = qdec_x0 function is selected. (1, 0, 0) = fsynsd_div[5] function is selected. (1, 0, 1) = Reserved. (1, 1, 0) = Reserved. (1, 1, 1) = Reserved.
[4]	MFP [4]	P5.4 Alternate Function Select Bit Bits EXT[4] (SYS_P5_MFP[20]), ALT[4] (SYS_P5_MFP[12]), and MFP[4] (SYS_P5_MFP[4]) determine the P5.4 function. (0, 0, 0) = GPIO function is selected.

		(0, 0, 1) = pwm2_ch2 function is selected. (0, 1, 0) = ks_o23 function is selected. (0, 1, 1) = qdec_z_idx function is selected. (1, 0, 0) = fsynsd_div[4] function is selected. (1, 0, 1) = Reserved. (1, 1, 0) = Reserved. (1, 1, 1) = Reserved.
[3]	MFP [3]	P5.3 Alternate Function Select Bit Bits EXT[3] (SYS_P5_MFP[19]), ALT[3] (SYS_P5_MFP[11]), and MFP[3] (SYS_P5_MFP[3]) determine the P5.3 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm1_ch3 function is selected. (0, 1, 0) = ks_o3 function is selected. (0, 1, 1) = qdec_y_idx function is selected. (1, 0, 0) = fsynsd_div[3] function is selected. (1, 0, 1) = Reserved. (1, 1, 0) = Reserved. (1, 1, 1) = Reserved.
[2]	MFP [2]	P5.2 Alternate Function Select Bit Bits EXT[2] (SYS_P5_MFP[18]), ALT[5] (SYS_P5_MFP[10]), and MFP[2] (SYS_P5_MFP[2]) determine the P5.2 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = adc_trg function is selected. (0, 1, 0) = spi1_cs function is selected. (0, 1, 1) = int1 function is selected. (1, 0, 0) = adc_ch3 function is selected. (1, 0, 1) = ks_o2 function is selected. (1, 1, 0) = ant_sw7 function is selected.
[1]	MFP [1]	P5.1 Alternate Function Select Bit Bits EXT[1] (SYS_P5_MFP[17]), ALT[1] (SYS_P5_MFP[9]), and MFP[1] (SYS_P5_MFP[1]) determine the P5.1 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm1_ch7 function is selected. (0, 1, 0) = ks_o1 function is selected. (0, 1, 1) = qdec_x_idx function is selected. (1, 0, 0) = fsynsd_div[2] function is selected. (1, 0, 1) = Reserved. (1, 1, 0) = Reserved. (1, 1, 1) = xth_32M function is selected.
[0]	MFP [0]	P5.0 Alternate Function Select Bit Bits EXT[0] (SYS_P5_MFP[16]), ALT[0] (SYS_P5_MFP[8]), and MFP[0] (SYS_P5_MFP[0]) determine the P5.0 function. (0, 0, 0) = GPIO function is selected. (0, 0, 1) = pwm2_ch1 function is selected. (0, 1, 0) = ks_o0 function is selected. (0, 1, 1) = mdm_stdby_en function is selected. (1, 0, 0) = fsynsd_div[1] function is selected. (1, 0, 1) = Reserved.

		(1, 1, 0) = Reserved. (1, 1, 1) = rco_32M function is selected.

3.1.4.7 Register Write-Protection Control Register (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
BODCTL	RCC_BA+0xC	
IPRST0	RCC_BA+0x4	
WDT_CTL	WDT_BA+0x00	
WDT_ALTCTL	WDT_BA+0x04	
ECC_TRG	ECC_BA+0x0C	
ECC_IRQ	ECC_BA+0x10	
ACC_TRG	ECC_BA+0x14	
SOURCE_A	ECC_BA+0x20	
SOURCE_B	ECC_BA+0x24	
ECC_LOGIC_SELECT	ECC_BA+0x18	
MULT_DIV_SELECT	ECC_BA+0xA8	
EFUSE_CTL	EF_BA+0x00	
EFUSE_TRG	EF_BA+0x1C	

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

3.1.4.8 DMA Channel Select Control Register (SYS_CNTRL0)

Register	Offset	R/W	Description	Reset Value
SYS_CNTRL0	SYS_BA+0x48	R/W	DMA Channel Select Control Register	0x3F3F_FFC0

Bits	Description	
[31:24]	pa_rampdw	The number of 32M XTH cycle of each step for pa ramp down time
[23:16]	pa_rampup	The number of 32M XTH cycle of each step for pa ramp up time
[15:6]	pa_rampout_bres_en	Each bit is the enable control of related pa_rampout_bres[x]. For each bit 0: the related pa_rampout_bres[x] is always 0 1: the related pa_rampout_bres[x] will be ramp up or down by pa ramp state machine.
[5]	fsynsd_div_sel	Rf fsynsd_div from pad for test 0: fsynsd_div from phy module 1: fsynsd_div from pad
[4]	Usb_test_sel	Usb data from/to pad for test 0: usb data from/to itself io 1: usb data from/to pad for test
[3]	mdm_ll_sel	Mdm data from/to pad for test 0: mdm data from/to ll 1: mdm data from/to pad
[2]	Reserved	Reserved
[1]	Rxadc_smpl_sel	Rx adc data sample clock edge select 0: sample rx adc data with clock negedge 1: sample rx adc data with clock posedge
[0]	Mdm_spi_sel	Mdm spi select 0: mdm spi source from ll 1: mdm spi source from pad

3.1.5 System Timer (SysTick)

The PAN108 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 PAN108 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	Modem	MODEM	Modem interrupt
16	30	standby	Low Power	standby interrupt
17	29	Sleep	Low Power	sleep interrupt
18	28	BOD	BOD	BOD interrupt
19	27	DMA	DMA	DMA interrupt
20	26	EINT2	GPIO	External signal interrupt from P2.0 pin
21	25	EINT1	GPIO	External signal interrupt from P5.2 pin
22	24	EINT0	GPIO	External signal interrupt from P3.2 pin
23	23	Reserved	-	-
24	22	USB	USB	USB interrupt
25	21	USB_DMA	USB_DMA	USB_DMA interrupt
26	20	PWM2	PWM2	PWM2 interrupt
27	19	PWM1	PWM1	PWM1 interrupt
28	18	GPIO_P0/P1/P2/P3/P4/P5	GPIO	External signal interrupt from GPIO group P2~P4 except P3.2
29	17	QDEC	QDEC	QDEC interrupt

30	16	KeyScan	KeyScan	KeyScan interrupt
31	15	CLKTRIM	CLKTRIM	CLKTRIM interrupt
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	ECC_ACC	ECC_ACC	ECC_ACC interrupt
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	I2S_M/I2S_S	I2S_M/I2S_S	I2S_M/I2S_S interrupt
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 PAN108 series MCU status and operating mode control are managed System Control Block Registers. Including CPUID, PAN108 series MCU interrupt priority and PAN108 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 PAN108 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	1.8-3.7	System 3V power supply input
VBAT_BK	I	1.8-3.7	BUCK power supply input
VIPIO2	I	1.8-3.7	IO power supply input(only P40-P44)
VOUT1_BK	O	1.5(typ)	BUCK power supply output1(VCC_xxx)
VOUT2_BK	O	1.8-3.7	Flash power output, connect with a capacitor
VCC_RF	I	1.4-3.7	RF VCC input / Digital (HLDO) VCC input / Analog 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.

PAN108 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.

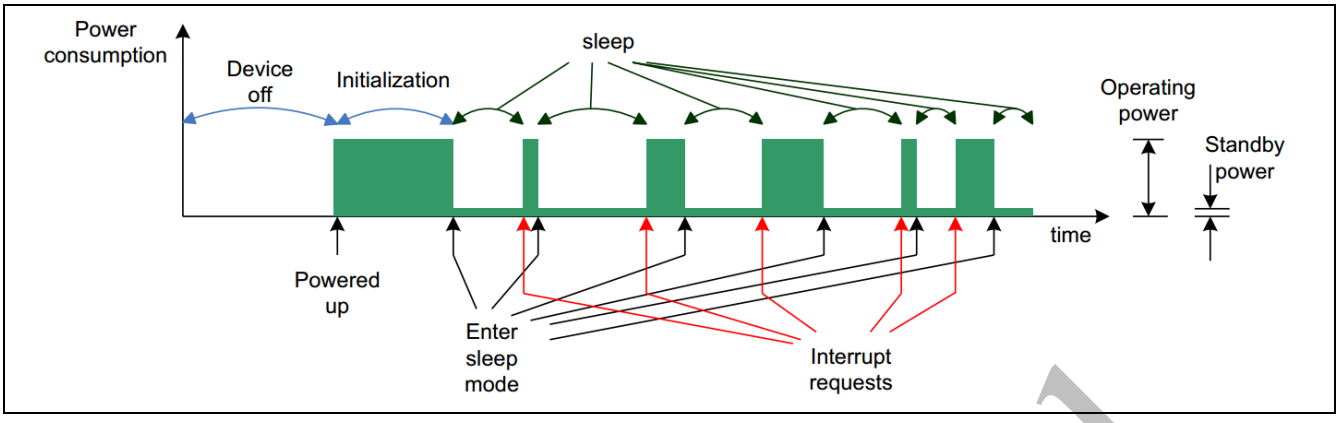


Figure 3-1 Power Consumption of Interrupt Driven System

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, PAN108 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.

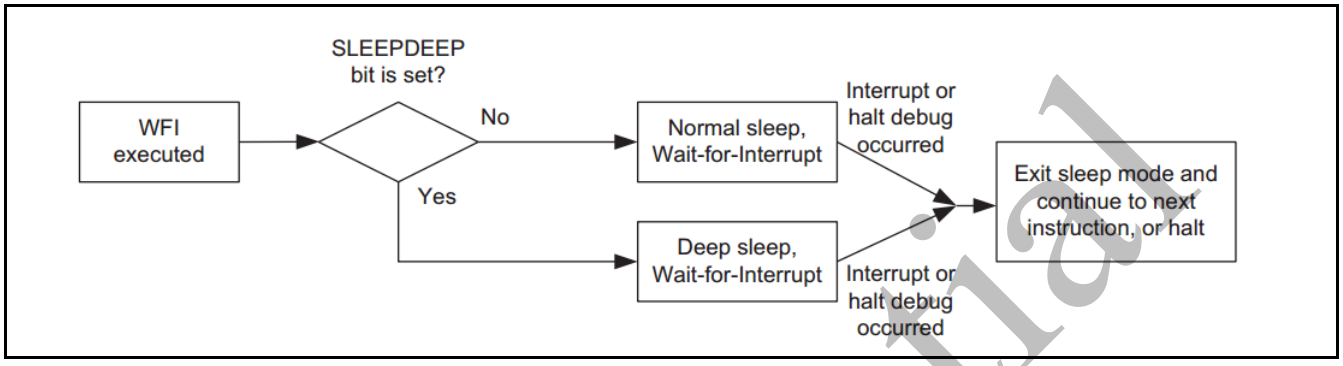


Figure 3-2 WFI Schedule

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.

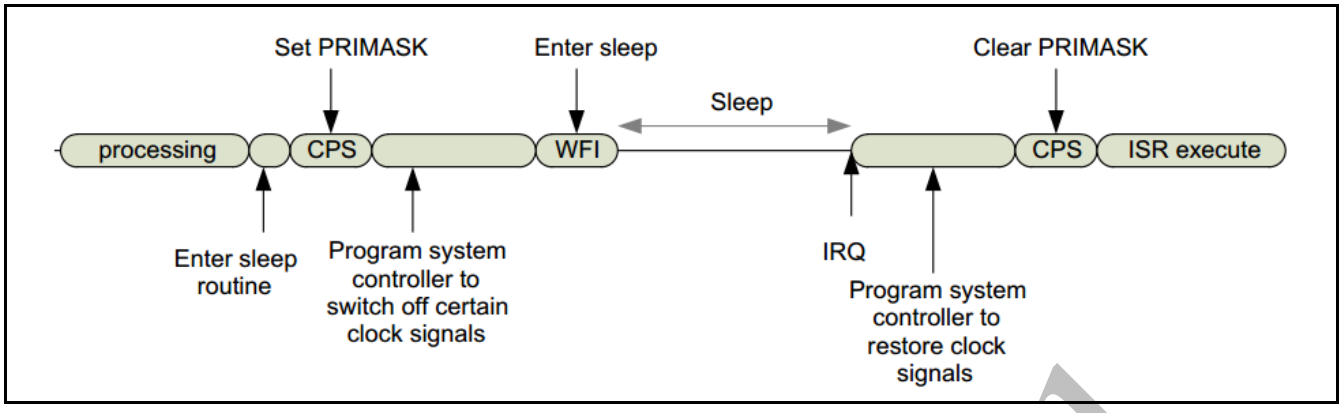


Figure 3-3 Execute RRIMASK during Sleep Mode

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.

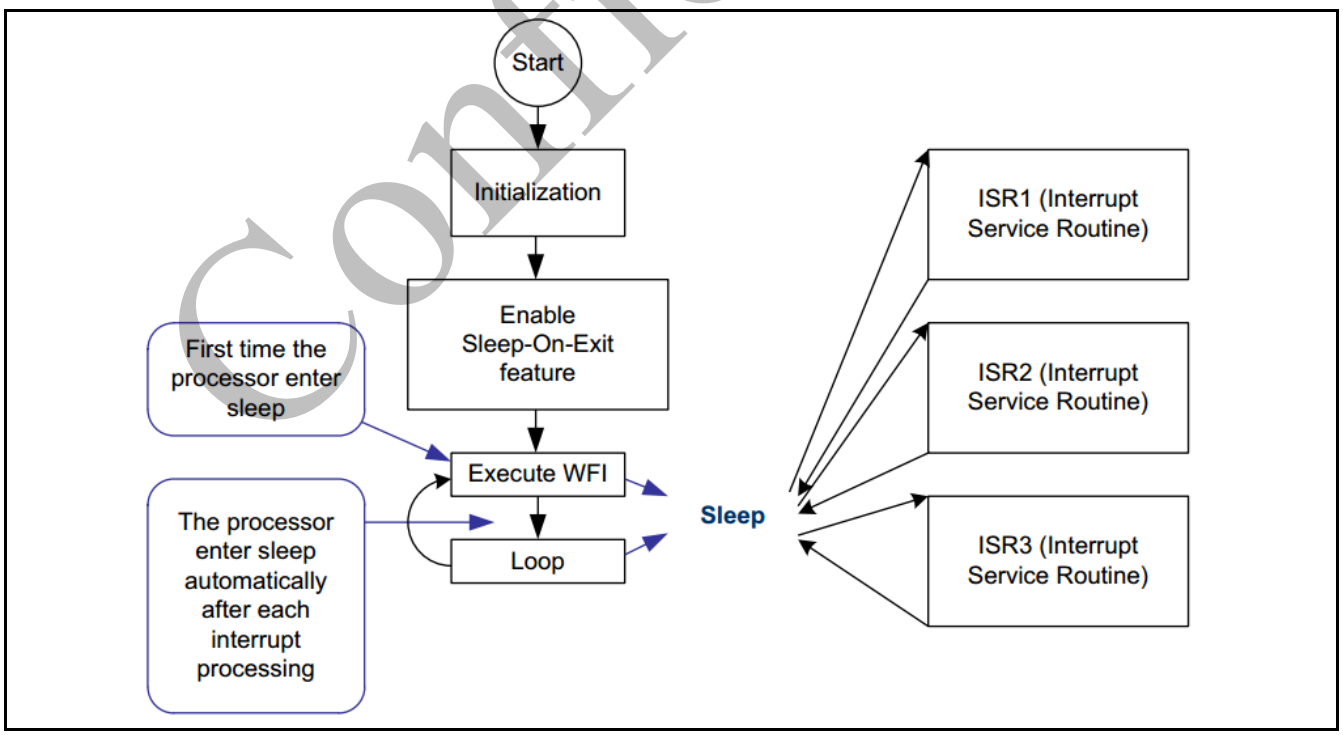


Figure 3-4 Sleep-On-Exit Process

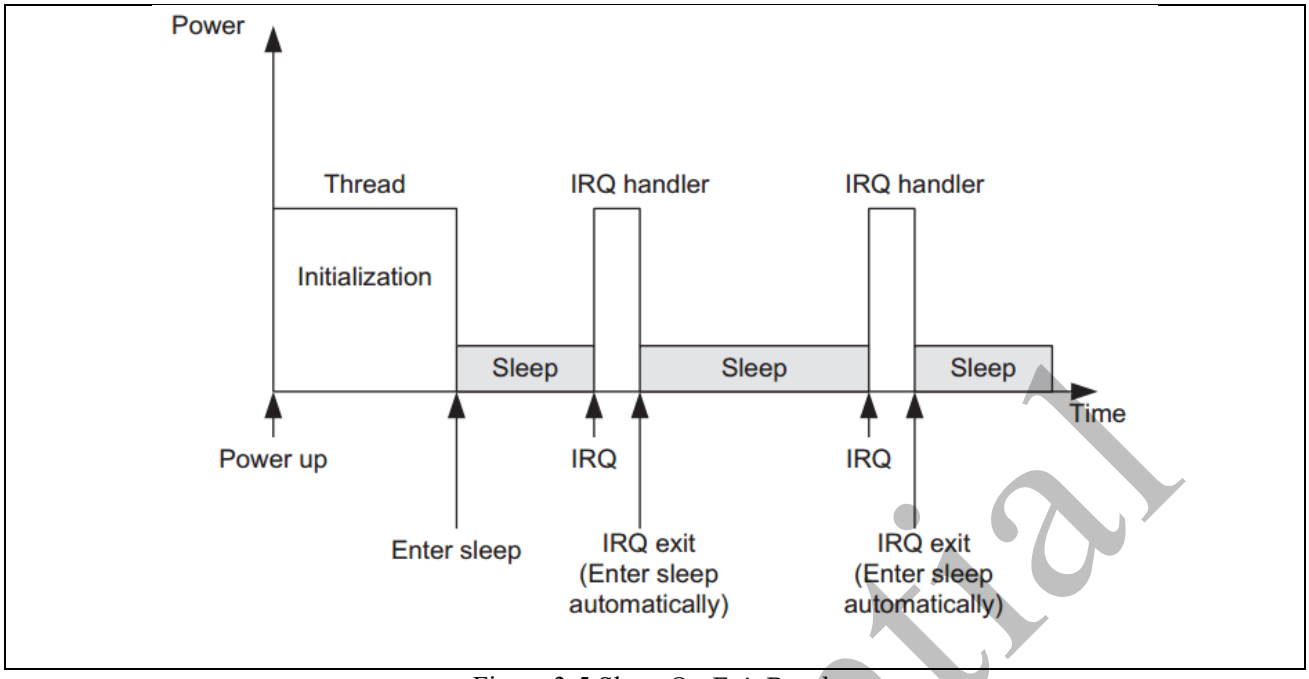


Figure 3-5 Sleep-On-Exit Result

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

3.2.4.1 Power Modes and Wake-up Sources

There are three levels of low power mode. In a different application, use can users different low power mode to get the lowest power and best performance. Table 3-8 lists the related descriptions of each power mode.

Table 3-8 Power Mode Difference Table

Chip work mode	Descriptions	Wake up	1.2V region clock	3V region clock	1.2V power supply
STANDBY_M0	standby_en = 1 SLEEPDEEP = 1 sleep_cnt_en = 0 WFI	GPIO (P5.6)	All closed	All closed	Power off

STANDBY_M1	standby_en = 1 SLEEPDEEP = 1 ldo_power_ctrl = 0 gpio_power_ctrl = 1 sram0/1/2/3_power_ctrl = 0/1 ll_ram_power_ctrl=0/1 decrypt_ram_power_ctrl=0/1 WFI	GPIO (all) LP_TMR	GPIO, SRAM, LL_RAM, Decrypt_RAM (you can choose to maintain power), and other 1.2V modules are powered off.	RCL or XTL	LP_LDO supplies power to GPIO, SRAM, decrypted data RAM and BLE (ll_data_ram, ll_seq_ram, medom sleep memory) according to the selection of *** _power_ctrl.
DEEPSLEEP	standby_en = 0 SLEEPDEEP = 1 WFI	GPIO (all) LP_TMR TIMER0/1/2 WDT QDEC KEYSCAN	All closed except CLK_32K	RCL or XTL	LP_LDO
SLEEP	standby_en = 0 SLEEPDEEP = 0 WFI	All peripherals interrupt	CPU_CLK closed; RCH/XTH/DPLL is selected to be turned on according to the software configuration	RCL or XTL	HP_LDO

Standby Mode

There are two modes in standby mode.

In Mode0 mode, the 1.2V area is powered off (both large and small LDOs are off), and the 3V area is powered; the CLK_32K (RCL or XTL) clock can be selected whether to turn on or not; in Mode1 mode, LP_LDO is turned on which supplies GPIO and SRAM.

Entry mode: register STBEN is turned on, WFI+SLEEPDEEP, exit from ISR (Sleep-on-Exit).

Wake-up conditions: Mode0, only external GPIO in the 3V area can wake up;

Mode1, all GPIOs and Timer in 3V area can wake up.

Deepsleep Mode

In this mode, the 1.2V area is powered by the small LDO, the large LDO is turned off, and the 3V area is powered; the CLK_32K (RCL or XTL) clock can be selected to be turned on or not. All other high-speed clocks are turned off.

Entry mode: register STBEN is closed, WFI+SLEEPDEEP, or exit from ISR (Sleep-on-Exit).

Wake-up conditions: all GPIOs, 3V area LP_TMR, TIMER0/1/2, WDT, QDEC, Keyscan;

Sleep Mode

In this mode, the 1.2V area is powered by the large LDO, the small LDO is off, and the 3V area is powered. The system clock remains in the state before entering sleep. The CPU clock is stopped, and other peripheral modules operate normally. In order to reduce power consumption, you can turn off the peripheral clock before entering sleep. When the CPU clock is stopped, the clocks of the following modules will also be turned off:

- EFUSE module
- rom module
- ICACHE module
- FLASH encryption module
- CPU
- SPI_FLCTL control module
- SYS module
- AHB Bridge of BLE
- BLE's register read and write clock
- AHB address decoder
- ROM_FLASH_DEC module

Entry mode: register STBEN is closed, WFI, or exit from ISR (Sleep-on-Exit).

Wake-up conditions: any interrupt, 3V area LP_TMR.

3.2.4.2 Operation Mode Switch Process

After the system powered on or GPIOs reset, it enters the normal operation mode. The three low power modes can be switched with the normal operation mode according to the different control mode. The following shows the 3V partial wake-up source(P5.6 and LP_TMR) configuration.

P56 wake-up steps:

1. Enable P56 as input mode
2. Enable external wake-up; (LP_FL_CTRL[0])
3. Select the wake-up level; (LP_FL_CTRL[1], 1 = high-level wake-up, 0 = low-level wake-up)
4. Turn off the low-power timer. (LP_FL_CTRL[3]), enter low-power mode.

LP_TMR wake up steps:

1. Enable low-power timer. (LP_FL_CTRL[3])
2. Set the timer threshold. (LP_TMR_CTRL)

3. Turn off RCL in other areas except for the low-power timer, LP_FL_CTRL[6] set to 0.

Enter and Exit Standby Mode

Entering Standby mode:

1. Enable Standby interrupt(interrupt number is 30)
2. Set Standby_en to 1(LP_FL_CTRL[4])
3. Set Dig_nrst_en to 1(LP_RST_CTRL[10])
4. Set Dig_nrst_dly_time(LP_RST_CTRL[9:0]), the delay time is determined according to the FLASH manufacturer's data， the counter reference source is 32K clock
5. Entering Mode1 requires additional configuration of LP_FL_CTRL[31:24], Mode0 skips this step;
6. Set CLK_TOP_CTLR[9:8] to 0,switch the system clock to RCH, turn off XTH, and turn off DPLL; (Ensure that the clock selection is consistent with the power-on default)
7. Wake-up source selection: P56 or LP_TMR, Mode1 can choose GPIO interrupt to wake up;
8. BUCK keeps power, the FLASH controller needs to exit the enhanced mode, and the FLASH enters the DEEPSLEEP mode;

BUCK does not protect the power, and the FLASH controller does not need to exit the enhanced mode, you need to set X_FL_DPCTL[0]=0, and the FLASH does not enter the DEEPSLEEP mode;If FLASH enters the DEEPSLEEP mode,it is necessary to set X_FL_DPCTL[31:8] to ensure normal wakeup;

1. Set the interrupt mask and open the mask as soon as it wakes up, If the wake-up source is GPIO, the wake-up level of IO needs to be switched after wake-up (for example, the wake-up source is GPIO high-level wake-up, in order to re-enter the low-power mode as soon as possible, the wake-up source needs to be switched to GPIO low-level after wake-up);
2. Set the CPU SLEEPDEEP register to 1; Address: 0xE000ED10
3. __WFI()

Exiting Standby mode:

1. P56 or LP_TMR wake up, (all GPIO in Mode1 mode can wake up), generate Standby interrupt;
2. Clear the Standby interrupt in the interrupt function, LP_INT_CTRL[8];
3. Check the interrupt source LP_INT_CTRL[16] (P56), LP_INT_CTRL[17] (Tmr), GPIO interrupt.

Enter and Exit DeepSleep Mode

Entering DeepSleep mode:

1. Enable Sleep interrupt, interrupt number is 29. (According to different wake-up sources, sleep interrupt can also be selected not to be turned on)
2. Set Standby_en to 0 (LP_FL_CTRL[4])
3. Wake-up source can be selected as P56 or LP_TMR; other GPIO; Timer0/1/2; WDT/WWDT; QDEC; KEYSCAN;
4. Set Dig_nrst_dly_time(LP_RST_CTRL[9:0]), the delay time is determined according to the FLASH manufacturer's data，the counter reference source is 32K clock
5. Set power switch LP_FL_CTRL[31:24];
6. Set CLK_TOP_CTLR[9:8] to 0,switch the system clock to RCH, turn off XTH, and turn off DPLL; (Ensure that the clock selection is consistent with the power-on default);
7. BUCK keeps power, the FLASH controller needs to exit the enhanced mode, and the FLASH enters the DEEPSLEEP mode,it is necessary to set X_FL_DPCTL[31:8] to ensure normal wakeup;
8. Set the interrupt mask and open the mask as soon as it wakes up, If the wake-up source is GPIO, the wake-up level of IO needs to be switched after wake-up (for example, the wake-up source is GPIO high-level wake-up, in order to re-enter the low-power mode as soon as possible, the wake-up source needs to be switched to GPIO low-level after wake-up);
9. Set the CPU SLEEPDEEP register to 1; Address: 0xE000ED10
10. _WFI();

Exiting DeepSleep mode:

1. P56 or LP_TMR wake up, (all GPIO in Mode1 mode can wake up), generate Standby interrupt;
2. Clear the Standby interrupt in the interrupt function, LP_INT_CTRL[8];
3. Check the interrupt source LP_INT_CTRL[16] (P56), LP_INT_CTRL[17] (Tmr), GPIO interrupt.

Enter and Exit Sleep Mode

Entering Sleep mode:

1. Enable the interrupt corresponding to the wake-up source.
2. Set Standby_en to 0 (LP_FL_CTRL[4])
3. Set the CPU SLEEPDEEP register(Address: 0xE000ED10) to 0.

4. __WFI().

Exiting Sleep mode:

1. The wake-up source generates an interrupt.
2. Handle the interrupt and clear the interrupt source.

Enter and Exit Standby Mode0 Timing

hl_gpio_isolate: It is always set as 0.

hl_sram_isolate: All SRAMs use the same control signal. It is asserted after phy_soc_lvlshift_en_global is pulled down half second of the slow clock cycle, and then deasserted after dig_nrst deaseerts 1.5 cycles.

Entering Standby mode0:

1. After the program executes the WFI instruction, pull up the deepsleep signal on the system clock (RCH rising edge).
2. After the Deepsleep signal is synchronized by the 3V CLK32K clock, sleep_status is pulled up and enters a low power consumption state at the third rising edge which is marked at time 3.
3. At time 4, turn off the system clock. If at Standby mode, turn off EN_RCH on the falling edge of RCH.
4. At time 4, turn off the Level Shift enable signals EN_LS and EN_LS_GPIO;
5. At time 6, turn off the power supplies including Flash, PMU and large/small LDOs.
6. At time 7, reset dig_nrst. And at the next falling edge of time 6, turn off EN_RCL/XTL;
7. Sleeping signal will be pulled down due to reset during standby mode.

Exiting normal mode0:

1. After P56 is pulled high, PMU, Flash power supply, large and small LDO, EN_RCH, EN_RCL/EN_XTL are pulled high at the same time.
2. After CLK32K (RCL or XTL) is stable, sleep_status is pulled down at the third rising edge, that is, at time 9. Level Shift is enabled at time 10.
3. T3 time is determined by dig_nrst_time.
4. T2 time of flash rdp timing, because of reset it is value is default value.

Enter and Exit DeepSleep Mode Timing

Entering deepsleep:

1. After the program executes the WFI instruction, pull up the deepsleep signal on the system clock (RCH rising edge).

2. After the Deepsleep signal is synchronized by the 3V CLK32K clock, sleep_status is pulled up and enters a low power consumption state at time 0.
3. At time 0, turn off the system clock. If at Standby mode, turn off EN_RCH on the falling edge of RCH.
4. At time 1 turn off the Level Shift enable signals EN_LS;
5. At time 2, turn off the power supplies including Flash, PMU and large/small LDOs.

Exiting deepsleep:

1. After sleeptime match set value, sleep_status is pulled down at time 0, that is. Level Shift and clks is enabled at time 2.
2. T3 time is determined by dig_nrst_time.
3. T2 time of flash rdp timing, it is set by software.

Enter and Exit Standby Mode1 Timing

Phy_soc_lvlshift_en_global stays high only when HLDO does not enter sleep mode, otherwise it will be pulled down when entering low power mode, and will be pulled up when exiting low power mode. The timing is the same as the standby m0 timing diagram.

Phy_soc_lvlshift_en_gpio is kept high only when m1 with gpio retentioin, otherwise it will be pulled down when entering low power mode, and will be pulled up when exiting low power mode. The timing is the same as the standby m0 timing diagram.

Isolation_en is the control signal of the isolation of the 3V module.

Isolation_peripherial is the isolation control signal of each peripheral. It is set to 1 only when the corresponding module maintains power during the low power consumption period, and it is set to 0 in other states.

Phy_soc_pmu_sw_dvddxx_en is the power control signal of the module that needs to be protected. It is set to 1 when the module needs to be protected, and it is set to 0 when it is not.

The Standby mode1 timing is difference from Standby mode0, illustrated as follow:

1. The RCL (CLK32K) clock is not turned off.
2. Regarding the power-related control, there are the following differences, the closing and opening sequence is the same as EN_PMU. The difference of power control are shown in Table 3-9.

Table 3-9 The Difference Of Power Control

	Standby Mode1	DeepSleep Mode1
EN_LS	0	1
EN_LS_GPIO	1	1
EN_LDO_LP	1	1
LDO_POWER_EN	0	1
GPIO_POWER_EN	1	1
DECRYPT_EN	1	1
LL_EN	1	1
SRAM0_EN	0/1	1
SRAM1_EN	0/1	1
SRAM2_EN	0/1	1
SRAM3_EN	0/1	1

Exiting normal mode1:

1. When P56 is waken up, it will drive the power clock wakeup signal and the enable signal is pulled up at the same time as P56.
2. When P56 is waken up, sleep_status is pulled down at the third clock edge after P56 is pulled high, which is marked as time 2.
3. Other wake-up sources, including LP_TMR, TMR, WDT, QDEC, KEYSCAN, will generate a wakeup_combine, wake up the system and pull down sleep_status.
4. T2 time of flash rdp timing, because of reset it is value is default value.

3.2.4.3 LP_WDT Instruction

A WDT function is added to the 3V area to prevent the program from running away after the system wakes up. The configuration shows as follows:

1. Before entering low power mode, set LP_WDT_CNT and LP_WDT_EN to 1.
2. If LP_WDT_EN is enabled, the WDT timer starts counting. After counting to the value of LP_WDT_CNT, the entire chip will be reset (except the testctrl module in the 3V area and the count and flag of LP_WDT).
3. 32K clock is enabled during low power consumption,WDT timer starts counting after LP_WDT_EN is enabled.
4. 32K clock is disabled during low power consumption,WDT timer starts counting after 32K clock enabled.
5. In normal operation, LP_WDT_CLR is set to 1 after low-power wake-up, and then LP_WDT_CLR and LP_WDT_EN are cleared to 0.
6. During low power consumption, the RCL must be selected.

3.2.4.4 LDO Instruction

Note when using ADCLDO, RFFELDO, VCOLDO:

When opening:

1. Write 1 for LDOEN and 0 for LDOENLY.
2. After the software waits for 10us, LDOENLY writes 1.

When closed:

1. Write 0 for LDOEN and 1 for LDOENLY.

Note when using HPLDO, ANALDO, FSYNLDO:

When opening:

Write 1 to LDOEN and LDOENLY (the LDOENLY hardware will automatically be written to 0).

When closed:

Write 0 to LDOEN and write 1 to LDOENLY.

3.2.4.5 Buck Instruction

Buck voltage output corresponding to different configurations shows in Table 3-10.

Table 3-10 Buck Voltage Output

bucken	bypass	vout1_en	vout2_en	vout1	vout2
0	X	X	X	0	Floating
1	0	0	0	Floating	Floating
1	0	0	1	Floating	3
1	0	1	0	1.5	Floating
1	0	1	1	1.5	3
1	1	0	0	3（weak）	Floating
1	1	0	1	3（weak）	3
1	1	1	0	1.5	Floating
1	1	1	1	3	3

The change sequence of Buck (bypass) state from working to low power consumption and then waking up.

Table 3-11 States Change Illustration

State	Bucken	Bypass	{vout1_en，vout2_en}
Working	1	1	{1,1}
Low power mode (deepsleep/standby_m0/m1)	1001(Recommended, it can actually enter any state)		
Wake_up	1	1	{1,1}
Working	1	1	{1,1}

Buck (normal operation) state changes from working to low power mode and then waking up to work (whether it can directly enter low power mode, it needs to be tested to confirm. If not, the software needs to switch to bypass mode first, and then enter low power mode)

Table 3-12 States Change Illustration

State	Bucken	Bypass	{vout1_en，vout2_en}
Working	1	0	{1,1}
Low power mode (deepsleep/standby_m0/m1)	1001(Recommended, it can actually enter any state)		
Working	1	0	{1,1}

Note: In normal working mode (1011), during the wake-up process, the LP_BUCK[0] bit in the corresponding register needs to be cleared in the interrupt function (before the 3V synchronization is turned on), otherwise the default value 1 will be synchronized to 3V Area, which results the status changing to 1111.

3.2.5 Register Map

Register	Offset	R/W	Description	Reset Value
ANA Base Address: ANA_BA = 0x4007_0000				
LP_REG_SYNC	ANA_BA+0x00	R/W	Low Power Synchronize Register	0x0000_0000
LP_FL_CTRL	ANA_BA+0x04	R/W	Low Power Control Register	0xFFFF_F44F
LP_TMR_CTRL	ANA_BA+0x08	R/W	Low Power Counter Configuration Register	0x0000_0000
LP_INT_CTRL	ANA_BA+0x0C	R/W	Low Power Interrupt Control Register	0x0000_0003
LP_RST_CTRL	ANA_BA+0x10	R/W	Low Voltage Reset Control Register	0x0000_0003
LP_WDT_CNT	ANA_BA+0x14	R/W	LP_WDT Counter Configuration Register	0xFFFF_FFFF
LP_WDT_CTRL	ANA_BA+0x18	R/W	LP_WDT Control Register	0x0000_0000
LP_PATA_POLY	ANA_BA+0x1C	R/W	PTAT and POLY Configuration Register	0x0001_01A1
LP_HPLDO	ANA_BA+0x20	R/W	HPLDO Configuration Register	0x0000_0043
LP_LPLDO	ANA_BA+0x24	R/W	LPLDO Configuration Register	0x0000_0029
LP_ANALDO	ANA_BA+0x28	R/W	ANALDO Configuration Register	0x0000_0043
LP_FSYNLDO	ANA_BA+0x2C	R/W	FSYNLDO Configuration Register	0x0000_0043
LP_SW	ANA_BA+0x30	R/W	Power Switch Enable Register	0x0000_06FF
LP_BUCK	ANA_BA+0x34	R/W	BUCK Configuration Register	0x8420_228F
LP_MISC	ANA_BA+0x38	R/W	Other High Voltage Analog Control Register	0x0000_0006
ANA_ADCLDO	ANA_BA+0x40	R/W	ADCLDO Configuration Register	0x0000_0042
ANA_RFFELDO	ANA_BA+0x44	R/W	RFFELDO Configuration Register	0x0000_0042
ANA_VCOLDO	ANA_BA+0x48	R/W	VCOLDO Configuration Register	0x0000_0042
ANA_DFT	ANA_BA+0x4C	R/W	Analog DFT Configuration Register	0x0000_0000
ANA_TEMP	ANA_BA+0x50	R/W	Temperature Detect Register	0x0000_0000
ANA_MISC	ANA_BA+0x54	R/W	Other low Voltage Analog Control Register	0x0000_3AC0
ANA_MISC2	ANA_BA+0x58	R/W	Other low Voltage Analog Control Register2	0x0000_0004
ANA_RESERVED	ANA_BA+0x5C	R/W	Reserved Analog Register	0xFF00_FF00

3.2.6 Register Description

3.2.6.1 LP_REG_SYNC

Register	Offset	R/W	Description	Reset Value
LP_REG_SYNC	ANA_BA+0x00	R/W	Low Power Synchronize Register	0x0000_0000

Bits	Description	
[31:2]	Reserved	Reserved.
[2]	Reg_sync_ctrl	High Voltage Auto Synchronization Enable Bit. 1 = Auto_3V synchronization 0 = Manual set reg_sync_3v to 1
[1]	reg_sync_sts	Register Synchronization State (Read only) When bit[0] (reg_sync_3v) is set to 1, subsequent operation will be executed until this bit has been cleared to 0
[0]	reg_sync_3v	1 = The register's values are synchronized to high voltage (3V) region. 0 = The register's values are synchronized to low voltage (1.5V) region.

3.2.6.2 LP_FL_CTRL

Register	Offset	R/W	Description	Reset Value
LP_FL_CTRL	ANA_BA+0x04	R/W	Low Power Control Register	0xFFFF_F44F

Bits	Description	
[31]	Gpio_power_ctrl	GPIO Power Control Bit 1 = GPIO supplied 0 = GPIO not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[30]	ldo_power_ctrl	HLDO and LLDO Power Control Bit 1 = HLDO and LLDO supplied 0 = HLDO and LLDO not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[29]	Decrypt_sram_power_ctrl	Decrypt_SRAM Power Control Bit 1 = Decrypt_SRAM supplied 0 = Decrypt_SRAM not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[28]	Ll_sram_power_ctrl	LL_SRAM Power Control Bit 1 = LL_SRAM supplied 0 = LL_SRAM not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[27]	Sram3_power_ctrl	SRAM3 Power Control Bit 1 = SRAM3 supplied 0 = SRAM3 not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[26]	Sram2_power_ctrl	SRAM2 Power Control Bit 1 = SRAM2 supplied 0 = SRAM2 not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[25]	Sram1_power_ctrl	SRAM1 Power Control Bit 1 = SRAM1 supplied 0 = SRAM1 not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[24]	Sram0_power_ctrl	SRAM0 Power Control Bit 1 = SRAM0 supplied 0 = SRAM0 not supplied **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[23]	En_pmu_ctrl	POLY, PTAT, ANALDO and FSYNLDO Power Control Bit 1 = POLY, PTAT, ANALDO and FSYNLDO supplied

		0 = POLY, PTAT, ANALDO and FSYNLDO not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[22]	En_ldo_hp_ctrl	HPLDO Power Control Bit 1 = HPLDO supplied 0 = HPLDO not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[21]	En_ldo_lp_ctl	LPLDO Power Control Bit 1 = LPLDO supplied 0 = LPLDO not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[20]	en_rc32k_ctrl	RCL Power Control Bit 1 = RCL supplied 0 = RCL not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[19]	En_xtal32k_ctrl	XTL Power Control Bit 1 = XTL supplied 0 = XTL not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[18]	En_xtal_ctrl	XTH Power Control Bit 1 = XTH supplied 0 = XTH not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[17]	En_osch_ctrl	RCH Power Control Bit 1 = RCH supplied 0 = RCH not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[16]	En_dpll_ctrl	DPLL Power Control Bit 1 = DPLL supplied 0 = DPLL not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[15]	Buck_en_ctrl	BUCK Power Control Bit 1 = BUCK supplied

		0 = BUCK not supplied Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: this bit is only valid when working at Standby or DEEPSLEEP mode.
[14]	Buck_bp_ctrl	When STB and DEEPSLEEP, control the closing and opening of buck_bp; (need 3V sync, support auto_3V sync)
[13]	Buck_vout1_en_ctrl	When STB and DEEPSLEEP, control the closing and opening of Buck_vout1_en; (need 3V sync, support auto_3V sync)
[12]	Buck_vout2_en_ctrl	When STB and DEEPSLEEP, control the closing and opening of Buck_vout2_en; (need 3V sync, support auto_3V sync)
[11:10]	Reserved	Reserved
[9]	Sleeping_req	Software Control Sleep Mode Request Bit **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[8]	Sleepdeep_req	Software Control DeepSleep Mode Request Bit **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7]	Pwr_ctrl_en	Low Power Manual Control Enable Bit 1 = Enabled. Before entering low power mode, manual configuration is required LP_FL_CTRL[31:15] need to be configured by manual. 0 = Disabled. LP_FL_CTRL[23:21] and LP_FL_CTRL[18:15] are auto-controlled by hardware. The others are controlled by software. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6]	Rcl_gate_en	RCL Gate Enable Bit 1 = CLK_32K is turned on. 0 = CLK_32K is turned off. Note1: this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) Note2: When working in low power mode, gate all the clock excep the timer.
[5]	Reserved	Reserved
[4]	Standby_en	Standby Mode Enable Bit 1 = Enter standby mode when Deepsleep is set to 1 0 = Enter Deepsleep mode when Deepsleep is set to 1 **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[3]	Sleep_cnt_en	Low Power Counter Enable Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2]	Sleep_en	Low Power Sleep Enable Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	Extwkup_sel	P56 Wake Up Level Selection

		1 = High level wake up 0 = Low level wake up **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	Extwkup_en	P56 Wake Up Enable Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.3 LP_TMR_CTRL

Register	Offset	R/W	Description	Reset Value
LP_TMR_CTRL	ANA_BA+0x08	R/W	Low Power Counter Configuration Register	0x0000_0000

Bits	Description	
[31:0]	sleep_time	Low Power Counter Configuration. The time step is 31.2us. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.4 LP_INT_CTRL

Register	Offset	R/W	Description	Reset Value
LP_INT_CTRL	ANA_BA+0x0C	R/W	Low Power Interrupt Control Register	0x0000_0003

Bits	Description	
[31:18]	Reserved	Reserved.
[18]	Sram_ret_flg	Standby mode1 Set to 1 after the system wakes up, write 1 to clear 0. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[17]	Tmr32k_wk_flg	32k timer Wake Up Interrput Flag This bit can write 1 to clear. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[16]	P56_wk_flg	P56 Wake Up Interrput Flag This bit can write 1 to clear. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[15:10]	Reserved	Reserved
[9]	Sleep_int_flg	DeepSleep/Sleep Wake Up Interrput Flag This bit can write 1 to clear. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[8]	Standby_int_flg	Standby Wake Up Interrput Flag This bit can write 1 to clear. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7:4]	Reserved	Reserved
[1]	Sleep_int_en	DeepSleep/Sleep Wake Up Interrupt Enable Bit 1 = Enabled 0 = Disabled
[0]	Standby_int_en	Standby Wake Up Interrupt Enable Bit 1 = Enabled 0 = Disabled

3.2.6.5 LP_RST_CTRL

Register	Offset	R/W	Description	Reset Value
LP_RST_CTRL	ANA_BA+0x10	R/W	Low Voltage Reset Control Register	0x0000_0003

Bits	Description	
[31:2]	Reserved	Reserved.
[11]	clk_mux_nrst_ctrl	Delay reset the MUX of the AHB clock; 0 = no delay 1 = delay **Note1:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) **Note2:** after DeepSleep wakes up, it is adviced to reserve Flash's power-on time. The delay time is determined by Dig_nrst_dly_time.
[10]	Dig_nrst_en	Low Power Reset Enable Bit 1 = Reset 0 = Not reset **Note1:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) **Note2:** this bit is 3V signal, which needs to be synchronized.
[9:0]	Dig_nrst_dly_time	Reset Delay Time Configuration The time step is 31.2us. **Note1:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1) **Note2:** this bit is 3V signal, which needs to be synchronized.

3.2.6.6 LP_WDT_CNT

Register	Offset	R/W	Description	Reset Value
LP_WDT_CNT	ANA_BA+0x14	R/W	LP_WDT Counter Configuration Register	0xFFFF_FFFF

Bits	Description	
[31:0]	lp_wdt_time	WDT Timer Out Value **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.7 LP_WDT_CTRL

Register	Offset	R/W	Description	Reset Value
LP_WDT_CTRL	ANA_BA+0x18	R/W	LP_WDT Control Register	0x0000_0000

Bits	Description	
[31:3]	Reserved	Reserved.
[2]	lp_wdt_flg	lp_wdt Reset Flag This bit can write 1 to clear. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	lp_wdt_clr	lp_wdt Clear Bit **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	lp_wdt_en	lp_wdt Enable Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.8 LP_PTAT_POLY

Register	Offset	R/W	Description	Reset Value
LP_PATA_POLY	ANA_BA+0x1C	R/W	PTAT and POLY Configuration Register	0x0001_01A1

Bits	Description	
[31:18]	Reserved	Reserved.
[17:16]	ipoly_trim	Poly Current Trimming Bit
[15:9]	Reserved	Reserved
[8]	polyen	POLY Enablde Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7]	ptatcorefltract	Poly top Reference Voltage Filter Mode Switch Enable Bit 1 = Small filter resistor mode 0 = Large filter resistor mode **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6]	ptatcorefltrshrt	Poly top Reference Voltage Filter Disable Bit 1 = Filter circuit closed 0 = Filter circuit opende **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[5:3]	ptatvtrim	BG Reference Voltage Adjust Control Signal. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2:1]	ptatcorebyp	PTAT casocde Enable Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	ptatcoreen	PMU-PTAT Enable Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.9 LP_HPLDO

Register	Offset	R/W	Description	Reset Value
LP_HPLDO	ANA_BA+0x20	R/W	HPLDO Configuration Register	0x0000_0043

Bits	Description	
[31:8]	Reserved	Reserved.
[9]	hpldo_sel	Off-chip Capacitor Selection 1 = With off-chip Capacitor 0 = Without Off-chip Capacitor **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[8]	hpldo_prbiptatref	IPTAT Reference Voltage Control Bit 1 = Reference voltage abnormal 0 = Reference voltage normal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7]	hpldo_iprbptatcore	OTA Bias Current Enable Bit 1 = With Bias Current 0 = Without Bias Current **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6:3]	hpldo_vtrim	Output Voltage Adjust Control Signal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2]	hpldo_bypass	Output Voltage Bypass Control Signal 1 = The output voltage is the power voltage. 0 = LDO works normally and the output voltage is normal. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	hpldo_endly	HPLDO Internal Reference Voltage Filter Short Circuit Control Bit 1 = Reference voltage short circuit 0 = Reference voltage access **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	hpldo_en	HPLDO Enable Bit 1 = HPLDO works normally 0 = HPLDO closes **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.10 LP_LPLDO

Register	Offset	R/W	Description	Reset Value
LP_LPLDO	ANA_BA+0x24	R/W	LPLDO Configuration Register	0x0000_0029

Bits	Description	
[31:6]	Reserved	Reserved.
[5:4]	lpbiasvr	LDO Reference Voltage Adjust Control Signal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[3:1]	lpldo_dvddsel	LPLDO Output Voltage Adjust Signal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	lpldo_en	LPLDO Enable Bit 1 = Enablde 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.11 LP_ANALDO

Register	Offset	R/W	Description	Reset Value
LP_ANALDO	ANA_BA+0x28	R/W	ANALDO Configuration Register	0x0000_0043

Bits	Description	
[31:9]	Reserved	Reserved.
[8]	analdoprbiptatref	IPTAT Reference Voltage Control Signal 1 = Reference voltage abnormal 0 = Reference voltage normal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7]	analdoiprbptatcore	ANALDO's OTA Bias Enable Bit 1 = With bias current 0 = Without bias current **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6:3]	analdovtrim	ANA LDO Output Voltage Adjust Signal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2]	analdobypass	ANALDO's Bypass Signal 1 = Bypass power tunnel. The output voltage is power voltage. 0 = Power tunnel works normally. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	analdoendly	ANA LDO Reference Voltage Connection/Disconnection Control Bit 1 = Disconnection 0 = Connection **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	analdoen	ANA LDO Enable Control Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.12 LP_FSYNLDO

Register	Offset	R/W	Description	Reset Value
LP_FSYNLDO	ANA_BA+0x2C	R/W	FSYNLDO Configuration Register	0x0000_0043

Bits	Description	
[31:9]	Reserved	Reserved.
[8]	fsynldoprbiptatref	IPTAT Reference Voltage Control Signal 1 = Reference voltage abnormal 0 = Reference voltage normal **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7]	fsynldoiprbptatcore	FSYNLDO's OTA Bias Enable Signal 1 = With bias current 0 = Without bias curren **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6:3]	fsynldovtrim	FSYN LDO Output Adjust Option 4'b0000 0.986V 4'b1000 1.2V 4'b1111 1.38V **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2]	fsynldobypass	FSYNLDO Bypass Signal 1 = Bypass power tunnel. The output voltage is power voltage. 0 = Power tunnel works normally. **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	fsynldoendly	FSYN LDO Reference Voltage Connection/Disconnection Control Bit 1 = Disconnection 0 = Connection **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	fsynldoen	FSYN LDO Enable Control Bit 1 = Enabled 0 = Disabled **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.13 LP_SW

Register	Offset	R/W	Description	Reset Value
LP_SW	ANA_BA+0x30	R/W	Power Switch Enable Register	0x0000_06FF

Bits	Description	
[31:11]	Reserved	Reserved.
[10]	dvddecrypt_en	dvddcrypt Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[9:8]	Reserved	Reserved.
[7]	dvddgpio_en	dvddgpio Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6]	dvddll_en	dvddll Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[5]	dvddsram3_en	dvddsram3 Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[4]	dvddsram2_en	dvddsram2 Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[3]	dvddsram1_en	dvddsram1 Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2]	dvddsram0_en	dvddsram0 Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	hpdvdd_en	dvddhpdvdd Open/Close Enable Bit 1 = Switch open 0 = Switch closed **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

[0]	hpdvdd_pd	dvddhpdvdd Pull Down Enable Bit 1 = Pull down 0 = Not pull down(floating) **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.14 LP_BUCK

Register	Offset	R/W	Description	Reset Value
LP_BUCK	ANA_BA+0x34	R/W	BUCK Configuration Register	0x8420_228F

Bits	Description	
[31:27]	buck_zercal	Zero-crossing detection manual calibration opt **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[26:22]	buck_refcal	Low voltage detection manual calibration opt **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[21:17]	buck_imxcal	Manual Calibration of Overcurrent Detection opt **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[16:14]	buck_calen	Self-calibration Enable Bit **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[13:11]	buck_imax	Current limit regulation opt **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[10:8]	buck_vout2	Vout2 Output Adjust **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[7:5]	buck_vout1	Vout1 Output Adjust **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[4]	buck_testen	Buck Test Enable Bit 1 = Enabled 0 = Disablde **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[3]	buck_vout2_en	Buck SIDO Enable Bit 1 = Enabled 0 = Disablde **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[2]	buck_vout1_en	Buck SIDO Enable Bit 1 = Enabled 0 = Disablde **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[1]	buck_en	Buck Enable Bit 1 = Enabled 0 = Disablde **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	buck_bp	Bypass Enable Bit

		1 = Enabled 0 = Disablde **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.15 LP_MISC

Register	Offset	R/W	Description	Reset Value
LP_MISC	ANA_BA+0x38	R/W	Other High Voltage Analog Control Register	0x0000_0006

Bits	Description	
[31:8]	Reserved	Reserved.
[3:1]	rst_vref1p2v_trim	register for rst vref trim (need 3V sync, support auto_3V sync)
[0]	ldobypass	register for ldo bypass(global) (need 3V sync, support auto_3V sync)

3.2.6.16 ANA_ADCLDO

Register	Offset	R/W	Description	Reset Value
ANA_ADCLDO	ANA_BA+0x40	R/W	ADCLDO Configuration Register	0x0000_0042

Bits	Description	
[31:8]	Reserved	Reserved.
[8]	rxadcldoprbiptatref	IPTAT Reference Voltage Control Signal 1 = Reference voltage abnormal 0 = Reference voltage normal
[7]	rxadcldoiprbptatcore	ADCLDO's OTA Bias Enable Signal 1 = With bias current 0 = Without bias current
[6:3]	rxadcldovtrim	RXADC LDO Output Adjust Option 4'b0000 0.986V 4'b1000 1.2V 4'b1111 1.38V
[2]	rxadcldobypass	ADCLDO's Bypass Signal 1 = Bypass power tunnel. The output voltage is power voltage. 0 = Power tunnel works normally.
[1]	rxadcldoendly	ADC LDO Reference Voltage Connection/Disconnection Control Bit 1 = Disconnection 0 = Connection
[0]	rxadcldoen	RXADC LDO Enable Control Bit 1 = Enabled 0 = Disabled

3.2.6.17 ANA_RFFELDO

Register	Offset	R/W	Description	Reset Value
ANA_RFFELDO	ANA_BA+0x44	R/W	RFFELDO Configuration Register	0x0000_0042

Bits	Description	
[31:8]	Reserved	Reserved.
[8]	rffeldoprbiptatref	IPTAT Reference Voltage Control Signal 1 = Reference voltage abnormal 0 = Reference voltage normal
[7]	rffeldoiprbptatcore	RFFELDO's OTA Bias Enable Signal 1 = With bias current 0 = Without bias current
[6:3]	rffeldovtrim	RFFELDO Output Adjust Option 4'b0000 0.986V 4'b1000 1.2V 4'b1111 1.38V
[2]	rffeldobypass	RFFELDO's Bypass Signal 1 = Bypass power tunnel. The output voltage is power voltage. 0 = Power tunnel works normally.
[1]	rffeldoendly	RFFE LDO Reference Voltage Connection/Disconnection Control Bit 1 = Disconnection 0 = Connection
[0]	rffeldoen	RFFE LDO Enable Control Bit 1 = Enabled 0 = Disabled

3.2.6.18 ANA_VCOLDO

Register	Offset	R/W	Description	Reset Value
ANA_VCOLDO	ANA_BA+0x48	R/W	VCOLDO Configuration Register	0x0000_0042

Bits	Description	
[31:8]	Reserved	Reserved.
[8]	fsynvcoldoprbiptatref	IPTAT Reference Voltage Control Signal 1 = Reference voltage abnormal 0 = Reference voltage normal
[7]	fsynvcoldoiprbptatcore	FSYNLDO's OTA Bias Enable Signal 1 = With bias current 0 = Without bias current
[6:3]	fsynvcoldovtrim	FSYN LDO Output Adjust Option
[2]	fsynvcoldobypass	FSYNLDO's Bypass Signal 1 = Bypass power tunnel. The output voltage is power voltage. 0 = Power tunnel works normally.
[1]	fsynvcoldoendly	FSYNVCOLDO Reference Voltage Connection/Disconnection Control Bit 1 = Disconnection 0 = Connection
[0]	fsynvcoldoen	FSYN LDO Enable Control Bit 1 = Enabled 0 = Disabled

3.2.6.19 ANA_DFT

Register	Offset	R/W	Description	Reset Value
ANA_DFT	ANA_BA+0x4C	R/W	Analog DFT Configuration Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved.
[14]	dft_en_v	Enable DFT voltage output mode
[13]	dft_en_i	Enable DFT current output mode
[12]	dft_en_clk	Enable DFT clock output mode
[11]	dft_flag	Enable DFT flag output mode
[10:7]	dft_v_sel	Registers for selecting proper output voltage
[6]	dft_i_sel	Registers for selecting proper output current
[5:3]	dft_clk_sel	Registers for selecting proper output clock
[2:1]	dft_flag_sel	Registers for selecting proper output flag
[0]	reg_v_dft_bypass	register for DFT voltage output mode buffer bypass function

3.2.6.20 ANA_TEMP

Register	Offset	R/W	Description	Reset Value
ANA_TEMP	ANA_BA+0x50	R/W	Temperature Detect Register	0x0000_0000

Bits	Description	
[31:4]	Reserved	Reserved.
[3:2]	temp_gain	Register for temp gain(slope)
[1]	temp_test	Register for temp test mode
[0]	temp_en	Enable for temp

3.2.6.21 ANA_MISC

Register	Offset	R/W	Description	Reset Value
ANA_MISC	ANA_BA+0x54	R/W	Other Low Voltage Analog Control Register	0x0000_3AC0

Bits	Description	
[31:28]	Reserved	Reserved.
[28]	usb_pu2	USB PAD Control
[27]	usb_pu	USB PAD Control
[26]	usb_en	USB PAD Control
[25:20]	Flash_pad_pden	Flash PAD Control
[19:14]	Flash_pad_puen	Flash PAD Control
[13:8]	Flash_pad_dien	Flash PAD Control
[7]	ls_en_gpio	GPIO Level Shift Enable Bit 1 = Enable 0 = Disable **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[6]	ls_en_global	Analog Level Shift Enable Bit 1 = Enable 0 = Disable **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[5]	pmufsynicpsel	Register for fsyn icp double
[4]	pmuvrefbufen	VBG_BUFFER's PowerDown Signal 1 = Work normally 0 = PowerDown is valid
[3]	pmuldo_tm	LDO test mode(all ldo reuse)
[2]	xtl_quick_en	XHL Quick Start Control Signal. This bit is used for testing. RCH is divided to CLK_32K and output to the two pins of XTL.
[1]	Mbist_mode	mbist Mode Enable Bit 1 = Enable 0 = Disable **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)
[0]	Scan_mode	Scan Mode Enable Bit 1 =Enable 0 = Disable **Note:** this bit is only valid when synchronized to 3V register, which support auto_3V synchronization(Reg_sync_ctrl set to 1)

3.2.6.22 ANA_MISC2

Register	Offset	R/W	Description	Reset Value
ANA_MISC	ANA_BA+0x58	R/W	Other Low Voltage Analog Control Register2	0x0000_0004

Bits	Description	
[15:0]	Valid_removal_cyles	For usb insert detection usage The conditions for USB deivce to detect the insertion event are: • First in the unplugged state • Then detect that DIP&DIM=0 When detecting the removal state, the DIP=DIM=1 state must maintain the specified number of 48MHz cycles to be considered as a valid removal state.

3.2.6.23 ANA_RESERVED

Register	Offset	R/W	Description	Reset Value
ANA_RESERVED	ANA_BA+0x5C	R/W	Reserved Analog Register	0xFF00_FF00

Bits	Description	
[31:24]	Reserved_3v_h	Reserved (need 3V sync, support auto_3V sync)
[23:18]	Reserved_3v_l[7:2]	Reserved (need 3V sync, support auto_3V sync)
[17]	BUCK_VLMT_EN	Enable Buck voltage limiting (need 3V sync, support auto_3V sync) 1 = Enable 0 = Disable
[16]	Reserved_3v_l[0]	Reserved (need 3V sync, support auto_3V sync)
[15:8]	Reserved_1p5v_h	Reserved
[7:0]	Reserved_1p5v_l	Reserved

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 range	Symbol
POR	After the power on/down voltage is higher/lower than the threshold voltage (typ: 1.65V), the reset release/pull down.	Digital 3V region; Digital 1.2V region.	PINRF
PAD_RESET	Reset pin is pressed or released	Digital 3V region (ROMFLAG will not be resset); Digital 1.2V region.	PINRF
LP_WDT_NRST	3V counter reaches the default value	Digital 3V region (ROMFLAG and its own control signal will not be resset); Digital 1.2V region.	PINRF & lp_wdt_flg (LP_WDT_CTRL[2])
LVR	After the power supply voltage is higher/lower than the threshold voltage (typ: 1.8V), the reset release/pull down	Digital 1.2V region.	LVRRF
BOD	After the power supply voltage is higher/lower than the threshold voltage (multiple classes), the reset release/pull down	Digital 1.2V region.	BODRF
CHIP	Software configuration	Digital 1.2V region.	CHIPRF
WDT WWDT	When the counter reaches the default value, the reset will be released by hardware after one cycle.	Digital 1.2V region.	WDTRF
SYS	Software configuration	Digital 1.2V region.(CPU debug module will not be reset).	SYSRF
CPU	Software configuration	Only CPU module, icache, flash's controller, ROM controller and decrypt. It is not recommended.	CPURF
Module_reset	Software configuration	Respective module reset	-

3.3.2.1 Reset Block Diagram

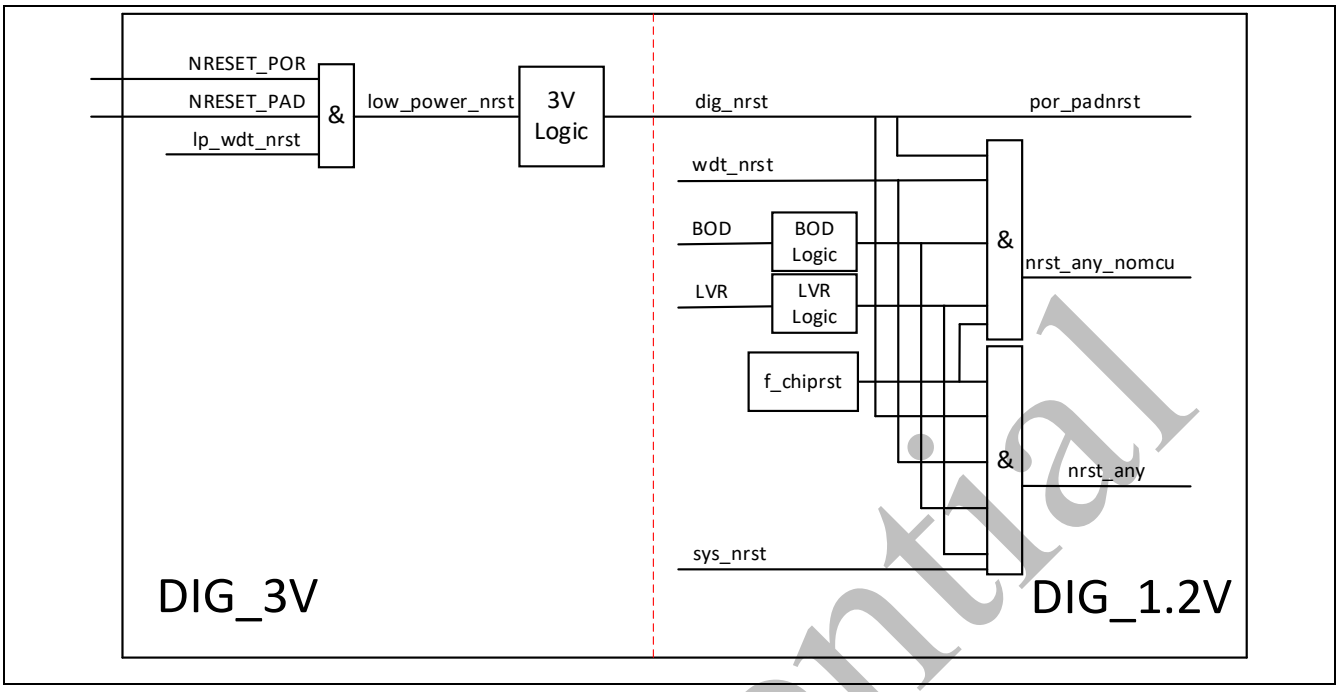


Figure 3-6 Reset Block Diagram

Por_padnrst: used internally by rcc, mainly used for reset flag power-on initialization reset.

Nrst_any_nomcu: used to reset the CPU debug module.

Nrst_any: used to reset all 1.2V logic except the CPU debug module.

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.

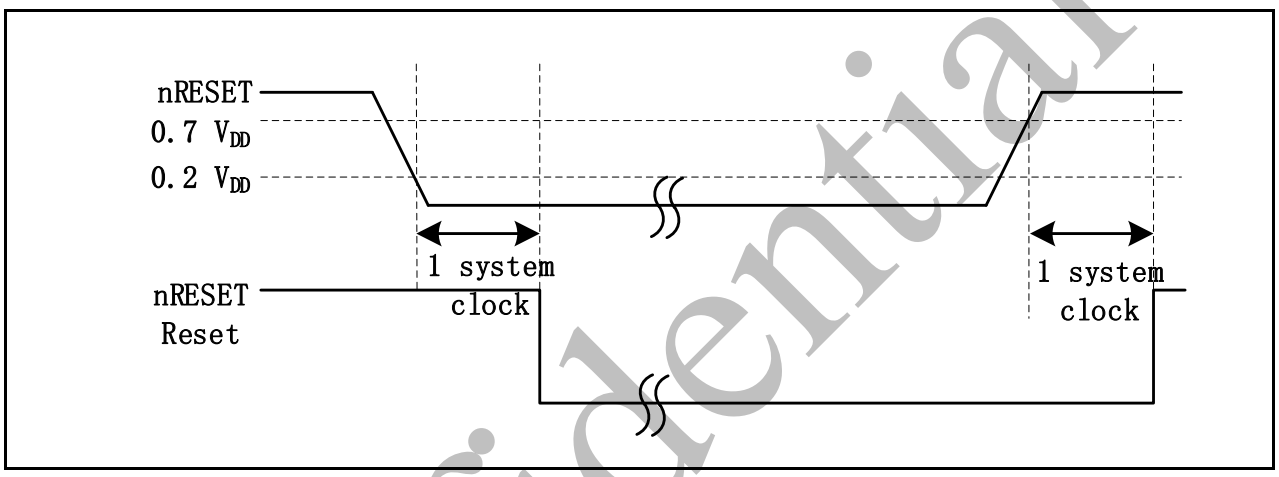


Figure 3-7 nRESET Reset Waveform

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.

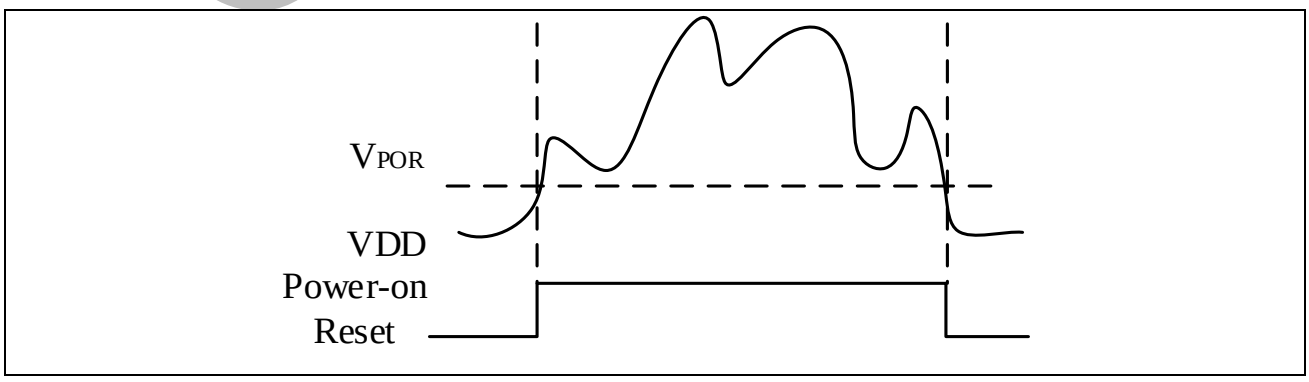


Figure 3-8 Power-on Reset (POR) Waveform

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

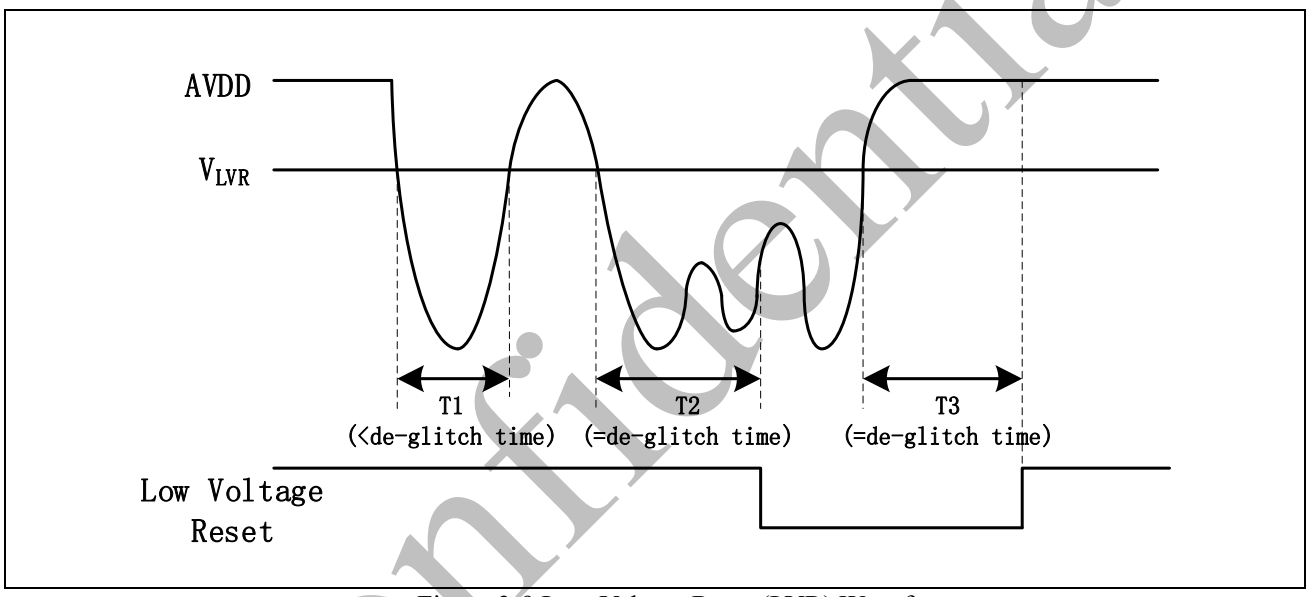


Figure 3-9 Low Voltage Reset (LVR) Waveform

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

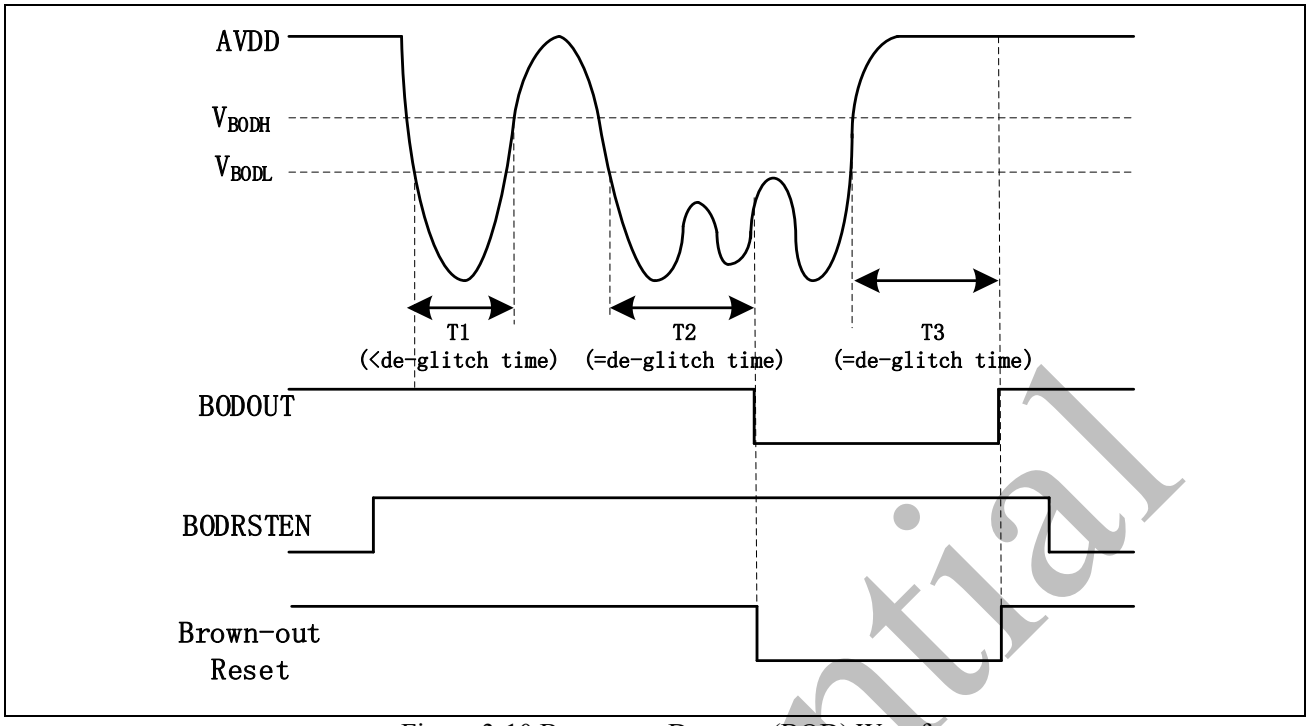


Figure 3-10 Brown-out Detector (BOD) Waveform

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.

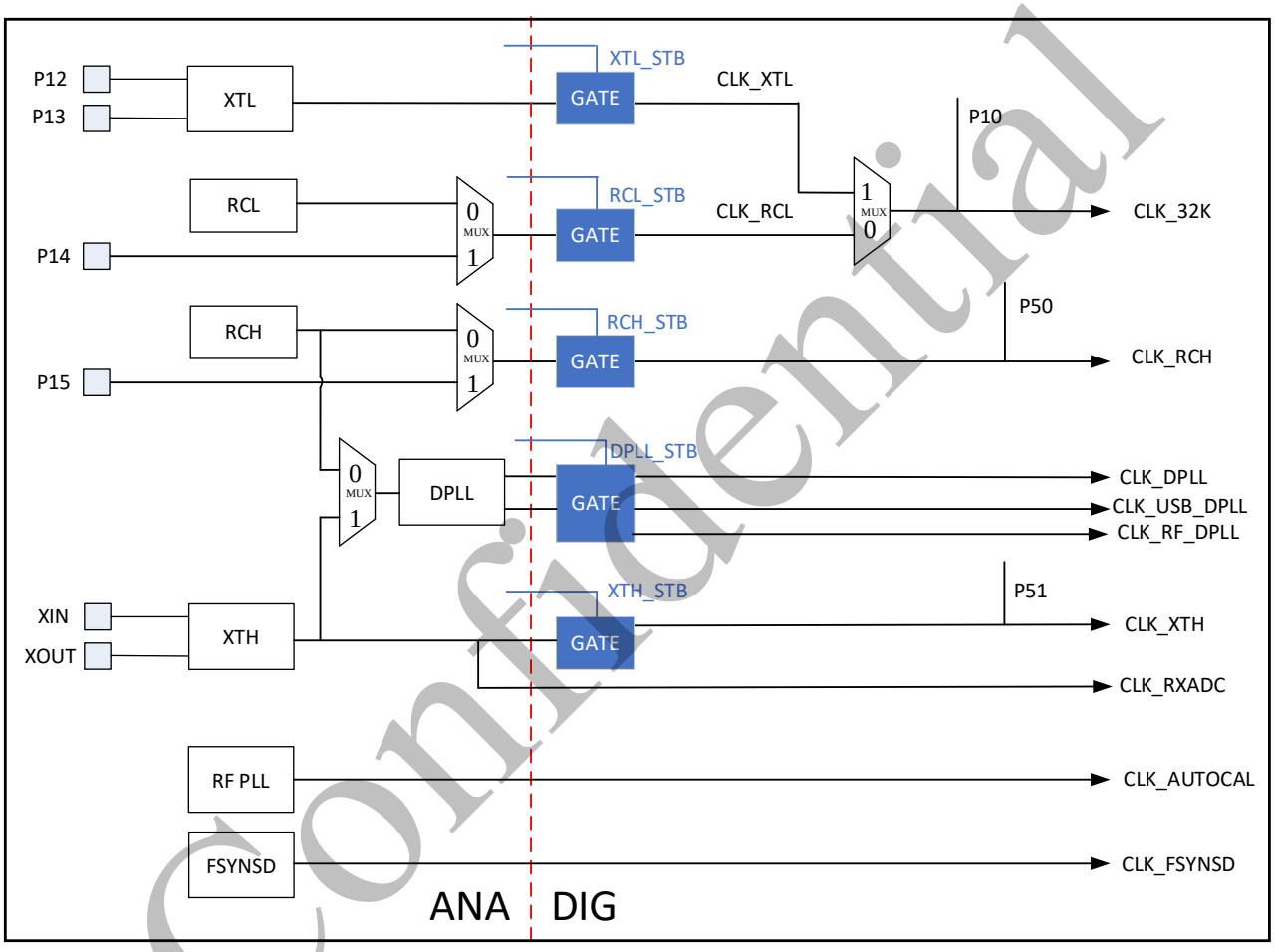


Figure 3-11 System Clock Generator Block Diagram

The analog clock are listed below:

Table 3-13 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

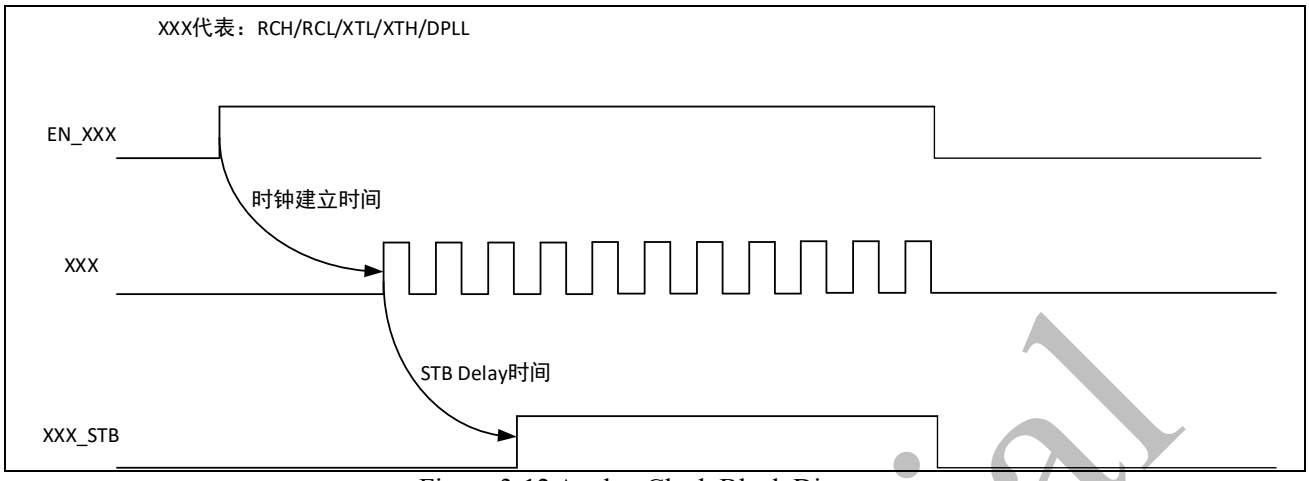


Figure 3-12 Analog Clock Block Diagram

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

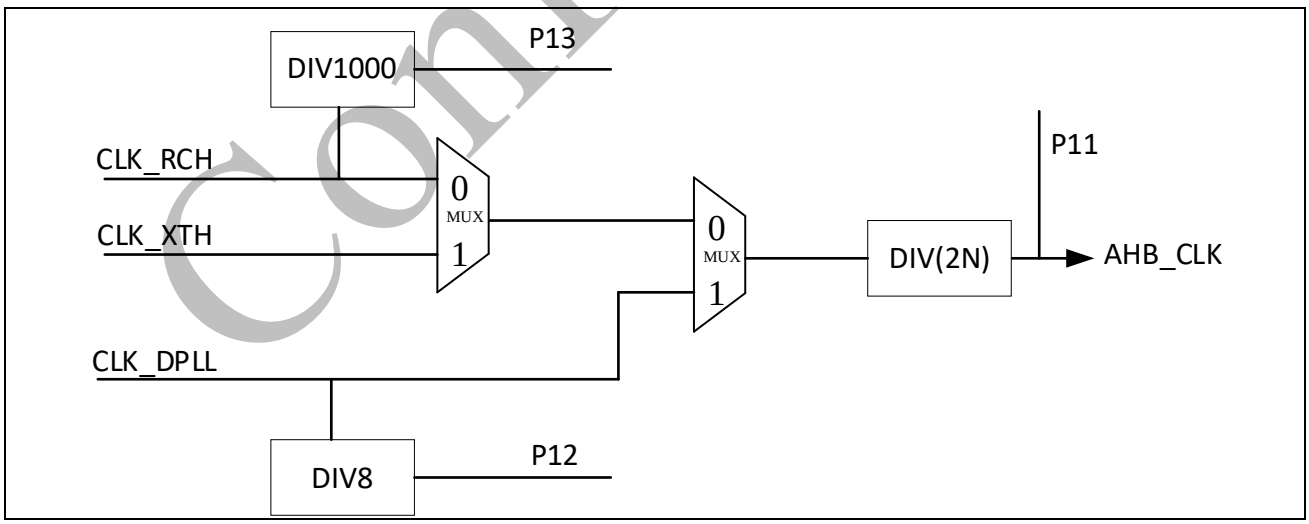


Figure 3-13 Digital Clock Block Diagram

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

In Figure 3-13, RCH is divided by 1000 and mux to P13. AHB_ CLK MUX to P11. DPLL MUX to p12 after divided by 8.

In iomux, the analog need to be output to digital RCL, XTL, RCH, XTH and DPLL clocks, and directly MUX to IO for subsequent debugging.

The clk of each periphral is as Figure 3-14.

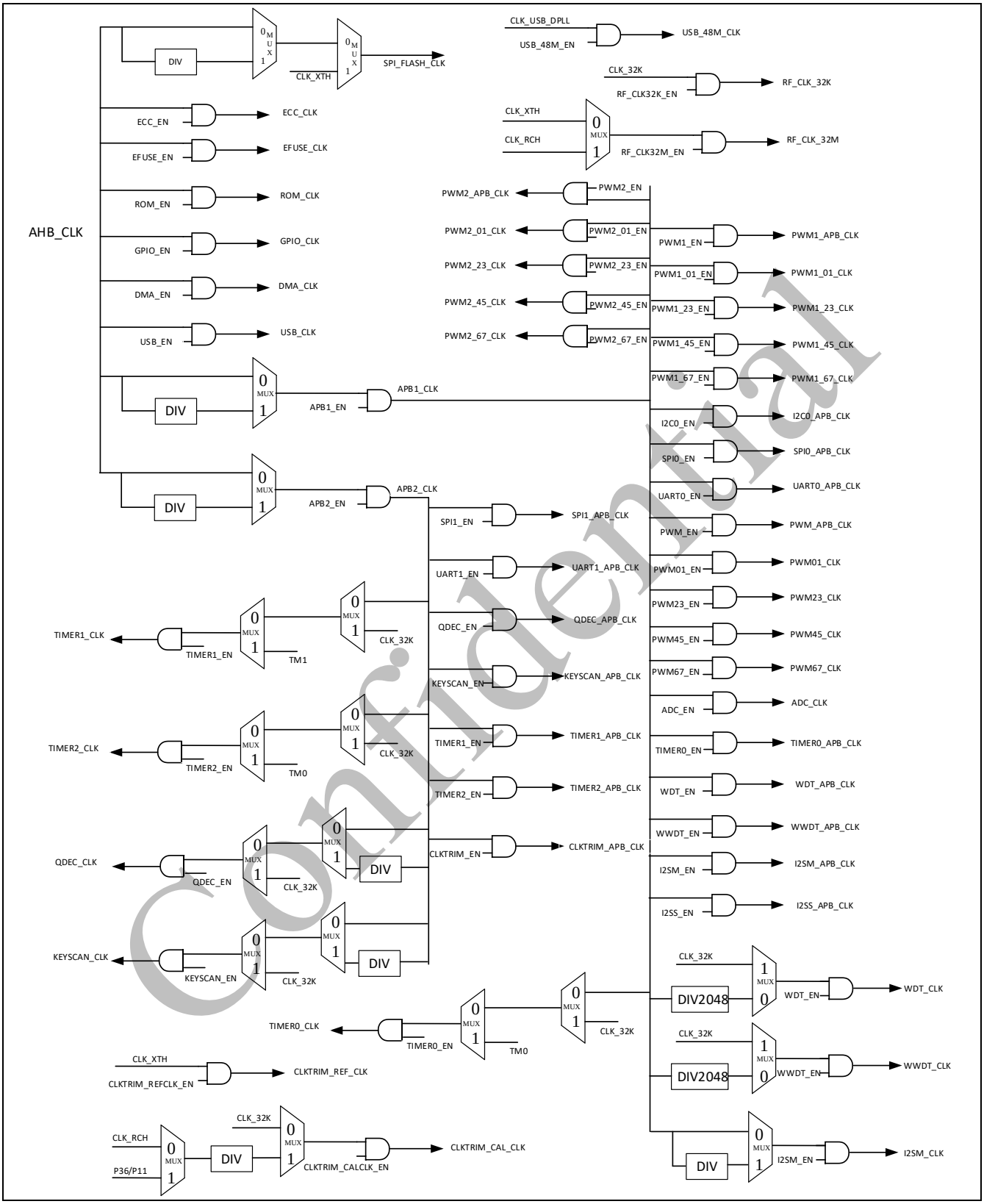


Figure 3-14 System Periphral Clock

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_0002
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+0x20	R/W	System Clock Global Control Register	0x0000_0003
RCL_CTRL	RCC_BA+0x24	R/W	RCL Control Register	0x0154_A010
RCH_CTRL	RCC_BA+0x28	R/W	RCH Control Register	0x0103_0080
XTL_CTRL	RCC_BA+0x2C	R/W	XTL Control Register	0x0003_002A
XTH_CTRL	RCC_BA+0x30	R/W	XTH Control Register	0x0000_0000
DPLL_CTRL	RCC_BA+0x34	R/W	DPLL Control Register	0x0000_0220
AHB_CLK_CTRL	RCC_BA+0x38	R/W	AHB Periphral Clock Control Register	0x0004_1C264
APB1_CLK_CTRL0	RCC_BA+0x3C	R/W	APB1 Periphral Clock Control Register	0x0000_0000
APB1_CLK_CTRL1	RCC_BA+0x40	R/W	APB1 Periphral Clock Control Register	0x0040_0000
APB2_CLK_CTRL0	RCC_BA+0x44	R/W	APB2 Periphral Clock Control Register	0x1906_4000
APB2_CLK_CTRL1	RCC_BA+0x48	R/W	APB1 Periphral Clock Control Register	0x00F4_0002
Act_32k_ctrl	RCC_BA+0x4c	R/W	BLE 32K clock control	0x0000_0000
Act_32k_basecorr	RCC_BA+0x50	R/W	BLE 32K clock counter	0x0000_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_0002

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]	Reserved	Reserved.
[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 or Power-on Reset (POR) Controller to indicate the previous reset source. 0 = No reset from nRESET pin, POR. 1 = Pin nRESET, POR 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	
[31:9]	Reserved	Reserved.
[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]	ECCRST	ECC Reset 0 = ECC normal operation. 1 = ECC reset.
[4]	EFUSERST	EFUSE Reset 0 = EFUSE normal operation. 1 = EFUSE reset.
[3]	LLRST	BLE LL Reset 0 = BLE normal operation. 1 = BLE 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.
[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	
[31:22]	Reserved	Reserved.
[24]	PWM2RST	PWM2 Reset 0 = PWM2 normal operation. 1 = PWM2 reset.
[23]	PWM1RST	PWM1 Reset 0 = PWM1 normal operation. 1 = PWM1 reset.
[22]	CLKTRIMRST	CLKTRIM Reset 0 = CLKTRIM normal operation. 1 = CLKTRIM reset.
[21]	QDECRST	QDEC Reset 0 = QDEC normal operation. 1 = QDEC reset.
[20]	KeyScanRST	KeyScan Reset 0 = KeyScan normal operation. 1 = KeyScan reset.
[19]	I2SSRST	I2S Slave Reset 0 = I2S Slave normal operation. 1 = I2S Slave reset.
[18]	I2SMRST	I2S Master Reset 0 = I2S Master normal operation. 1 = I2S Master reset.
[17:16]	Reserved	Reserved
[15]	GPIORST	GPIO (P0~P5) Controller Reset 0 = GPIO controller normal operation. 1 = GPIO controller reset.
[14]	TMR2RST	Timer2 Controller Reset 0 = Timer1 controller normal operation. 1 = Timer1 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	
[31:21]	Reserved	Reserved.
[22]	BOD_TST	BOD analog test enable control (need 3V sync, support auto_3V sync) 0 = BOD test disable 1 = LVR test enable
[21]	LVR_TST	LVR analog test enable control (need 3V sync, support auto_3V sync) 0 = LVR test disable 1 = LVR test enable
[20:18]	BOD_SEL	BOD voltage threshold select control (need 3V sync, support auto_3V sync) 000 = 1.95V 001 = 2.15V 010 = 2.35V 011 = 2.55V 100 = 2.75V 101 = 2.95V
[17]	EN_BOD	BOD enable control (need 3V sync, support auto_3V sync) 0 = BOD is disable 1 = BOD is enable
[16]	EN_LVR	LVR enable control (need 3V sync, support auto_3V 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	Brown-out Reset Enable Bit (Write Protect) The default value is set by flash controller user configuration register CBORST(CONFIG0[20]) bit. If config0 bit[20] is set to 1, default value of BODRSTEN is 0. If config0 bit[20] is set to 0, default value of BODRSTEN is 1. 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_2020

Bits	Description	
[31:14]	Reserved	Reserved.
[13:8]	LVRDBSEL	LVR De-Bounce (glitch) time Control register [0]=1: about 2^4 SYS clock [1]=1: about 2^7 SYS clock [2]=1: about 2^9 SYS clock [3]=1: about 2^11 SYS clock [4]=1: about 2^13 SYS clock [5]=1: about 2^15 SYS 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	BOD De-Bounce (glitch) time Control register [0]=1: about 2^4 SYS clock [1]=1: about 2^7 SYS clock [2]=1: about 2^9 SYS clock [3]=1: about 2^11 SYS clock [4]=1: about 2^13 SYS clock [5]=1: about 2^15 SYS 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 Clock top control

Register	Offset	R/W	Description	Reset Value
CLK_TOP_CTRL	RCC_BA+0x20	R/W	System clock global control register	0x0000_0003

Bits	Description		
[31:24]	Reserved	-	Reserved
[23:20]	APB2_DIV	RW	APB2clock frequency division, based on AHB Clock [3:0]=0: clock does not divide [3:0]=N: clock is divided, clock = 1/(2*N), (0<N<16)
[19:16]	APB1_DIV	RW	APB1 clock frequency division, based on AHB Clock [3:0]=0: clock does not divide [3:0]=N: clock is divided, clock = 1/(2*N), (0<N<16)
[15:12]	AHB_DIV	RW	AHB clock frequency division, based on System Clock [3:0]=0: clock does not divide [3:0]=N: clock is divided, clock = 1/(N+1), (0<N<16)
[10]	CLK32K_SEL	RW	0: RCL; 1: XTL (need 3V sync, support auto_3V sync)
[9:8]	SYS_CLK_SEL	RW	System Clock Selection, default as RCH 00: RCH 01: XTH 10/11: DPLL
[7:5]	Reserved	-	Reserved
[4]	DPLL_EN	RW	DPLL Enable Bit. (need 3V sync, support auto_3V sync)
[3]	XTH_EN	RW	External High-speed RC Enable Bit (need 3V sync, support auto_3V sync)
[2]	XTL_EN	RW	External 32K Low-speed RC Enable Bit (need 3V sync, support auto_3V sync)
[1]	RCH_EN	RW	Internal High-speed RC Enable Bit (need 3V sync, support auto_3V sync)
[0]	RCL_EN	RW	Internal 32K Low-speed RC Enable Bit (need 3V sync, support auto_3V sync)

3.3.5.7 RCL_CTRL

Register	Offset	R/W	Description	Reset Value
RCL_CTRL	RCC_BA+0x24	R/W	RCL Control Register	0x0155_0820

Bits	Description		
[31:28]	Reserved	-	Reserved
[27]	clk_sel	RW	Register for RCO_LOSC clock selection (need 3V sync, support auto_3V sync)
[26:25]	startup	RW	RCO_LOSC startup counter register (need 3V sync, support auto_3V sync)
[24]	Stable	RO	RCO_LOSC clock stable flag (need 3V sync, support auto_3V sync)
[23:17]	Reserved	-	Reserved
[16:14]	freq_coarse	RW	Self-calibration coarse adjustment gear is phy_soc_rco_losc_rsvd_1<2:0> Default value: 3'b100
[13:12]	Reserved	-	Reserved
[11:6]	freq_precision	RW	Self-calibration coarse adjustment gear is phy_soc_rco_losc_freq_fine<5:0> Default value: 6'b100000
[5:0]	freq_fine	RW	Self-calibration coarse adjustment gear is phy_soc_rco_losc_freq_coarse<5:0> Default value: 6'b100000

3.3.5.8 RCH_CTRL

Register	Offset	R/W	Description	Reset Value
RCH_CTRL	RCC_BA+0x28	R/W	RCH Control Register	0x0103_0080

Bits	Description		
[31:25]	Reserved	-	Reserved
[24]	Stable	RO	RCO_HOSC clock stable flag
[17:16]	Startup	RW	RCO_HOSC startup counter register (need 3V sync, support auto_3V sync)
[15:9]	Reserved	-	Reserved
[8]	clk_sel	RW	Register for RCO_HOSC clock selection (need 3V sync, support auto_3V sync)
[7:0]	freq	RW	RCH_HOSC frequency register, 0.4% per step

3.3.5.9 XTL Control (XTL_CTRL)

Register	Offset	R/W	Description	Reset Value
XTL_CTRL	RCC_BA+0x2C	R/W	XTL Control Register	0x0003_002A

Bits	Description		
[31:25]	Reserved	-	Reserved
[24]	Stable	RO	XO_LOSC clock stable flag
[23:18]	Reserved	-	Reserved
[17:16]	Startup	RW	XO_LOSC startup counter register (need 3V sync, support auto_3V sync)
[15:3]	Reserved	-	Reserved
[2:0]	core	RW	XO_LOSC core bias register (need 3V sync, support auto_3V sync)

3.3.5.10 XTH_CTRL

Register	Offset	R/W	Description	Reset Value
XTH_CTRL	RCC_BA+0x30	R/W	XTH Control Register	0x0000_0000

Bits	Description		
[31:25]	Reserved	-	Reserved
[24]	Stable	RO	XO_FSYN clock stable flag
[23:7]	Reserved	-	Reserved
[6:4]	xocap_sel	RW	XO cap select control
[3]	Reserved	-	Reserved
[2]	startup_fast	RW	XO_FSYN fast startup register (need 3V sync, support auto_3V sync)
[1]	startup_counter	RW	XO_FSYN startup counter register (need 3V sync, support auto_3V sync)
[0]	fsyn_digclken	RW	XO_FSYN digital clock register (need 3V sync, support auto_3V sync)

3.3.5.11 DPLL_CTRL

Register	Offset	R/W	Description	Reset Value
DPLL_CTRL	RCC_BA+0x34	R/W	DPLL Control Register	0x0000_0220

Bits	Description		
[31:25]	Reserved	-	Reserved
[24]	Stable	RO	DPLL stable flag
[17:16]	Startup	RW	PLL startup counter register
[15:10]	Reserved	-	Reserved
[10]	pll_tm	RW	Register for pll test mode
[9:2]	Reserved	-	Reserved
[1]	ref_clk	RW	PLL reference clock register 0: RCO 1: XO
[0]	freq	RW	PLL frequency register 0: 48MHz 1: 64MHz

3.3.5.12 AHB_CLK_CTRL

Register	Offset	R/W	Description	Reset Value
AHB_CLK_CTRL	RCC_BA+0x38	R/W	AHB Periphral Clock Control Register	0x0004_1C26

Bits	Description		
[31:21]	Reserved	-	Reserved
[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	-	Reserved
[14]	USB_48M_CLK_EN	RW	USB 48Mhz clock enable control
[13]	USB_AHB_CLK_EN	RW	USB AHB clock enable control
[12]	ECC_CLK_EN	RW	ECC clock enable control
[11]	EFUSE_CLK_EN	RW	EFUSE clock enable control. Default 1
[10]	Reserved	-	Reserved
[9:8]	BLE_CLK32M_SEL	RW	BlE 32MHz clock select 00 = 32M XTAL 01 = 32M RC 10 = 32M DPLL 11 = 32M DPLL
[7]	BLE_CLK32K_EN	RW	BLE 32KHz clock enable control
[6]	BLE_CLK32M_EN	RW	BLE 32MHz clock enable control
[5]	RCC_AHB_CLK_EN	RW	RCC ahb_clk_noscan enable control. Default 1
[4]	APB2_CLK_EN	RW	APB2 Clock Enable Bit 0 = All APB2 peripheral APB clocks are off. 1 = APB2_CLK enabled.
[3]	APB1_CLK_EN	RW	APB1 Clock Enable Bit 0 = All APB1 peripheral APB clocks are off. 1 = APB1_CLK enabled.
[2]	ST_CLK_EN	RW	System Tick clock enable 0 = Clock disable 1 = Clock enable
[1]	GPIO_CLK_EN	RW	GPIO Clock Enable Bit 0 = Disabled. 1 = Enabled.
[0]	DMA_CLK_EN	RW	DAM Clock Enable Bit 0 = Disabled. 1 = Enabled.

3.3.5.13 APB1_CLK_CTRL0

Register	Offset	R/W	Description	Reset Value
APB1_CLK_CTRL0	RCC_BA+0x3C	R/W	APB1 Periphral Clock Control Register	0x0000_0000

Bits	Description		
[31]	Pwm2cken	RW	Pwm apb clock enable 0: clock disable 1: clock enable
[30]	Pwm2_67cken	RW	Pwm67 clock enable 0: clock disable 1: clock enable
[29]	Pwm2_45cken	RW	Pwm45 clock enable 0: clock disable 1: clock enable
[28]	Pwm2_23cken	RW	Pwm23 clock enable 0: clock disable 1: clock enable
[27]	Pwm2_01cken	RW	Pwm01 clock enable 0: clock disable 1: clock enable
[26]	Pwm1cken	RW	Pwm apb clock enable 0: clock disable 1: clock enable
[25]	Pwm1_67cken	RW	Pwm67 clock enable 0: clock disable 1: clock enable
[24]	Pwm1_45cken	RW	Pwm45 clock enable 0: clock disable 1: clock enable
[23]	Pwm1_23cken	RW	Pwm23 clock enable 0: clock disable 1: clock enable
[22]	Pwm1_01cken	RW	Pwm01 clock enable 0: clock disable 1: clock enable
[21]	I2smcken	RW	I2S master clock enable 0: clock disable 1: clock enable
[20]	I2sscken	RW	I2S slave clock enable 0: clock disable 1: clock enable
[19:18]	tmr0sel	RW	00: APB1_CLK 01: RCL/XTL 10, 11: TM0_I
[17]	wwdtsel	RW	0: milli_pulse 1: RCL/XTL

[16]	wdtsel	RW	0: milli_pulse 1: RCL/XTL
[15:13]	Reserved	-	Reserved
[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 APB1_CLK_CTRL1

Register	Offset	R/W	Description	Reset Value
APB1_CLK_CTRL1	RCC_BA+0x40	R/W	APB1 Periphral Clock Control Register	0x0000_0000

Bits	Description		
[31:23]	Reserved		Reserved
[22]	I2S_MASTER_DMA_REQ	RW	When I2S_MASTER_DMA_REQ=1, the master enables the DMA request channel When I2S_MASTER_DMA_REQ=0, the slave enables the DMA request channel Default value 1
[21:18]	I2S_MCLK_DIV	RW	I2S MCLK_DIV Frequency Dicider, based on I2S MCLK(output to codec) [2:0]=0: clock does not divide [2:0]=N: clock is divided, clock = 1/(N+1), (0<N<8)
[17:16]	I2S_MCLK_SEL	RW	I2S MCLK Selection, default as RCH 00 = RCH 01 = XTH 10/11 = DPLL
[15:0]	i2s_master_div	RW	I2S Master CLK_DIV 0: clock does not divide N: clock is divided, clock = 1/(2N), (0<N<2^16)

3.3.5.15 APB2_CLK_CTRL0

Register	Offset	R/W	Description	Reset Value
APB2_CLK_CTRL0	RCC_BA+0x44	R/W	APB2 Periphral Clock Control Register	0x1906_4000

Bits	Description		
[31:23]	Qdec_div	RW	Qdec clock source divisor 0: clock is not divided N: clock is divided, clock = 1/(2N), (0<N<512)
[22]	qdeccken	RW	QDEC clock enable 0: Clock disable 1: Clock enable
[21:13]	Keyscan_div	RW	Keyscan clock source divisor 0: clock is not divided N: clock is divided, clock = 1/(2N), (0<N<512)
[12]	keyscancken	RW	KeyScan clock enable 0: Clock disable 1: Clock enable
[11:10]	tmr2sel	RW	00: APB2_CLK 01: RCL/XTL 10, 11: TM2_I
[9:8]	tmr1sel	RW	00: APB2_CLK 01: RCL/XTL 10/11: TM1_I
[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

3.3.5.16 APB2_CLK_CTRL1

Register	Offset	R/W	Description	Reset Value
APB2_CLK_CTRL1	RCC_BA+0x48	R/W	APB2 Periphral Clock Control Register	0x00F4_0002

Bits	Description		
[31:24]	Reserved	-	Reserved
[23:15]	xth_div	RW	Xth divider for xtl quick setup Default value：0x1e8
[14]	keyscan_clk_sel	RW	0: apb_clk 1: CLK32K
[13]	qdec_clk_sel	RW	0: apb_clk 1: CLK32K
[12:4]	calclk_div	RW	-
[3:2]	Calclk_sel	RW	x0: 32K clk 01: 32M RC clk 11: ext clk
[1]	Clktrim_calcken	RW	Clktrim cal/ref clock enable 0: clock disable 1: clock enable
[0]	Clktrimcken	RW	Clktrim clock enable 0: clock disable 1: clock enable

3.3.5.17 Act_32k_ctrl

Register	Offset	R/W	Description	Reset Value
Act_32k_ctrl	RCC_BA+0x4c	R/W	BLE 32K clock control	0x0000_0000

Bits	Description		
[31:22]	finecorr	RW	The fine correct value of ACT CNT, units is 32MHz
[21:12]	fineoffset	R	The offset between SLP_32K and ACT_32K, units is 32MHz
[11:4]	reserved		reserved
[3]	Act_clk_up_done	R	Active clock update done flag 0: do not update done 1: done, active clock is being used
[2]	Slp_cnt_up	RW	Sleep count update act_32k count control 0: disable 1: enable, clear to 0 by hardware when update is done
[1]	ll_clk_32k_sel	RW	Link layer 32k clock source select 0: SLP_CLK (32K RC or 32K XTL) 1: ACT_32K
[0]	Ack_ 32k_tmr_en	RW	Active 32k clock enable 0: clock disable 1: clock enable

3.3.5.18 Act_32k_basecorr

Register	Offset	R/W	Description	Reset Value
act_32k_basecorr	RCC_BA+0x50	R/W	BLE 32K clock counter	0x0000_0000

Bits	Description		
[31:0]	basecorr	RW	The base correct value of ACT CNT, units is 32KHz

3.4 Flash Memory Controller (FMC)

3.4.1 Overview

In PAN108 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.

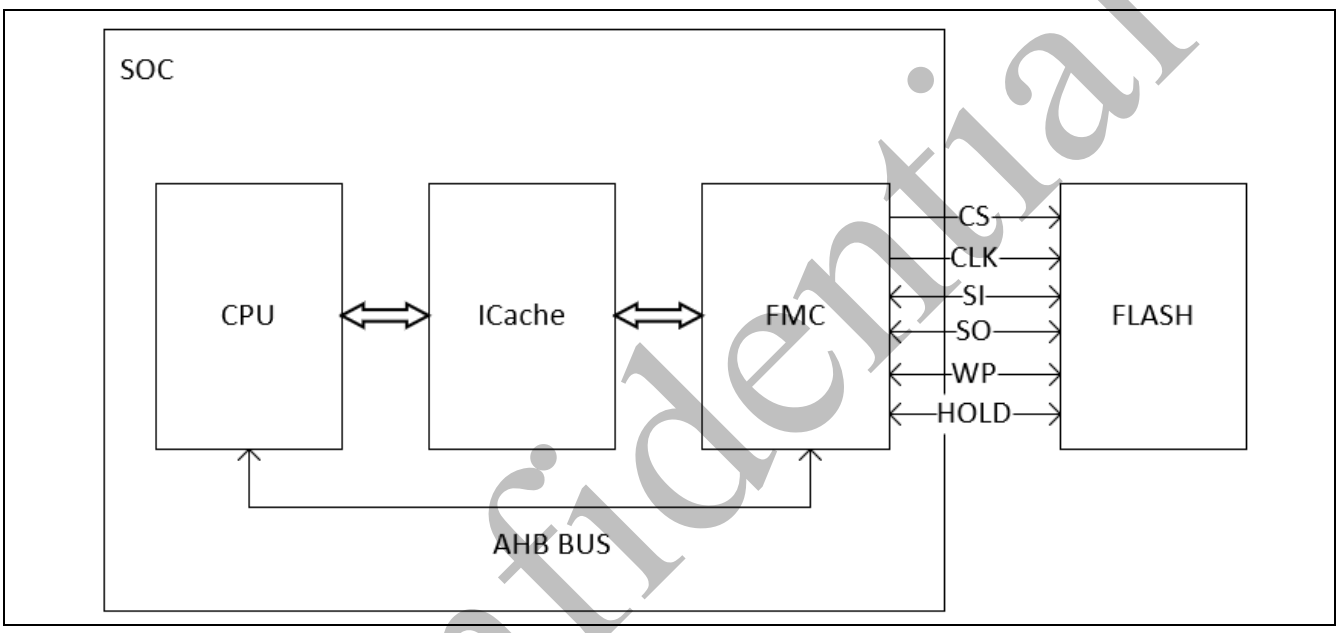


Figure 3-15 SPI Flash Controller Diagram

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 Feature

- 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.

3.4.3 I-cache

The I-cache applied in the PAN108 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

PAN108 series supports vector table mapping.

The PAN108 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

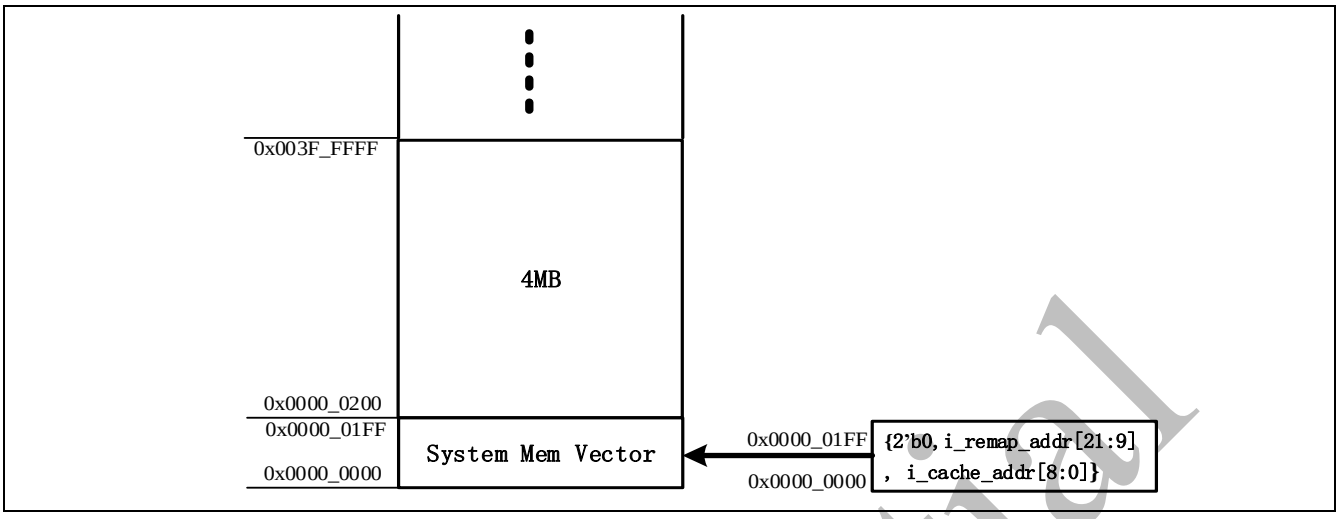


Figure 3-16 Remap Diagram

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.

3.4.4.2 Timing Calibration

The flash output data on the falling edge of the clock, and there is a delay of about 6ns from the falling edge to stable state. There is also a transmission delay before the data reaches the sampling register, and the total delay is about 10ns. In order to improve the operating frequency of the flash, the sampling edge of PAN108 series can be configured to sample the correct data in the sampling register.

1. The clock is relatively slow, and the delay is less than half a clock cycle. The sampling edge can be the first rising edge of the clock, as shown in the figure at time b. It can also be the first falling edge of the clock.

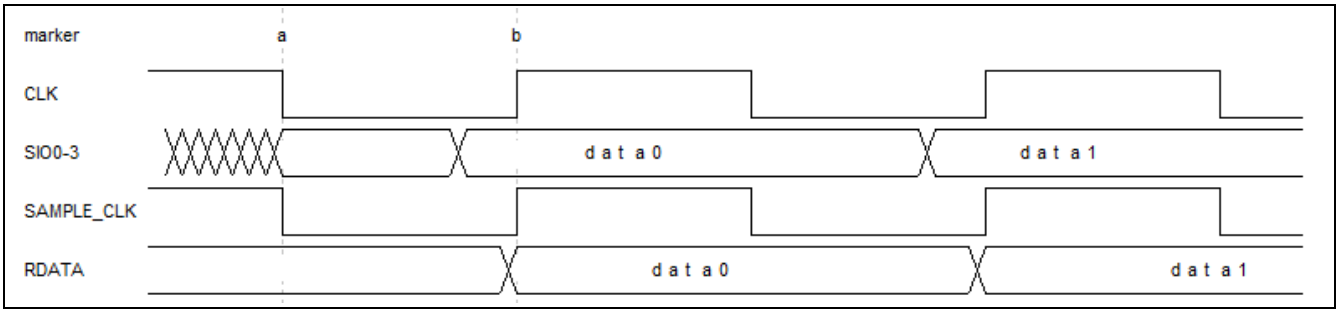


Figure 3-17 Timing Calibration(t_{delay}<1/2*CLK)

2. The clock is relatively fast, and the delay is greater than half a clock cycle and less than one clock cycle. The sampling edge can be the first falling edge of the clock, as shown at time b, or the second rising edge of the clock.

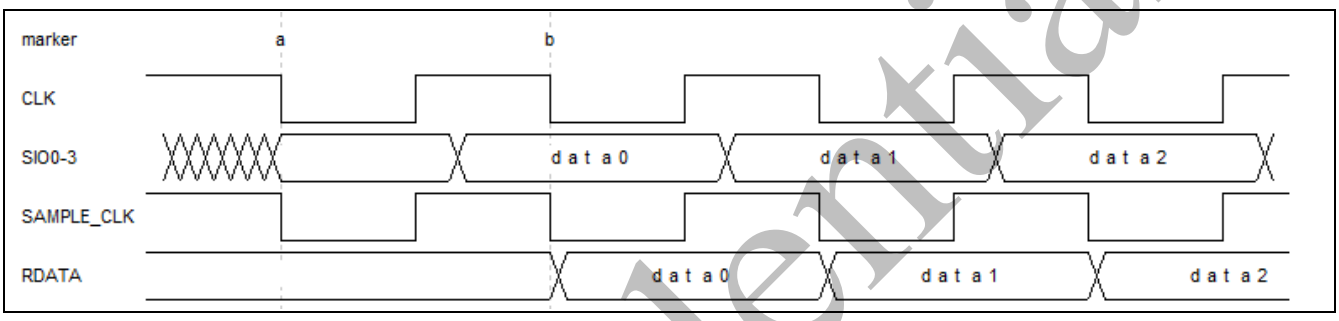


Figure 3-18 Timing Calibration(1/2*CLK <t_{delay}<1*CLK)

3. The delay is greater than one clock cycle and less than one and a half clock cycles. The sampling edge can be the second rising edge of the clock, at time b in the figure, or the second falling edge of the clock.

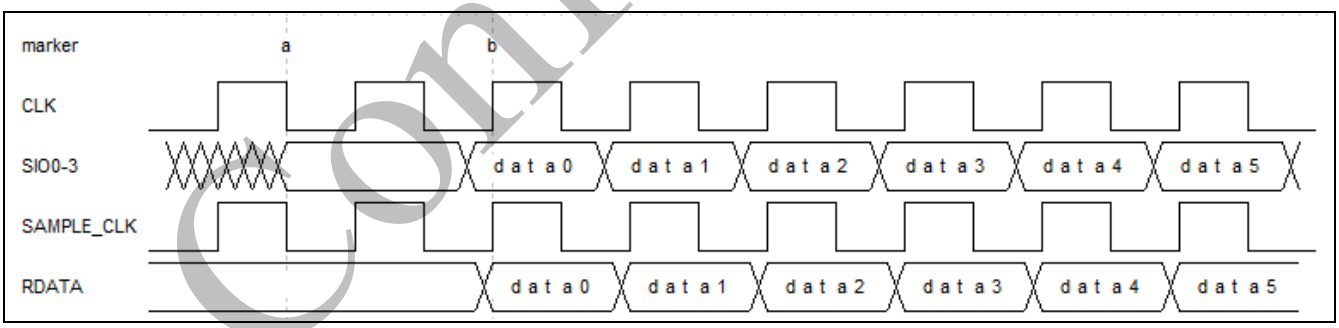


Figure 3-19 Timing Calibration(1*CLK <t_{delay}<3/2*CLK)

Register

[23:21]	clk_dly	Control the delay clock cycle number of FMC sampling clock compared to the main clock, which can be set to 0-7.
[20]	rx_neg	Control FMC sampling clock edge to latch the flash return data 0: Latch data on the rising edge 1: Latch data on the falling edge

Software manual

1. Switch the program to RAM for execution;
2. Switch the flash operating frequency to high frequency, 64M;
3. Reasonably configure rx_neg and clk_dly registers, and configure the sampling edge;
4. Send read commands (READ, FAST_READ, DREAD, 2READ, QREAD, 4READ);
5. If the read data is consistent with the written data, it indicates that the sampling clock configuration is reasonable, and the program is switched back to flash to run, otherwise repeat steps 3 and 4.

Note: The value of cs_idle_cycle should be greater than or equal to the value of clk_dly.

3.4.5 SPI Flash Register Map

Register	Offset	R/W	Description	Reset Value
FMC Base Address FMC_BA = 0x4005_0000				
X_FL_CTL	FMC_BA+0x00	R	Read Data Register	0x0000_0000
X_FL_TRG	FMC_BA+0x04	R/W	Trigger Register	0x00
X_FL_ CONFIG	FMC_BA+0x005	R/W	Configure Register	0x00
X_FL_WDATA1	FMC_BA+0x006	R/W	Write Data Register 1	0x00
X_FL_WDATA2	FMC_BA+0x007	R/W	Write Data Register 2	0x00
X_FL_WDATA3	FMC_BA+0x008	R/W	Write Data Register 3	0x00
X_FL_WDATA4	FMC_BA+0x009	R/W	Write Data Register 4	0x00
X_FL_WDATA5	FMC_BA+0x00A	R/W	Write Data Register 5	0x00
X_FL_WDATA6	FMC_BA+0x00B	R/W	Write Data Register 6	0x00
X_FL_X_MODE	FMC_BA+0x00C	R/W	SPI Flash Mode Select Register	0x0000_1001
X_FL_X2_CMD	FMC_BA+0x010	R/W	X2 Command Select Register	0x3B
X_FL_X4_CMD	FMC_BA+0x011	R/W	X4 Command Select Register	0xEB
X_FL_DP_CMD	FMC_BA+0x12	R/W	DP command register	0xB9
X_FL_RDP_CMD	FMC_BA+0x13	R/W	RDP command register	0xAB
X_FL_REMAP_ADDR	FMC_BA+0x14	R/W	Remap address register	0x0000_0000
X_FL_DP_CTL	FMC_BA+0x18	R/W	Deep power down control register	0x0001_0001

3.4.6 SPI Flash Register Description

3.4.6.1 Read Data Register (X_FL_CTL)

Register	Offset	R/W	Description	Reset Value
X_FL_CTL	FMC_BA+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.

3.4.6.2 Trigger Register (X_FL_TRG)

Register	Offset	R/W	Description	Reset Value
X_FL_TRG	FMC_BA+0x04	R/W	Trigger Register	0x00

Bits	Descriptions	
[7:1]	Reserved	Reserved
[0]	trg	This bit can be started by software writing 1 and will be auto-cleared after hardware finished.

3.4.6.3 Configure Register (X_FL_CONFIG)

Register	Offset	R/W	Description	Reset Value
X_FL_ CONFIG	FMC_BA+0x05	R/W	Configure Register	0x00

Bits	Descriptions	
[7:3]	Reserved	Reserved
[2]	x_address_transaction	Does W_data2/3/4 transmit the address? 1: is the address, then the actual address is {w_data2, w_data3,w_data4}+0x1000 0: If it is not an address, no changes will be made
[1]	info_en	The first 4kB of the physical address of the Flash is used as the info area 1: fmc can read and write this area 0: If tx_address_transaction==1, flash controller can not erase and write this area, if forced to erase the info area, then the actual erasing area is 4k-8k; if tx_address_transaction set as 0, then flash controller can read and write this area area When (info_en==1) || (info_en==0 && tx_address_transaction==0), flash controller can read and write the info 4k area of flash 0x0~0xfff
[0]	pp_active	Page Program function active bit. 0 = Not active. 1 = Active. Note: when the function is active, the hardware will be auto-clear.

3.4.6.4 Write Data Register1 (X_FL_WDATA1)

Register	Offset	R/W	Description	Reset Value
X_FL_WDATA1	FMC_BA+0x06	R/W	Write Data Register 1	0x00

Bits	Descriptions	
[7:0]	w_data1	Write Data Num=1 which is usually used to place command phase.

3.4.6.5 Write Data Register2 (X_FL_WDATA2)

Register	Offset	R/W	Description	Reset Value
X_FL_WDATA2	FMC_BA+0x07	R/W	Write Data Register 2	0x00

Bits	Descriptions	
[7:0]	w_data2	Write Data Num=2 which is usually used to place address phase.

3.4.6.6 Write Data Register3 (X_FL_WDATA3)

Register	Offset	R/W	Description	Reset Value
X_FL_WDATA3	FMC_BA+0x08	R/W	Write Data Register 3	0x00

Bits	Descriptions	
[7:0]	w_data3	Write Data Num=3 which is usually used to place address phase.

3.4.6.7 Write Data Register4 (X_FL_WDATA4)

Register	Offset	R/W	Description	Reset Value
X_FL_WDATA4	FMC_BA+0x09	R/W	Write Data Register 4	0x00

Bits	Descriptions	
[7:0]	w_data4	Write Data Num=4 which is usually used to place address phase.

3.4.6.8 Write Data Register5 (X_FL_WDATA5)

Register	Offset	R/W	Description	Reset Value
X_FL_WDATA5	FMC_BA+0x0A	R/W	Write Data Register 5	0x00

Bits	Descriptions	
[7:0]	w_data5	Write Data Num=5 which is usually used to place parameters phase.

3.4.6.9 Write Data Register6 (X_FL_WDATA6)

Register	Offset	R/W	Description	Reset Value
X_FL_WDATA6	FMC_BA+0x0B	R/W	Write Data Register 6	0x00

Bits	Descriptions	
[7:0]	w_data6	Write Data Num=6 which is usually used to place parameters phase.

3.4.6.10 SPI Flash Mode Select Register (X_FL_X_MODE)

Register	Offset	R/W	Description	Reset Value
X_FL_X_MODE	FMC_BA+0x0C	R/W	SPI Flash Mode Select Register	0x0000_1001

Bits	Descriptions	
[31:24]	Reserved	Reserved
[23:21]	clk_dly	Control the delay clock cycle number of FMC sampling clock compared to the main clock, which can be set to 0-7.
[20]	rx_neg	Control FMC sampling clock edge to latch the flash return data 0: Latch data on the rising edge 1: Latch data on the the falling edge
[19]	crc32_en	CRC32 enable bit. 0 = Disabled 1 = Enable Note: The hardware will not be cleared automatically.
[18]	enhance_mode	Enhance mode enable bit. 0 = Normal mode. 1 = Enhance mode. Note: only 2READ and 4READ support the enhance mode.
[17]	long_time_op	This bit should be enabled before sending the WRSR/PP/PE/CE command. After this bit is set to 1, the hardware will automatically send the RDSR command to read the WIP status. After detecting that the WIP is 0, the bit is automatically set to 0. When WIP is 1, the trg signal is also be set to 1.
[16]	en_burst_wrap	This bit should be enabled before sending the SBL (77h) command. The hardware is automatically cleared.
[15:8]	cs_idle_cyle	Control the time for CS signal pulled high between two commands. The value of cs_idle_cycle must be greater than the value of clk_dly.
[7:2]	Reserved	Reserved
[1:0]	x_mode	Mode Select bits. 00 = x1 mode. 01 = x2 mode 10 = x4 mode. 11 = Forbiden. Note: Determine the configuration of x_mode based on how many lines are used to fetch data.

3.4.6.11 X2 Command Select Register (X_FL_X2_CMD)

Register	Offset	R/W	Description	Reset Value
X_FL_X2_CMD	FMC_BA+0x10	R/W	X2 Command Select Register	0x3B

Bits	Descriptions	
[7:0]	x2_cmd	X2 Command Select. 0xBB = X2 command is select. Others = X2 command is not select.

3.4.6.12 X4 Command Select Register (X_FL_X4_CMD)

Register	Offset	R/W	Description	Reset Value
X_FL_X4_CMD	FMC_BA+0x11	R/W	X4 Command Select Register	0xEB

Bits	Descriptions	
[7:0]	x4_cmd	X4 Command Select. 0xEB = X4 command is select. Others = X4 command is not select.

3.4.6.13 X_FL_DP_CMD

Register	Offset	R/W	Description	Reset Value
X_FL_DP_CMD	FMC_BA+0x12	R/W		8'hB9

Bits	Descriptions	
[7:0]	dp_cmd	Deep sleep mode

3.4.6.14 X_FL_RDP_CMD

Register	Offset	R/W	Description	Reset Value
X_FL_RDP_CMD	FMC_BA+0x13	R/W		8'hAB

Bits	Descriptions	
[7:0]	rdp_cmd	Release from deep sleep mode

3.4.6.15 X_FL_REMAP_ADDR

Register	Offset	R/W	Description	Reset Value
X_FL_REMAP_ADDR	FMC_BA+0x14	R/W		0x0000_0000

Bits	Descriptions	
[31:0]	remap_addr	

3.4.6.16 X_FL_DP_CTL

Register	Offset	R/W	Description	Reset Value
X_FL_DP_CTL	FMC_BA+0x18	R/W		0x0001_0001

Bits	Descriptions	
[31:8]	rdp_wt_cnt	The waiting time after the RDP command has been sent, which is take flash_clk as unit and generally needs to be greater than or equal to 8us. For example, under a 32MHz clock, the value of this register must be greater than or equal to 0x100, and under a 64MHz clock, the value of this register must be greater than or equal to 0x200.
[7:1]	Reserved	Reserved
[0]	dp_en	DP/RDP send command enable bit 1 = enabled 0 = Disabled

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 PN108 chip has an encryption function.

3.5.2 Feature

- 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

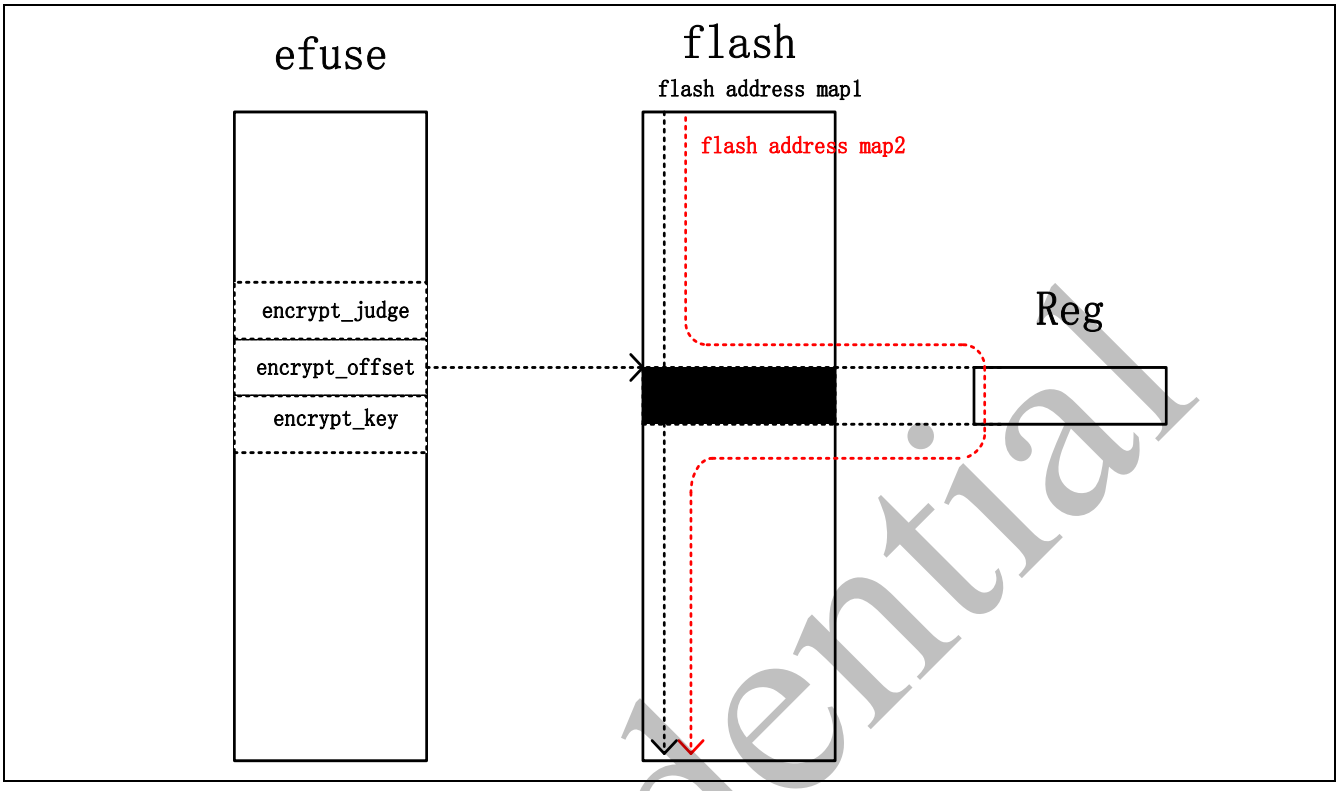


Figure 3-20 Encryption Block Diagram

In order to improve the encryption performance, efuse/AES and other IPs are introduced to cooperate with the encryption and decryption schemes. In Figure 3-20, 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, which are as follows. Please refer to the chapter of efuse planning space for specific address reservations.

1. **encrypt_judge(Encryption and decryption switch)**

   The main function of encrypt_judge are shown below:

   1) Used to determine whether some areas in the flash are encrypted, and enable the hardware logic of encryption and decryption mode 2.
   2) Encryption key software read permission control.
   3) Used to determine whether debug mode shielding is required, and enable the hardware

logic of encryption and decryption mode 1.

The specific meaning are shown below:

1) 2'b11: Indicates that the hardware logic of the decryption mode is fully open (hardware logic 1&hardware logic 2). If the key is programmed correctly, the software does not need to be read out, that is, no access to the key/encrypted address, etc.
2) 2'b01: Only the hardware logic 2 is turned on. If the key is programmed correctly, the software does not need to be read out, that is, no access to the key/encrypted address, etc.
3) 2'bx0: Do not open the encryption and decryption process, the hardware logic of the decryption mode is all off.

2. **encrypt_offset(Encryption and decryption offset)**

   Encrypt_offset is used to specify the offset address of the flash encryption area in the case of encryption. The size of the encryption area is fixed, such as 256/512/1024 bytes. They must be the basic erasing unit of the flash;

   Note: The size of the flash encryption area should be determined according to the selected external flash type to determining the above two parameters. Commonly, the basic erasing unit of the external flash is 256Bytes, PAN108 series adopts 256Bytes.

3. **encrypt_key(Encryption and decryption key)**

   The encryption key in the case of encryption, here we use the AES symmetric encryption algorithm, that is, the same key is used for encryption and decryption.

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 3-21.

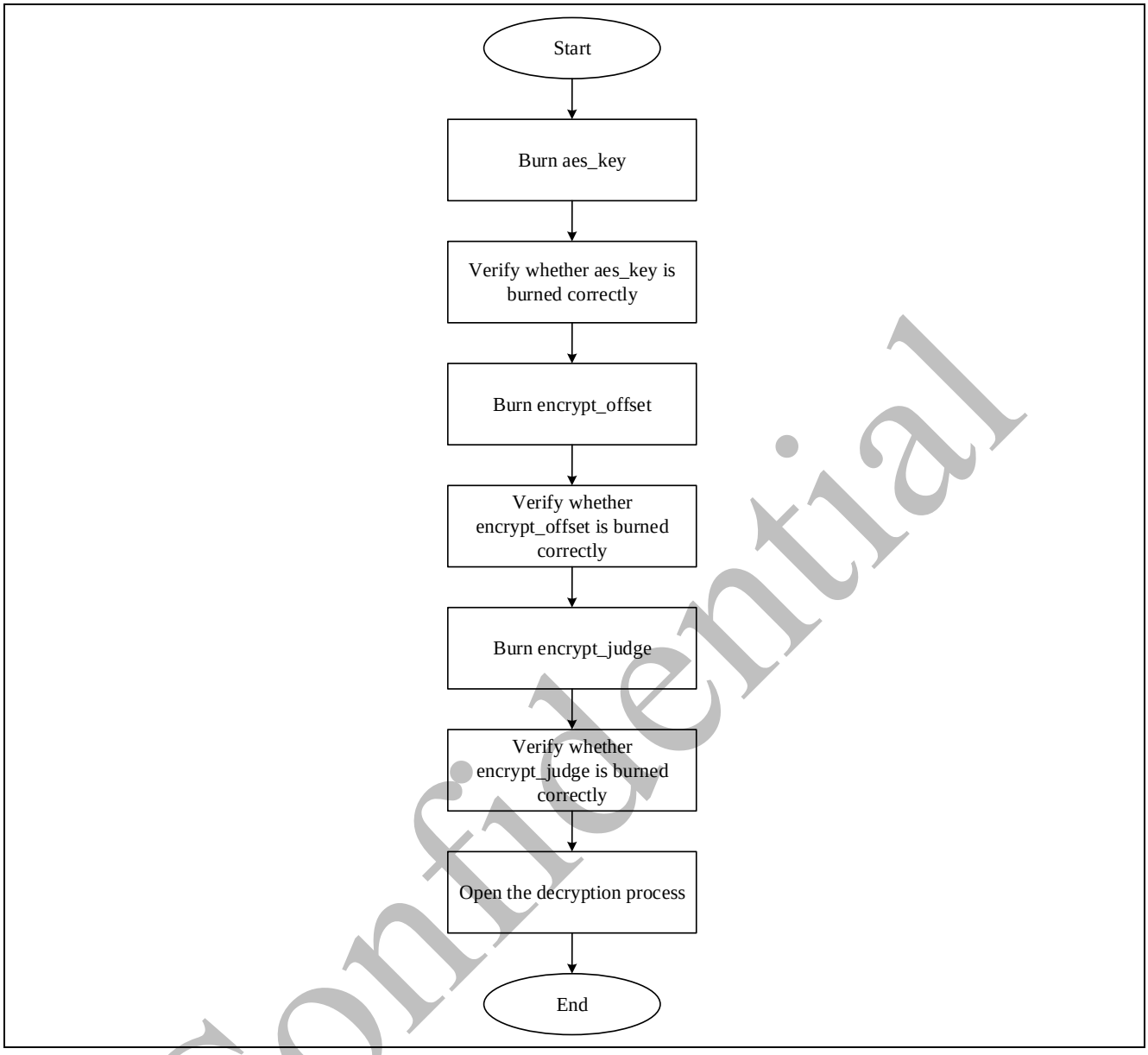


Figure 3-21 Efuse Encryption Area Burning Flowchart

Note:

- *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

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 Feature

- 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.

3.6.3 Block Diagram

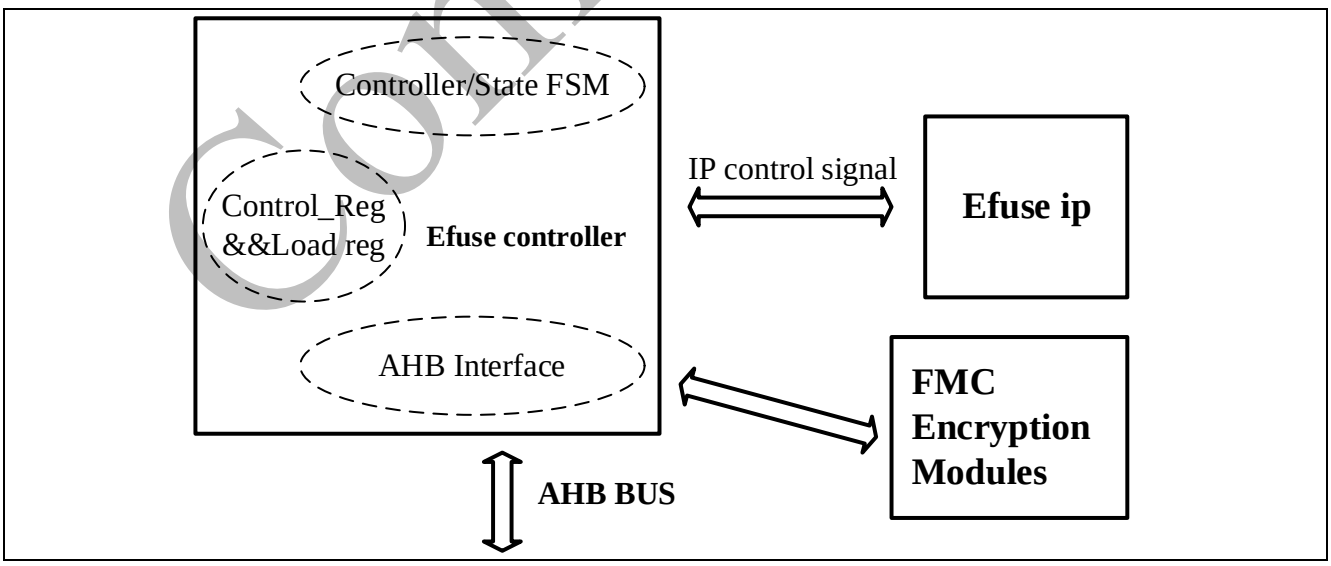


Figure 3-22 eFuse Controller Block Diagram

1. **Control state machine:**

   Mainly used to ensure the jump of the basic read and write state.

2. **Control register & initialization register:**

   The control register is mainly used to register the controller start condition (trigger condition), address and data, etc. The initialization register is used to store some system configuration information in the power-on load efuse. For details, see 2.3 Register Configuration Information.

3. **Control interface:**

   1) AHB bus interface
   2) Interaction with eFuse IP control signal
   3) AVDD & DVDD enable signal
   4) System interaction (register write protection, flash & ROM mode selection)
   5) Interaction with FMC encryption module

3.6.4 Functional Description

3.6.4.1 eFuse Introduction

eFuse can only be programmed once, but it can be read multiple times. eFUSE memory is burned by bit. In the initial state, all eFUSE bits are 0. These bits can be changed from 0 to 1 through a special programming sequence. Once a bit is programmed to 1, it cannot be modified anymore. (It can be understood that the hardware fuse is blown and cannot be recovered). The main function is to store user configuration information, which can be used as security protection.

eFuse Feature

- With a serial interface, 1bit can be written (write mode) and 8bit can be read (read mode) at each moment
- 1.2V typical voltage (DVDD)
- AVDD (voltage range [1.32V, 2.75V]) is not allowed to accumulate time greater than 1s
- AVDD voltage is not allowed to exceed 2.75V, otherwise there will be stability problems
- Temperature: -40℃~125℃

Fusing Requirements

- 2.5V typical fuse voltage (AVDD), AVDD must be pulled high in write mode, and must be pulled low or suspend in read mode/idle mode
- 4us fusing pulse width

3.6.5 eFuse Controller Register Map

Register	Offset	R/W	Description	Reset Value
EF_CTL Base Address: EF_BA = 0x4008_0000				
EFUSE_CTL	EF_BA+0x00	R/W	EF Control Register	0x0000_0000
EFUSE_ADDR	EF_BA+0x04	R/W	EF ADDR Register	0x0000_0000
EFUSE_DAT	EF_BA+0x08	R/W	EF DAT Register	0x0000_0000
EFUSE_VDD	EF_BA+0x10	R/W	EF DVDD Register	0x0000_0001
EFUSE_CMD	EF_BA+0x18	R/W	EF CMD Register	0x0000_0002
EFUSE_TRG	EF_BA+0x1C	R/W	EF TRG Register	0x0000_0000
EFUSE_PROG_TIMING1	EF_BA+0x20	R/W	EFUSE_PROG_TIMING1 Register	0x0010_4E06
EFUSE_PROG_TIMING2	EF_BA+0x24	R/W	EFUSE_PROG_TIMING2 Register	0x009C_0884
EFUSE_PROG_TIMING3	EF_BA+0x28	R/W	EFUSE_PROG_TIMING3 Register	0x0000_0C27
EFUSE_READ_TIMING4	EF_BA+0x2C	R/W	EFUSE_PROG_TIMING4 Register	0x0030_180F
EFUSE_READ_TIMING5	EF_BA+0x30	R/W	EFUSE_PROG_TIMING5 Register	0x003C_100C
EFUSE_OP_ERROR	EF_BA+0x64	R	Operation error flag	0x0000_0000

3.6.6 eFuse Controller Register Description

3.6.6.1 EFUSE_CTL

Register	Offset	R/W	Description	Reset Value
EFUSE_CTL	EF_BA+0x00	R/W	EF Control Register	0x0000_0000

Bits	Descriptions	
[31:19]	Reserved	Reserved.
[18]	Tsp_rd_ctl	Tsp_rd time sequence parameter setting (write protect) 0:Tsp_rd =9'd6 1:Tsp_rd =Tsp_rd _set
[17]	Tsp_pg_avdd_ctl	Tsp_pg_avdd time sequence parameter setting (write protect) 0:Tsp_pg_avdd = 9'd39 1:Tsp_pg_avdd = Tsp_pg_avdd_set
[16]	Tsp_pgm_ctl	Tsp_pgm_ctl time sequence parameter setting (write protect) 0:Tsp_pgm = 9'd4 1:Tsp_pgm = Tsp_pgm_set
[15]	Tpgm_ctl	Tpgm time sequence parameter setting (write protect) 0:Tpgm = 9'd154 1:Tpgm = Tpgm_set
[14]	Thp_pgm_ctl	Thp_pgm time sequence parameter setting (write protect) 0:Thp_pgm = 9'd4 1:Thp_pgm = Thp_pgm_set
[13]	Taen_pgm_ctl	Taen_pgm time sequence parameter setting (write protect) 0:Taen_pgm = 9'd39 1:Taen_pgm = Taen_pgm_set
[12]	Thp_pg_avdd_ctl	Thp_pg_avdd time sequence parameter setting (write protect) 0:Taen_pgm = 9'd39

		1:Taen_pgm = Taen_pgm_set
[11]	Thp_rd_ctl	Thp_rd time sequence parameter setting (write protect) 0:Taen_pgm = 9'd6 1:Taen_pgm = Taen_pgm_set
[10]	Tsr_dvdd_ctl	Tsr_dvdd time sequence parameter setting (write protect) 0: Tsr_dvdd= 9'd15 1: Tsr_dvdd= Tsr_dvdd_set
[9]	Tsr_rd_ctl	Tsr_dvdd time sequence parameter setting (write protect) 0: Tsr_rd= 9'd12 1: Tsr_rd= Tsr_rd_set
[8]	Trd_ctl	Trd time sequence parameter setting (write protect) 0: Trd_ctl= 9'd12 1: Trd_ctl= Trd_set
[7]	Thr_rd_ctl	Thr_rd time sequence parameter setting (write protect) 0: Thr_rd= 9'd12 1: Thr_rd= Thr_rd_set
[6]	Taen_rd_ctl	Taen_rd time sequence parameter setting (write protect) 0: Taen_rd= 9'd8 1: Taen_rd= Taen_rd_set
[5]	Thr_dvdd	Thr_dvdd time sequence parameter setting (write protect) 0: Thr_dvdd= 9'd15 1: Taen_rd= Taen_rd_set
[4]	Efusectl_bank4	A：0xC0 -0xFF (bits) zone operation enable (write protect) 1: Enabled， 0: Disabled
[3]	Efusectl_bank3	A：0x80 -0xBF (bits) zone operation enable (write protect) 1: Enabled， 0: Disabled
[2]	Efusectl_bank2	A：0x40 -0x7F (bits) zone operation enable (write protect) 1: Enabled， 0: Disabled
[1]	Efusectl_bank1	A：0x00 -0x3F (byte) zone operation enable (write protect) 1: Enabled， 0: Disabled
[0]	Efusect_en	efuse operation enable (write protect) 1: Enabled， 0: Disabled

3.6.6.2 EFUSE_ADDR

Register	Offset	R/W	Description	Reset Value
EFUSE_ADDR	EF_BA+0x04	R/W	EF ADDR Register	0x0000_0000

Bits	Descriptions	
[31:11]	Reserved	
[7:0]	efuseaddr_adr	Read &PROG ADDR (Unit: BYTE)

3.6.6.3 EFUSE_DAT

Register	Offset	R/W	Description	Reset Value
EFUSE_DAT	EF_BA+0x08	R/W	EF DAT Register	0x0000_0000

Bits	Descriptions	
[31:11]	Reserved	
[7:0]	efusedat_dat	(1) Put the byte data to be written into EFUSE_DAT[7:0] when writing (2) Put the read data into EFUSE_DAT[7:0].

3.6.6.4 EFUSE_VDD

Register	Offset	R/W	Description	Reset Value
EFUSE_VDD	EF_BA+0x10	R/W	EF DVDD Register	0x0000_0001

Bits	Descriptions	
[31:2]	Reserved	
[1]	AVDD_reg	AVDD_enable signal configuration bit 1: Wake up 0: Turn off
[0]	DVDD_reg	DVDD_enable signal configuration bit 1: Wake up 0: Turn off

3.6.6.5 EFUSE_CMD

Register	Offset	R/W	Description	Reset Value
EFUSE_CMD	EF_BA+0x18	R/W	EF CMD Register	0x0000_0002

Bits	Descriptions	
[31:2]	Reserved	
[0:1]	efusecmd_cmd	10: Inactive 01: Prog 00: Read

3.6.6.6 EFUSE_TRG

Register	Offset	R/W	Description	Reset Value
EFUSE_TRG	EF_BA+0x1C	R/W	EF TRG Register	0x0000_0000

Bits	Descriptions	
[31:1]	Reserved	
[1]	efusetrg_go	1: Trg flag, in operation (write protect) 0: No operation request or no operation completed

3.6.6.7 EFUSE_PROG_TIMING1

Register	Offset	R/W	Description	Reset Value
EFUSE_PROG_TIMING1	EF_BA+0x20	R/W	EFUSE_PROG_TIMING1 Register	0x0010_4E06

Bits	Descriptions	
[31:27]	Reserved	
[26:18]	Tsp_pgm_set	Default value: 9'd4, software customizable 6(48M) / 8(64M)
[17:9]	Tsp_pg_avdd_set	Default value: 9'd39, software customizable 58(48M) / 78(64M)
[8:0]	Tsp_rd_set	Default value: 9'd6, software customizable 9(48M) / 12(64M)

3.6.6.8 EFUSE_PROG_TIMING2

Register	Offset	R/W	Description	Reset Value
EFUSE_PROG_TIMING2	EF_BA+0x24	R/W	EFUSE_PROG_TIMING2 Register	0x009C_0884

Bits	Descriptions	
[31:27]	Reserved	
[26:18]	Taen_pgm_set	Default value: 9'd39, software customizable 60(48M) / 78(64M)
[17:9]	Thp_pgm_set	Default value: 9'd4, software customizable 6(48M) / 8(64M)
[8:0]	Tpgm_set	Default value: 9'd132, software customizable 198(48M) / 264(64M)

3.6.6.9 EFUSE_PROG_TIMING3

Register	Offset	R/W	Description	Reset Value
EFUSE_PROG_TIMING3	EF_BA+0x28	R/W	EFUSE_PROG_TIMING3 Register	0x0000_0C27

Bits	Descriptions	
[31:18]	Reserved	
[17:9]	Thp_rd_set	Default value: 9'd6, software customizable 9(48M)/12(64M)
[8:0]	Thp_pg_avdd_set	Default value: 9'd39, software customizable 60(48M)/78(64M)

3.6.6.10 EFUSE_READ_TIMING4

Register	Offset	R/W	Description	Reset Value
EFUSE_READ_TIMING4	EF_BA+0x2C	R/W	EFUSE_READ_TIMING4 Register	0x0030_180F

Bits	Descriptions	
[31:27]	Reserved	
[26:18]	Trd_set	Default value: 9'd12, software customizable 18(48M)/ 24(64M)
[17:9]	Tsr_rd_set	Default value: 9'd12, software customizable 18(48M)/ 24(64M)
[8:0]	Tsr_dvdd_set	Default value: 9'd15, software customizable 23(48M)/ 30(64M)

3.6.6.11 EFUSE_READ_TIMING5

Register	Offset	R/W	Description	Reset Value
EFUSE_READ_TIMING5	EF_BA+0x30	R/W	EFUSE_READ_TIMING5 Register	0x003C_100C

Bits	Descriptions	
[31:27]	Reserved	
[26:18]	Thr_dvdd_set	Default value: 9'd15, software customizable 23(48M)/ 30(64M)
[17:9]	Taen_rd_set	Default value: 9'd8, software customizable 12(48M)/ 16(64M)
[8:0]	Thr_rd_set	Default value: 9'd12, software customizable 18(48M)/ 24(64M)

3.6.6.12 EFUSE_OP_ERROR

Register	Offset	R/W	Description	Reset Value
EFUSE_OP_ERROR	EF_BA+0x64	R/W	Operation error flag	0x0000_0000

Bits	Descriptions	
[31:1]	reserve	
[0]	op_error	1: Indicates that this operation is a misoperation 0: normal operation The software detects that op_error is pulled high, write 1 and clear 0 to clear this flag

3.7 ECC Accelerator

3.7.1 Overview

ECC accelerator is mainly used to accelerate ECC encryption algorithm.

3.7.2 Features

- Support dot product calculation, 32bit division with or without sign. According to the actual 32bit multiplication/division calculation cycle, adjust the actual latch cycle (register's cycle can be configured)
- Software trigger operation/interrupt report function
- Clock is default as CPU clock

3.7.3 Block Diagram

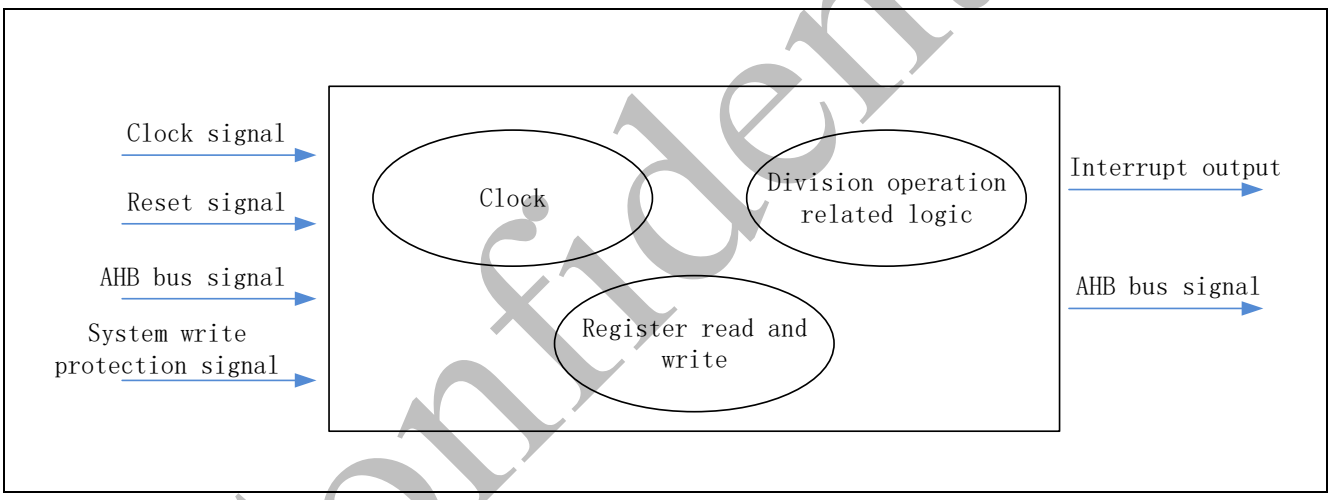


Figure 3-23 Block Diagram

3.7.4 Software Process

1. Dot product calculation operation process

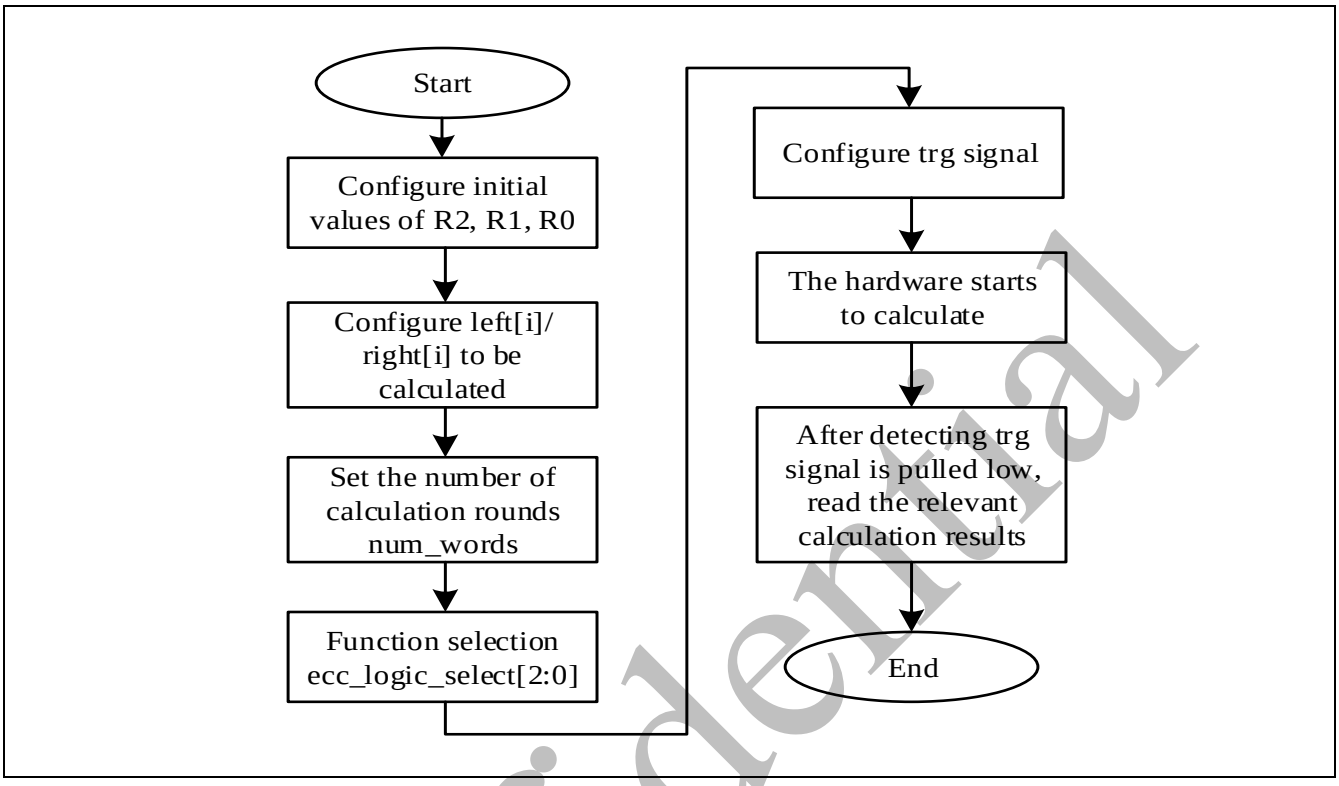


Figure 3-24 Software Operation Process

①Configure R2, R1, R0 ②Configure the required left[i]/right[i] ③Set the number of calculation rounds ④Function selection ⑤Configure trg signal ⑥Hardware calculation is completed, read the result out. Please refer to *Chapter 4.7.6* related register description for details.

2. Interrupt the reporting system

After the software configures the initial calculation parameters, configure the interrupt register bit[0] to enable the interrupt, and then configure the trg signal to start the hardware calculation. After the calculation is completed, the interrupt register flag signal bit[2] is pulled high, and the software can write 1 to clear 0. The actual system will detect the rising edge of bit[2] to trigger a pulse interrupt signal ecc_int_irq to return to the system. For details, see

*ecc*_irq register description.

3. 32-bit division operation register configuration flow

a. Open the write protection register

b. Configure the input register source_a/source_b

c. Configuration function selection mult_div_select (hardware calculation reserved time, symbol calculation selection, division enable selection)

d. Configure the acc_trg register trg signal to start the calculation

e. Detect the trg signal whether pulled low to read the result register result_l

Note: Refer to Chapter 4.7.6 or a detailed description of registers

3.7.5 ECC Accelerator Register Map

Register	offset	R/W	Description	Reset Value
ECC base addr:0x4009_0000				
ecc_r0	ECC_BA+0x00	R/W	R0 register	0x0000_0000
ecc_r1	ECC_BA+0x04	R/W	R1 register	0x0000_0000
ecc_r2	ECC_BA+0x08	R/W	R2 register	0x0000_0000
ecc_trg	ECC_BA+0x0C	R/W	Trg signal register for ECC operation	0x0000_0000
ecc_irq	ECC_BA+0x10	R/W	Interrupt flag register	0x0000_0000
acc_trg	ECC_BA+0x14	R/W	Trg signal register for division operations	0x0000_0000
ecc_logic_select	ECC_BA+0x18	R/W	ECC logic function selector register	0x0000_0500
num_words	ECC_BA+0x1C	R/W	Counting round register	0x0000_0008
source_a	ECC_BA+0x20	R/W	Dividend enters 32’b register	0x0000_0000
source_b	ECC_BA+0x24	R/W	Divisor input 32’b register	0x0000_0000
reg_left_x	ECC_BA+(0x28-44)	R/W	Left[i](I=0~7)	0x0000_0000
reg_right_x	ECC_BA+(0x48-64)	R/W	Right[i] (I=0~7)	0x0000_0000
ecc_result[i]	ECC_BA+(0x68-a4)	R	Result[i](I=0~15)	0x0000_0000
mult_div_select	ECC_BA+0xa8	R/W	Division function selector	0x0000_00fc
result_l	ECC_BA+0xac	R	Divisor output quotient 32’b register	0x0000_0000

3.7.6 ECC Accelerator Register Description

3.7.6.1 ecc_r0

Register	Offset	R/W	Description	Reset Value
ecc_r0	ECC_BA+0x00	R/W	R0 register	0x0000_0000

Bits	Descriptions
[31: 0]	R0[31:0]

3.7.6.2 ecc_r1

Register	Offset	R/W	Description	Reset Value
ecc_r1	ECC_BA+0x04	R/W	R1 register	0x0000_0000

Bits	Descriptions
[31:0]	R1[31:0]

3.7.6.3 ecc_r2

Register	Offset	R/W	Description	Reset Value
ecc_r2	ECC_BA+0x08	R/W	R2 register	0x0000_0000

Bits	Descriptions
[31: 0]	R2[31:0]

3.7.6.4 ecc_trg

Register	Offset	R/W	Description	Reset Value
ecc_trg	ECC_BA+0x0C	R/W	Trg signal register for division operations	0x0000_0000

Bits	Descriptions
[31: 0]	trg signal 1: in operation 0: Operation completed Note: These bits are all Write-protected

3.7.6.5 ecc_irq

Register	Offset	R/W	Description	Reset Value
ecc_irq	ECC_BA+0x10	R/W	Interrupt flag register	0x0000_0000

Bits	Descriptions
[31: 2]	reseverd
[2]	ecc_if interrupt flag when hardware is completed (bit[0]=1'b1), software write 1 to clear 0 (write protection)
[1]	ecc_f original completion signal (bit[0]=1'b0), software write 1 to clear 0 (write protection)
[0]	ecc_irq_en enable polling mode, query interruption (write protection)

3.7.6.6 acc_trg

Register	Offset	R/W	Description	Reset Value
acc_trg	ECC_BA+0x14	R/W	Trg signal register for division operations	0x0000_0000

Bits	Descriptions
[31: 1]	reseverd
[0]	(Write protection) trg signal 1: in operation 0: Operation completed

3.7.6.7 ecc_logic_select

Register	Offset	R/W	Description	Reset Value
ecc_logic_select	ECC_BA+0x18	R/W	ECC logic function selector register	0x0000_0500

Bits	Descriptions
[31: 11]	reseverd
[8-10]	ecc_mult_calc 32-bit multiplier reserved calculation period (default is 5clk, write protection)
[1-7]	reseverd
[0]	ecc_mult_en dot product function (write protection)

3.7.6.8 num_words

Register	Offset	R/W	Description	Reset Value
num_words	ECC_BA+0x1c	R/W	Counting round register	0x0000_0008

Bits	Descriptions
[31: 3]	reseverd
[3:0]	Num_words (num_words in the calculation rounds software)

3.7.6.9 source_a

Register	Offset	R/W	Description	Reset Value
source_a	ECC_BA+0x20	R/W	Dividend enters 32'b register	0x0000_0000

Bits	Descriptions
[31:0]	32-bit dividend input

3.7.6.10 source_b

Register	Offset	R/W	Description	Reset Value
source_b	ECC_BA+0x24	R/W	Divisor input 32'b register	0x0000_0001

Bits	Descriptions
[31:0]	32-bit divisor input, if the divisor is set to 0 by mistake, From the perspective of EDA simulation, the result will be uncertain (the simulation result is 32'h xxxx_xxxx uncertain state), which may affect the program operation

3.7.6.11 reg_left_x

Register	Offset	R/W	Description	Reset Value
reg_left_x	ECC_BA+(0x28-0x44)	R/W	Left[0]~Left[7]	0x0000_0000

Bits	Descriptions
[31: 0]	Data to be calculated Left[i]

3.7.6.12 reg_right_x

Register	Offset	R/W	Description	Reset Value
reg_right_x	ECC_BA+(0x48-64)	R/W	Right[0]~Right[7]	0x0000_0000

Bits	Descriptions
[31: 0]	Data to be calculated right[i]

3.7.6.13 result[i]

Register	Offset	R/W	Description	Reset Value
result[i]	ECC_BA+(0x68-a4)	R	Result[i] (0-15)	0x0000_0000

Bits	Descriptions
[31: 0]	Calculation result result[i] The carry and borrow returned by addition calculation and subtraction calculation are placed in bit[0] of result[15]

3.7.6.14 mult_div_select

Register	Offset	R/W	Description	Reset Value
mult_div_select	ECC_BA+a8	R	Division function selector	0x0000_00fc

Bits	Descriptions
[31: 8]	Reseverd
[7: 2]	Multiplication and division hardware calculation time can be configured。
[1]	Signed calculation option 1: signed 0: unsigned
[0]	Reseverd

3.7.6.15 result_l

Register	Offset	R/W	Description	Reset Value
Result_l	ECC_BA+0xac	R	32’b divisor output quotient register	0x0000_0000

Bits	Descriptions
[31: 0]	Divisor output quotient 32’b

3.8 DMA Serial Interface Controller (DMA)

3.8.1 Overview

The DMA in PAN108 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.

3.8.2 Features

- Up to 3 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

3.8.3 Block Diagram

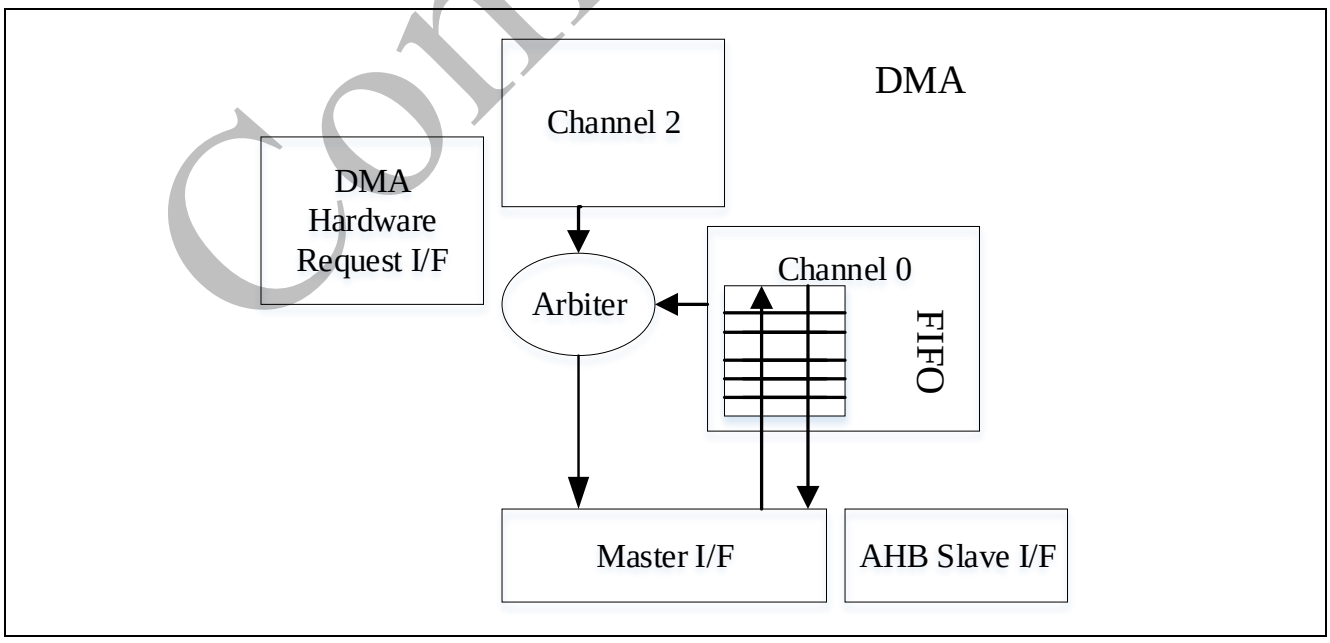


Figure 3-25 DMA Controller Block Diagram

3.8.4 Functional Description

3.8.4.1 AHB master interface

The DMA contains one full AHB masters. Figure 3-26 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. PAN108 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.

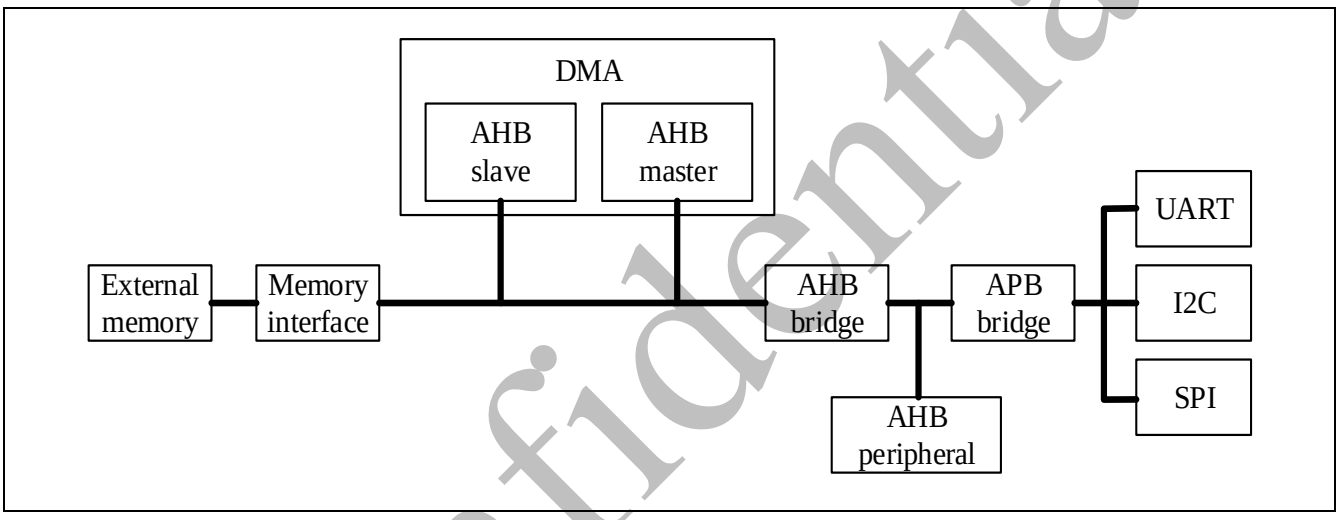


Figure 3-26 DMA Transfer Hierarchy

3.8.4.2 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 3-27 illustrates the hardware handshaking interface between a peripheral – whether a destination or source – and the DMA when the DMA is the flow controller.

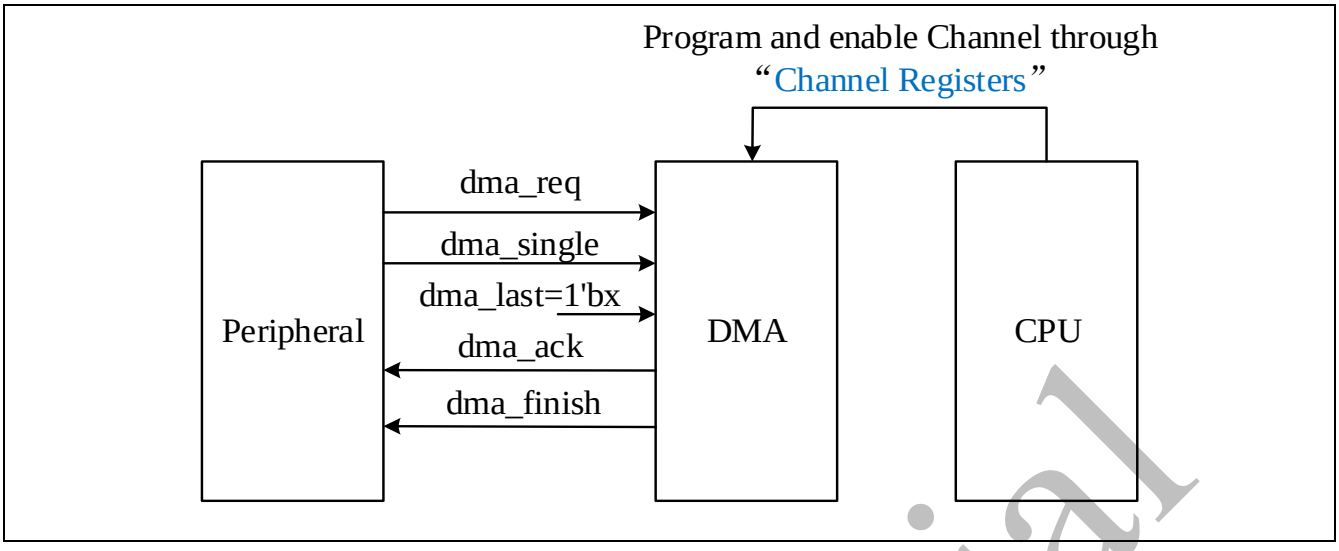


Figure 3-27 Hardware Handshaking Interface

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 3-28.

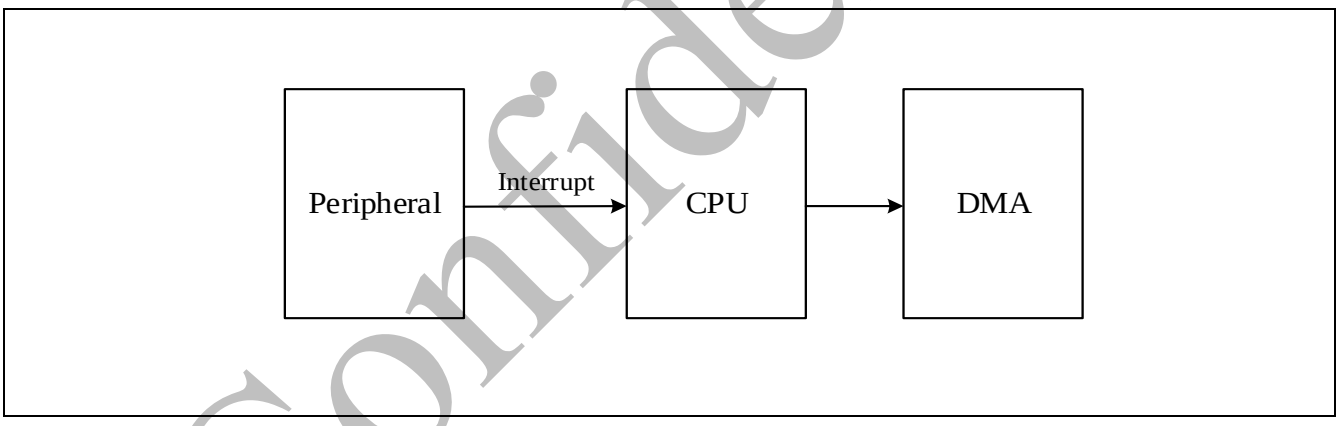


Figure 3-28 Software Controlled DMA Transfers

On completion of the transaction – single or burst – the relevant channel bit in the *ReqSrcReg/ReqDstReg* register is cleared by hardware. Software can therefore poll this bit in order to determine when the requested transaction has completed. Alternatively, the IntSrcTran or IntDstTran 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.

3.8.4.3 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 PAN108 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.TT_FC.

Table 3-14 lists the decoding for CTLx.TT_FC field.

Table 3-14 CTLx.TT_FC Field Decoding

CTLx.TT_FC 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 3-29 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.

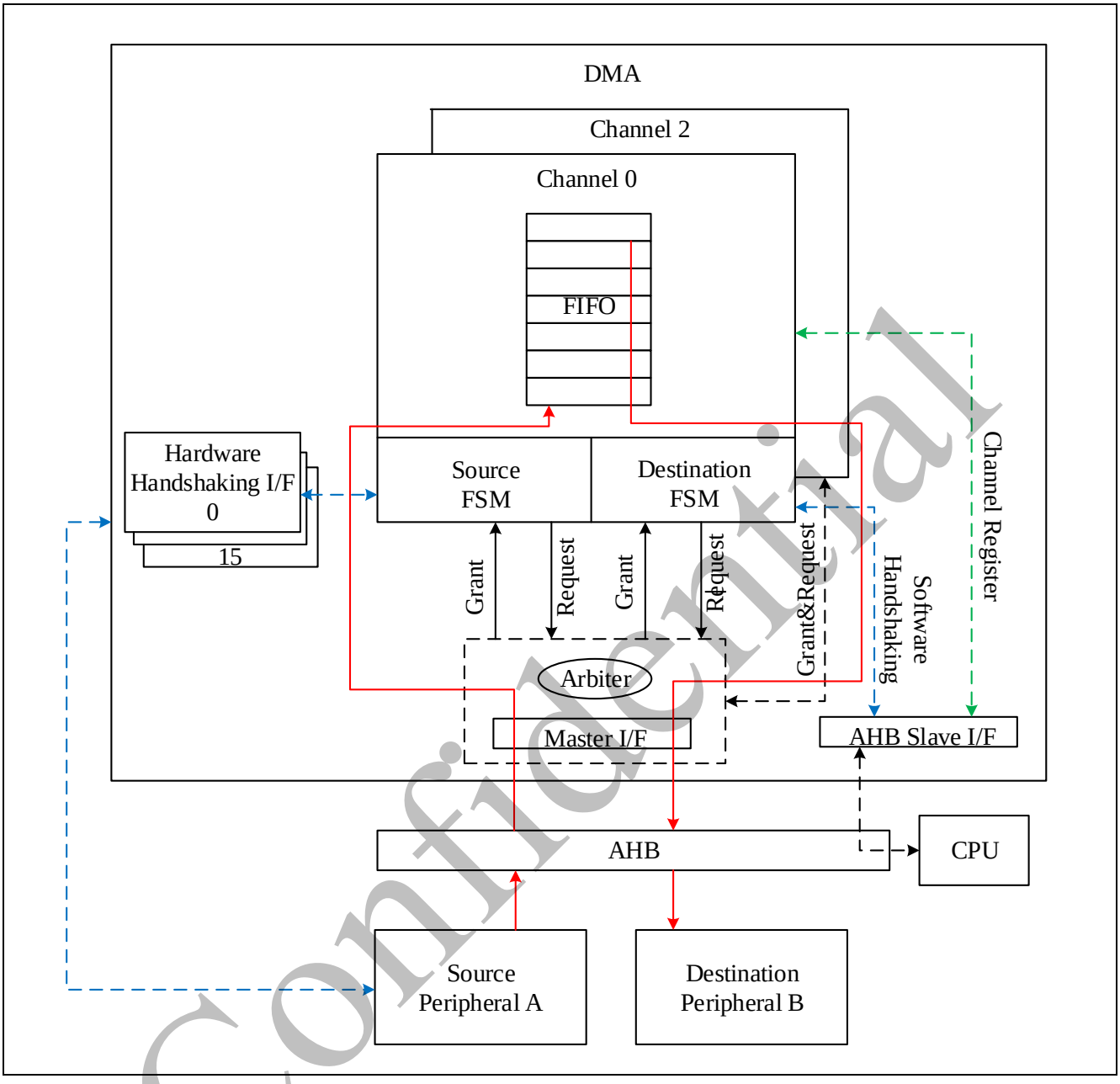


Figure 3-29 Peripheral-to-Peripheral DMA Transfer

3.8.4.4 Basic Interface Definitions

The following definitions are used in this chapter:

Source single transaction size in bytes

- src_single_size_bytes = CTLx.SRC_TR_WIDTH/8 (1)

Source burst transaction size in bytes

- src_burst_size_bytes = CTLx.SRC_MSIZE * src_single_size_bytes (2)

Destination single transaction size in bytes

- dst_single_size_bytes = CTLx.DST_TR_WIDTH/8 (3)

Destination burst transaction size in bytes

- dst_burst_size_bytes = CTLx.DEST_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.SRC_TR_WIDTH) to be transferred by the DMA in a block transfer; this is programmed into the CTLx.BLOCK_TS field. Therefore, the total number of bytes to be transferred in a block is:
- blk_size_bytes_DMA = CTLx.BLOCK_TS * src_single_size_bytes (5)

3.8.4.5 Single-block Transfer

The PAN108 series only supports single-block transfer. As shown in Figure 3-30, 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.

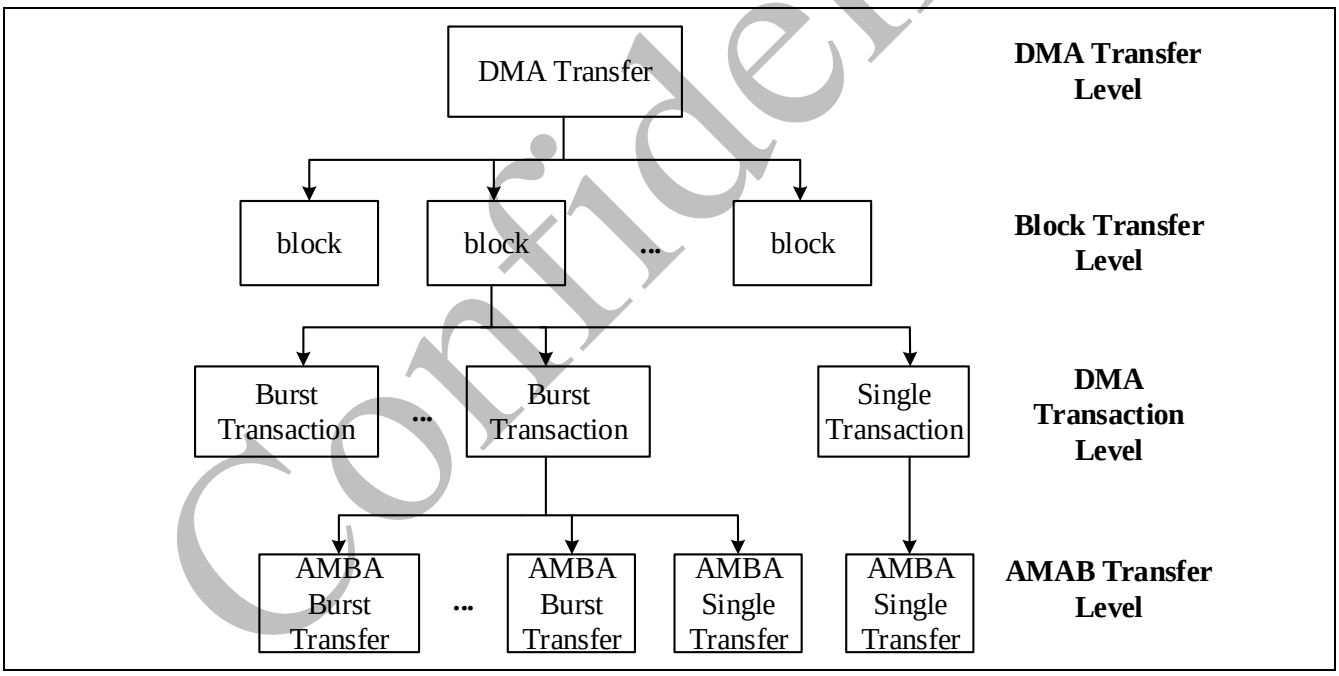


Figure 3-30 DMAC Transfer Hierarchy

The programing processes are shown as follow:

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

2 Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: *ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, and ClearErr*. 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 SARx register for channel x
2) Write the starting destination address in the DARx register for channel x
3) Program CTLx and 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 TT_FC of the CTLx register.
   b. Set up the transfer characteristics, such as:
      - Transfer width for the source in the SRC_TR_WIDTH field.
      - Transfer width for the destination in the DST_TR_WIDTH field.
      - Incrementing/decrementing or fixed address for the source in the SINC field.
      - Incrementing/decrementing or fixed address for the destination in the DINC field.
5) Write the channel configuration information into the *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 HS_SEL_SRC/HS_SEL_DST 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 SRC_PER and DEST_PER bits, respectively.

4 Ensure that bit 0 of the *DmaCfgReg* register is enabled before writing to *ChEnReg*.

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 (RawTfr[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, ClearTfr[n], before the channel is enabled.

The flow diagram in Figure 3-31 shows an overview of programming the DMAC.

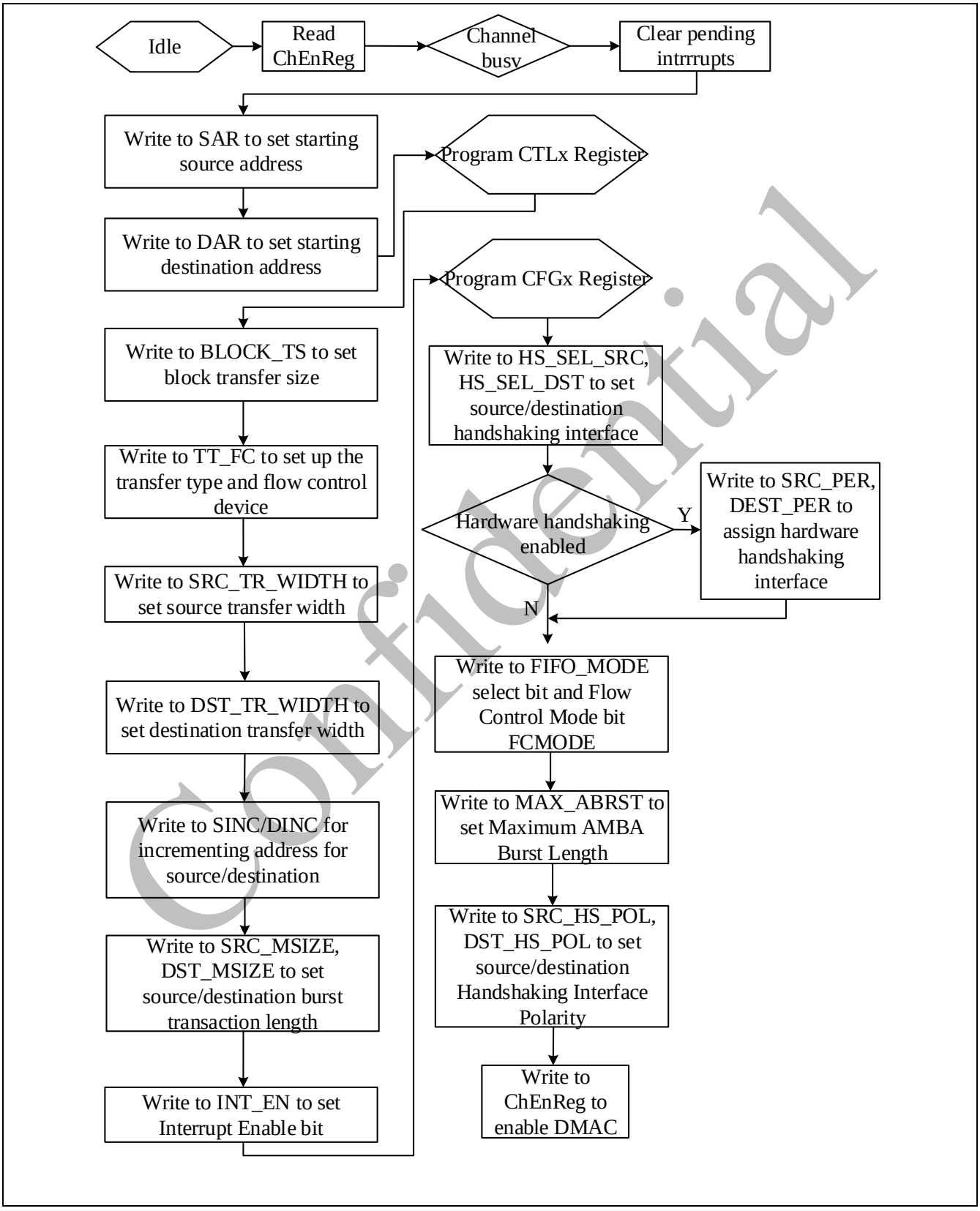


Figure 3-31 Flowchart for single-block DMAC Programming

3.8.4.6 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.*SRC_MSIZE*
- Destination transaction length, CTLx.*DEST_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_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.SRC_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.

3.8.4.7 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 FIFO_MODE field of the CFGx register.

The definition of FIFO readiness is the same for:

- Single transactions
- Burst transactions, where FIFO_MODE = 0
- Transfers involving memory peripherals, where FIFO_MODE = 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 SRC_TR_WIDTH width. FIFO readiness for destination transfers occurs when the channel FIFO contains data to form at least a single transfer of DST_TR_WIDTH width.

When FIFO_MODE = 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 FIFO_MODE = 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 FIFO_MODE 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.

3.8.4.8 Interrupt

There are five interrupt registers in the DMA:

- IntBlock – Block Transfer Complete Interrupt
  This interrupt is generated on DMA block transfer completion to the destination peripheral.
- IntDstTran – 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.
- IntErr – 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.
- IntSrcTran – 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.
- IntTfr – 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.

3.8.4.9 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 3-32 illustrates the arbitration flow of the master bus interface.

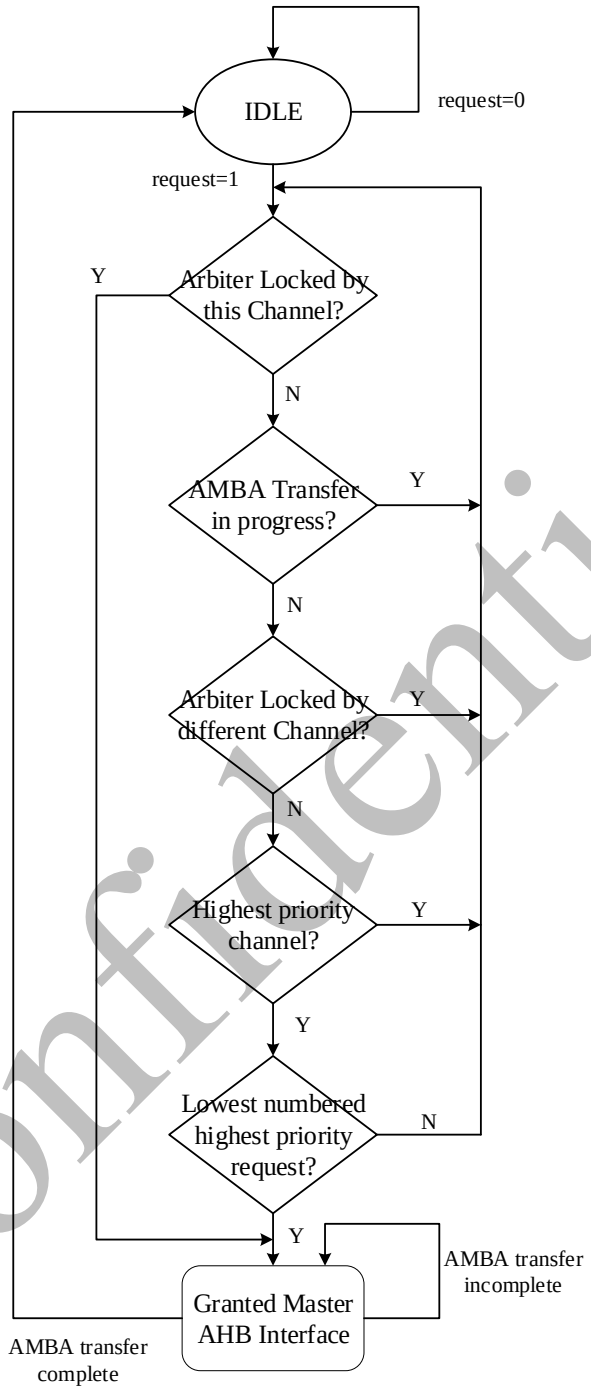


Figure 3-32 Arbitration Flow for Master Bus Interface

An arbitration scheme is used to decide which of the request lines (2*DMAH_NUM_CHANNELS) 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 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.

3.8.4.10 IP DMA Control

Table 3-15 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

3.8.5 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				
SAR0	DMA_BA+0x000	R/W	Source Address Register for Channel 0	0x0
DAR0	DMA_BA+0x008	R/W	Destination Address Register for Channel 0	0x0
CTL0	DMA_BA+0x018	R/W	Control Register for Channel 0	Configuration dependent
CFG0	DMA_BA+ 0x040	R/W	Configuration Register for Channel 0	0x0000_0004_0000_0E00
SAR1	DMA_BA+0x058	R/W	Source Address Register for Channel 1	0x0
DAR1	DMA_BA+0x060	R/W	Destination Address Register for Channel 1	0x0
CTL1	DMA_BA+0x070	R/W	Control Register for Channel 1	Configuration dependent
CFG1	DMA_BA+ 0x098	R/W	Configuration Register for Channel 1	0x0000_0004_0000_0E20

SAR2	DMA_BA+0x0B0	R/W	Source Address Register for Channel 2	0x0
DAR2	DMA_BA+0x0B8	R/W	Destination Address Register for Channel 2	0x0
CTL2	DMA_BA+0x0C8	R/W	Control Register for Channel 2	Configuration dependent
CFG2	DMA_BA+ 0x0F0	R/W	Configuration Register for Channel 2	0x0000_0004_ 0000_0E40
RawTfr	DMA_BA+0x2C0	R/W	Interrupt Raw Status Registers	0x0
RawBlock	DMA_BA+0x2C8	R/W	Interrupt Raw Status Registers	0x0
RawSrcTran	DMA_BA+ 0x2D0	R/W	Interrupt Raw Status Registers	0x0
RawDstTran	DMA_BA+0x2D8	R/W	Interrupt Raw Status Registers	0x0
RawErr	DMA_BA+0x2E0	R/W	Interrupt Raw Status Registers	0x0
StatusTfr	DMA_BA+0x2E8	R	Interrupt Status Registers	0x0
StatusBlock	DMA_BA+0x2F0	R	Interrupt Status Registers	0x0
StatusSrcTran	DMA_BA+0x2F8	R	Interrupt Status Registers	0x0
StatusDstTran	DMA_BA+0x300	R	Interrupt Status Registers	0x0
StatusErr	DMA_BA+0x308	R	Interrupt Status Registers	0x0
MaskTfr	DMA_BA+0x310	R/W	Interrupt Mask Registers	0x0
MaskBlock	DMA_BA+0x318	R/W	Interrupt Mask Registers	0x0
MaskSrcTran	DMA_BA+0x320	R/W	Interrupt Mask Registers	0x0
MaskDstTran	DMA_BA+0x328	R/W	Interrupt Mask Registers	0x0
MaskErr	DMA_BA+0x330	R/W	Interrupt Mask Registers	0x0
ClearTfr	DMA_BA+0x338	W	Interrupt Clear Registers	0x0
ClearBlock	DMA_BA+0x340	W	Interrupt Clear Registers	0x0
ClearSrcTran	DMA_BA+0x348	W	Interrupt Clear Registers	0x0
ClearDstTran	DMA_BA+0x350	W	Interrupt Clear Registers	0x0
ClearErr	DMA_BA+0x358	W	Interrupt Clear Registers	0x0
Statusint	DMA_BA+0x360	R	Combined Interrupt Status Register	0x0
ReqSrcReg	DMA_BA+0x368	R/W	Source Software Transaction Request Register	0x0
ReqDstReg	DMA_BA+0x370	R/W	DestinationSoftware Transaction Request Register	0x0
SglReqSrcReg	DMA_BA+0x378	R/W	Destination Software Transaction Request Register	0x0
SglReqDstReg	DMA_BA+0x380	R/W	Destination Software Transaction Request Register	0x0
LstSrcReg	DMA_BA+0x388	R/W	Last Source Transaction Request Register	0x0
LstDstReg	DMA_BA+0x390	R/W	Last Destination Transaction Request Register	0x0
DmaCfgReg	DMA_BA+0x398	R	DMA Configuration Register	0x0
ChEnReg	DMA_BA+0x3A0	R/W	DMAC Channel Enable Register	0x0
DmaIdReg	DMA_BA+0x3A8	R/W	DMA ID Register	0x0
DMA_COMP_ PARAMS_3	DMA_BA+0x3E0	R	DMAC Component Parameters Register 3	Depend on user configuration
DMA_COMP_ PARAMS_2	DMA_BA+0x3E8	R	DMAC Component Parameters Register 2	Depend on user configuration
DMA_COMP_ PARAMS_1	DMA_BA+0x3F0	R	DMAC Component Parameters Register 1	Depend on user configuration
DCIR	DMA_BA+0x3F8	R	DMA Component ID Register	0x4457_1110

3.8.6 DMA Register Description

3.8.6.1 Configuration and Channel Enable Register

DMA Configuration Register (DmaCfgReg)

Register	Offset	R/W	Description	Reset Value
DmaCfgReg	DMA_BA+0x398	R	DMA Configuration Register	0x0

Bits	Description	
[63:1]	Reserved	Reserved.
[0]	DMA_EN	DMA Enable bit. 0 = DMA Disabled 1 = DMA Enabled.

DMA Channel Enable Register (ChEnReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10 :8]	CH_EN_WE	Channel enable write enable(only writable)
[7:3]	Reserved	Reserved
[2:0]	CH_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 ChEnReg. CH_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, DmaCfgReg[0], is 0.

3.8.6.2 Channel Register

Source Address Register for Channel x Register (SARx)

Register	Offset	R/W	Description	Reset Value
Source Address Register for Channel x x=[0,1,2] SAR0 – 0x000 SAR1 – 0x058 SAR2 – 0x0B0	DMA_BA+SARx	R/W	Source Address Register for Channel x	0x0

Bits	Description	
[63:32]	Reserved	Reserved.
[31:0]	SAR	Current Source Address of DMA transfer. Updated after each source transfer. The SINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every source transfer throughout the block transfer.

Destination Address Register for Channel x(DARx)

Register	Offset	R/W	Description	Reset Value
DAR*x* x=[0,1,2] DAR0 – 0x008 DAR1 – 0x060 DAR2 – 0x0B8	DMA_BA+ DARx	R/W	Destination Address Register for Channel x	0x0

Bits	Description	
[63:32]	Reserved	Reserved.
[31:0]	DAR	Current Destination address of DMA transfer. Updated after each destination transfer. The DINC field in the CTLx register determines whether the address increments, decrements, or is left unchanged on every destination transfer throughout the block transfer.

Control Register for Channel x(CTLx)

Register	Offset	R/W	Description	Reset Value
CTL*x* x=[0,1,2] CTL0–0x018 CTL1– 0x070 CTL2– 0x0C8	DMA_BA+CTLx	R/W	Control Register for Channel x	0x0000_0000_0206_1201

Bits	Description	
[63:45]	Reserved	Reserved.
[44]	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
[43:42]	Reserved	Reserved.
[41:32]	BLOCK_TS	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 BLOCK_TS 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 SRC_TR_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
[31:23]	Reserved	Reserved
[22:20]	TT_FC	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.

CTLx.TT_FC Field	Transfer Type	Flow Controller
000	Memory to Memory	DMA
001	Memory to Peripheral	DMA
010	Peripheral to Memory	DMA
011	Peripheral to Peripheral	DMA

		100 Peripheral to Memory Peripheral 101 Peripheral to Peripheral Source Peripheral 110 Memory to Peripheral Peripheral 111 Peripheral to Peripheral Destination Peripheral Reset value: 3'h011
[19:17]	Reserved	Reserved
[16:14]	SRC_MSIZE	Source Burst Transaction Length. Number of data items, each of width SRC_TR_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
[13:11]	DEST_MSIZE	Destination Burst Transaction Length. Number of data items, each of width DST_TR_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. CTLx.SRC_MSIZE / CTLx.DEST_MSIZE: Number of data items to be transferred (of width CTLx.SRC_TR_WIDTH or CTLx.DST_TR_WIDTH) 000 1 001 4 010 8 011 16 100 32 101 64 110 128 111 256 Reset value: 3'b001
[10:9]	SINC	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]	DINC	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]	SRC_TR_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]	DST_TR_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.SRC_TR_WIDTH / CTLx.DST_TR_WIDTH: Size (bits) 000: 8 001: 16 010: 32 011: 64 100: 128 101: 256 11X: 256 Reset value: 3'b000
[0]	INT_EN	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_EN = 0. Reset value: 1'b1

Configuration Register for Channel x(CFGx)

Register	Offset	R/W	Description	Reset Value
CFG0	DMA_BA+ 0x040	R/W	Configuration Register for Channel 0	0x0000_0004_0000_0E00
CFG1	DMA_BA+ 0x098	R/W	Configuration Register for Channel 1	0x0000_0004_0000_0E20
CFG2	DMA_BA+ 0x0f0	R/W	Configuration Register for Channel 2	0x0000_0004_0000_0E40

Bits	Description	
[63:47]	Reserved	Reserved.
[46:43]	DEST_PER	Assigns a hardware handshaking interface (0 - 15) to the destination of channel x if the CFGx.HS_SEL_DST 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]	SRC_PER	Assigns a hardware handshaking interface (0 - 15) to the source of channel x if the CFGx.HS_SEL_SRC 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]	PROTCTL	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] CFGx.PROTCTL[1] → HPROT[1] CFGx.PROTCTL[2]-> → HPROT[2] CFGx.PROTCTL[3]-> → HPROT[3]
[33]	FIFO_MODE	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]	FCMODE	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]	RELOAD_DST	Automatic Destination Reload. The DARx 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]	SRC_HS_POL	Source Handshaking Interface Polarity. 0 = Active high 1 = Active low
[18]	DST_HS_POL	Destination Handshaking Interface Polarity. 0 = Active high 1 = Active low
[17:12]	Reserved	Reserved
[11]	HS_SEL_SRC	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]	HS_SEL_DST	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]	FIFO_EMPTY	Indicates if there is data left in the channel FIFO. Can be used inconjunction with CFGx.CH_SUSP to cleanly disable a channel. For more information. 1 = Channel FIFO empty 0 = Channel FIFO not empty
[8]	CH_SUSP	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 CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data. 0 = Not suspended. 1 = Suspend DMAC transfer from the source
[7:5]	CH_PRIOR	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

3.8.6.3 Interrupt Registers

Interrupt Raw Status Registers (IRSR)

Each bit in these registers is cleared by writing a 1 to the corresponding location in the ClearTfr, ClearBlock, ClearSrcTran, ClearDstTran, ClearErr registers.

Register	Offset	R/W	Description	Reset Value
RawBlock, RawDstTran, RawErr, RawSrcTran, RawTfr RawTfr_offset – 0x2C0 RawBlock_ offset– 0x2C8 RawSrcTran– 0x2D0 RawDstTran_ offset– 0x2D8 RawErr_ offset – 0x2E0	DMA_BA+x_offset	R/W	Interrupt Raw Status Registers	0x0

Bits	Description	
[63:3]	Reserved	Reserved.
[2:0]	RAW	Raw interrupt status.

Interrupt Status Registers(ISR)

All interrupt events from all channels are stored in these Interrupt Status registers after masking: StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, and StatusTfr.

Register	Offset	R/W	Description	Reset Value
StatusBlock, StatusDstTran, StatusErr, StatusSrcTran, StatusTfr StatusTfr_ offset – 0x2E8 StatusBlock_ offset – 0x2F0 StatusSrcTran_offset – 0x2F8 StatusDstTran_offset– 0x300 StatusErr_ offset – 0x308	DMA_BA+x_offset	R	Interrupt Status Registers	0x0

Bits	Description	
[63:3]	Reserved	Reserved.
[2:0]	STATUS	Interrupt status.

Interrupt Mask Registers (IMR)

Register	Offset	R/W	Description	Reset Value
MaskBlock, MaskDstTran, MaskErr, MaskSrcTran, MaskTfr MaskTfr_offset – 0x310 MaskBlock_offset – 0x318 MaskSrcTran_offset – 0x320 MaskDstTran_offset – 0x328 MaskErr_offset – 0x330	DMA_BA+x_offset	R/W	Interrupt Mask Registers	0x0

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	INT_MASK_WE	Interrupt Mask Write Enable 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	INT_MASK	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.

Interrupt Clear Registers (ICR)

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, and ClearTfr.

Register	Offset	R/W	Description	Reset Value
ClearBlock, ClearDstTran, ClearErr, ClearSrcTran, ClearTfr ClearTfr – 0x338 ClearBlock – 0x340 ClearSrcTran – 0x348 ClearDstTran – 0x350 ClearErr – 0x358	DMA_BA+x_offset	W	Interrupt Clear Registers	0x0

Bits	Description	
[63:3]	Reserved	Reserved.
[2:0]	CLEAR	Interrupt clear. 0 = no effect 1 = clear interrupt

Combined Interrupt Status Register (Statusint)

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

Bits	Description	
[63:5]	Reserved	Reserved.
[4]	ERR	OR of the contents of StatusErr register
[3]	DSTT	OR of the contents of StatusDst register.
[2]	SRCT	OR of the contents of StatusSrcTran register.
[1]	BLOCK	OR of the contents of StatusBlock register.
[0]	TFR	OR of the contents of StatusTfr register.

3.8.6.4 Software Handshaking Registers

Source Software Transaction Request Register (ReqSrcReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	SRC_REQ_WE	Source request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	SRC_REQ	Source request **Note**: A channel SRC_REQ bit is written only if the corresponding channel write enable bit in the SRC_REQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the ChEnReg register.

Destination Software Transaction Request Register (ReqDstReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	DST_REQ_WE	Destination request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	DST_REQ	Destination request **Note**: A channel DTS_REQ bit is written only if the corresponding channel write enable bit in the DTS_REQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the ChEnReg register.

Single Source Transaction Request Register (SglReqSrcReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	SRC_SGLREQ_WE	Single write enable(only writable) 0 = write disabled 1 = write enabled

[7:3]	Reserved	Reserved
[2:0]	SRC_SGLREQ	Source single request **Note**: A channel SRC_SGLREQ bit is written only if the corresponding channel write enable bit in the SRC_SGLREQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the ChEnReg register.

Single Destination Transaction Request Register (SglReqDstReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	DST_SGLREQ_WE	Destination write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	DST_SGLREQ	Destination single or burst request **Note**: A channel DTS_SGLREQ bit is written only if the corresponding channel write enable bit in the DTS_SGLREQ_WE field is asserted on the same AHB write transfer, and if the channel is enabled in the ChEnReg register.

Last Source Transaction Request Register(LstSrcReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	LSTSRC_WE	Source last transaction request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	LSTSRC	Source last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

Last Destination Transaction Request Register(LstDstReg)

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

Bits	Description	
[63:11]	Reserved	Reserved.
[10:8]	LSTDST_WE	Destination last transaction request write enable(only writable) 0 = write disabled 1 = write enabled
[7:3]	Reserved	Reserved
[2:0]	LSTDST	Destination last transaction request 0 = Not last transaction in current block 1 = Last transaction in current block

3.8.6.5 Miscellaneous DMA Registers

DMA ID Register (DmaIdReg)

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

Bits	Description	
[63:32]	Reserved	Reserved.
[31:0]	DMA_ID	Hardcoded DMA Peripheral ID

DMA Test Register (DmaTestReg)

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

Bits	Description	
[63:1]	Reserved	Reserved.
[0]	TEST_SLV_IF	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

DMA Component Parameters Register 3(DMA_COMP_PARAMS_3)

Register	Offset	R/W	Description	Reset Value
DMA_COMP_PARAMS_3	DMA_BA+0x3E0	R	DMA Component Parameters Register 3	0x0

Bits	Description	
[63]	Reserved	Reserved.
[62:60]	CH1_FIFO_DEPTH	The value of this register is derived from the DMAH_CH1_FIFO_DEPTH coreConsultant parameter. 0x0 = 8 0x1 = 16 0x2 = 32 2x3 = 64 0x4 = 128 Default value: 0x1
[59:51]	Reserved	Reserved.
[50:48]	CH1_MAX_MULT_SIZE	The value of this register is derived from the DMAH_CH1_MULT_SIZE coreConsultant parameter. 0x0 = 4 0x1 = 8 0x2 = 16

		0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x1
[47:46]	CH1_FC	The value of this register is derived from the DMAH_CH1_FC coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY Default value: 0x0
[45]	Reserved	Reserved
[43]	CH1_MULTI_BLK_EN	The value of this register is derived from the DMAH_CH1_MULTI_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[42]	CH1_LOCK_EN	The value of this register is derived from the DMAH_CH1_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[41]	CH1_SRC_GAT_EN	The value of this register is derived from the DMAH_CH1_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[40]	CH1_DST_SCA_EN	The value of this register is derived from the DMAH_CH1_DST_SCA_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[39:38]	Reserved	Reserved
[37:35]	CH1_STW	The value of this register is derived from the DMAH_CH1_STW coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0
[34:32]	CH1_DTW	The value of this register is derived from the DMAH_CH1_DTW coreConsultant parameter.

		0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0
[31]	Reserved	Reserved
[30:28]	CH2_FIFO_DEPTH	The value of this register is derived from the DMAH_CH2_FIFO_DEPTH coreConsultant 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 coreConsultant 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 coreConsultant parameter. 0x0 = DMA 0x1 = SRC 0x2 = DST 0x3 = ANY Default value: 0x0
[13:12]	Reserved	Reserved
[11]	CH2_MULTI_BLK_EN	The value of this register is derived from the DMAH_CH2_MULTI_BLK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[10]	CH2_LOCK_EN	The value of this register is derived from the DMAH_CH2_LOCK_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0

[9]	CH2_SRC_GAT_EN	The value of this register is derived from the DMAH_CH2_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[8]	CH2_DST_SCA_EN	The value of this register is derived from the DMAH_CH2_DST_SCA_EN coreConsultant 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 coreConsultant 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 coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32 0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0

DMA Component Parameters Register 2(DMA_COMP_PARAMS_2)

Register	Offset	R/W	Description	Reset Value
DMA_COMP_PARAMS_2	DMA_BA+0x3E8	R	DMA Component Parameters Register 2	0x0

Bits	Description	
[63:44]	Reserved	Reserved
[43:40]	CH2_MULTI_BLK_TYPE	The values of these bit fields are derived from the DMAH_CHx_MULTI_BLK_TYPE coreConsultant 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
[39:36]	CH1_MULTI_BLK_TYPE	
[35:32]	CH0_MULTI_BLK_TYPE	
[31]	Reserved	Reserved
[30:28]	CH0_FIFO_DEPTH	The value of this register is derived from the DMAH_CH0_FIFO_DEPTH coreConsultant 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 DMAH_CH0_MULT_SIZE coreConsultant 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 DMAH_CH0_FC coreConsultant 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 DMAH_CH0_HC_LLP coreConsultant 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 coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[11]	CH0_MULTI_BLK_EN	The value of this register is derived from the DMAH_CH0_MULTI_BLK_EN coreConsultant 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 coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[9]	CH0_SRC_GAT_EN	The value of this register is derived from the DMAH_CH0_SRC_GAT_EN coreConsultant parameter. 0x0 = FALSE 0x1 = TRUE Default value: 0x0
[8]	CH0_DST_SCA_EN	The value of this register is derived from the DMAH_CH0_DST_SCA_EN coreConsultant 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 coreConsultant 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 coreConsultant parameter. 0x0 = NO_HARDCODE 0x1 = 8 0x2 = 16 0x3 = 32

		0x4 = 64 0x5 = 128 0x6 = 256 0x7 = reserved Default value: 0x0

DMA Component Parameters Register 1(DMA_COMP_PARAMS_1)

Register	Offset	R/W	Description	Reset Value
DMA_COMP_PARAMS_1	DMA_BA+0x3F0	R	DMA Component Parameters Register 1	0x0

Bits	Description	
[63:62]	Reserved	Reserved
[61]	STATIC_ENDIAN_SELECT	The value of this register is derived from the DMAH_STATIC_ENDIAN_SELECT coreConsultant parameter. 0 = FALSE 1 = TRUE Default value: 0x1
[60]	ADD_ENCODED_PARAMS	The value of this register is derived from the DMAH_ADD_ENCODED_PARAMS coreConsultant parameter. 0 = FALSE 1 = TRUE Default value: 0x0
[59:55]	NUM_HS_INT	The value of this register is derived from the DMAH_NUM_HS_INT coreConsultant parameter. 0x00 = 0 to 0x10 = 16 Default value: 0x10
[54:53]	M1_HDATA_WIDTH	The value of this register is derived from the DMAH_M1_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits Default value: 0x0
[52:47]	Reserved	Reserved
[46:45]	S_HDATA_WIDTH	The value of this register is derived from the DMAH_S_HDATA_WIDTH coreConsultant parameter. 0x0 = 32 bits 0x1 = 64 bits 0x2 = 128 bits 0x3 = 256 bits Default value: 0x0
[44:43]	NUM_MASTER_INT	The value of this register is derived from the DMAH_NUM_MASTER_INT coreConsultant parameter.

		0x0 = 1 to 0x3 = 4 Default value: 0x0
[42:40]	NUM_CHANNELS	The value of this register is derived from the DMAH_NUM_CHANNELS coreConsultant parameter. 0x0 = 1 to 0x7 = 8 Default value: 0x2
[39:36]	Reserved	Reserved
[35]	MABRST	The value of this register is derived from the DMAH_MABRST coreConsultant parameter. 0 = FALSE 1 = TRUE Default value: 0x0
[34:33]	INTR_IO	The value of this register is derived from the DMAH_INTR_IO coreConsultant parameter. 0x0 = ALL 0x1 = TYPE 0x2 = COMBINED 0x3 = reserved Default value: 0x2
[32]	BIG_ENDIAN	The value of this register is derived from the DMAH_BIG_ENDIAN coreConsultant parameter. 0 = FALSE 1 = TRUE Default value: 0x0
[31:12]	Reserved	Reserved
[11:8]	CH2_MAX_BLK_SIZE	The values of these bit fields are derived from the DMAH_CHx_MAX_BLK_SIZE coreConsultant 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_MAX_BLK_SIZE	
[3:0]	CH0_MAX_BLK_SIZE	

DMA Component ID Register(DCIR)

Register	Offset	R/W	Description	Reset Value
DCIR	DMA_BA+0x3F8	R	DMA Component ID Register	0x3232_302A 4457_1110

Bits	Description	
[63:32]	DMA_COMP_VERSION	Version of the component.
[31:0]	DMA_COMP_TYPE	Component Type number = 0x44_57_11_10

3.9 General Purpose I/O (GPIO)

3.9.1 Overview

The PAN108 series has up to 48 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. These 48 pins are arranged in 6 ports named as P0, P1, P2, P3, P4 and P5. Each of the 48 pins is independent and has the corresponding register bits to control the pin mode function and data.

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. Each I/O pin is equipped with a very weak individual pull-up resistor about 50kΩ for VDD is from 1.8V to 3.7V. Each I/O pin is equipped with a very weak individual pull-down resistor about 100kΩ for gnd.

3.9.2 Features

- Up to 48 gpio
- 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 support
- 2 supply voltage for different io
  - P40-P44 are supplied by a special voltage
  - The other pins are supplied by the other voltage

3.9.3 Block Diagram

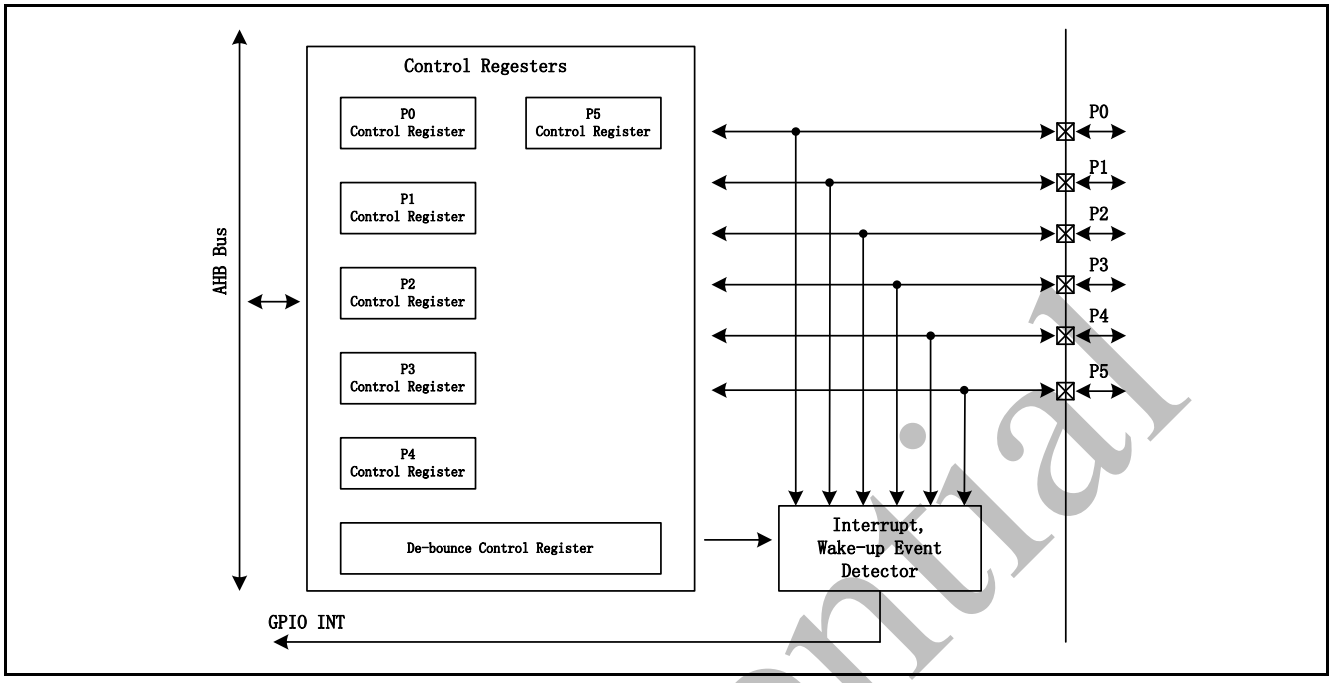


Figure 3-33 GPIO Controller Block Diagram

3.9.4 Basic Configuration

The GPIO 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.

3.9.5 Functional Description

The PAD diagram is introduced in Figure 3-34.

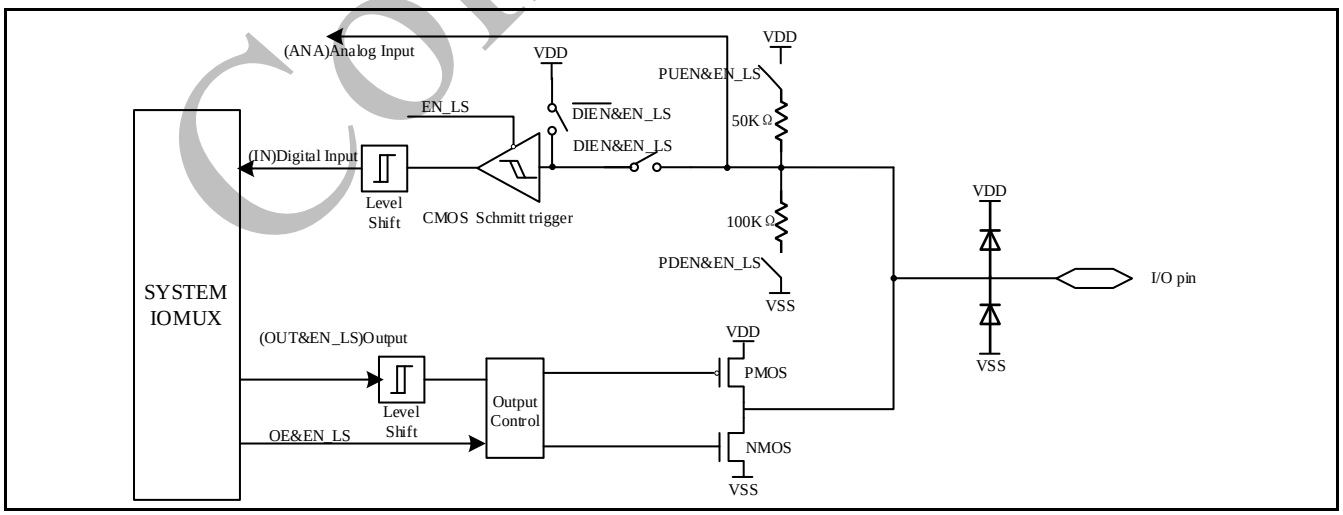


Figure 3-34 PAD Diagram

3.9.5.1 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.

3.9.5.2 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.

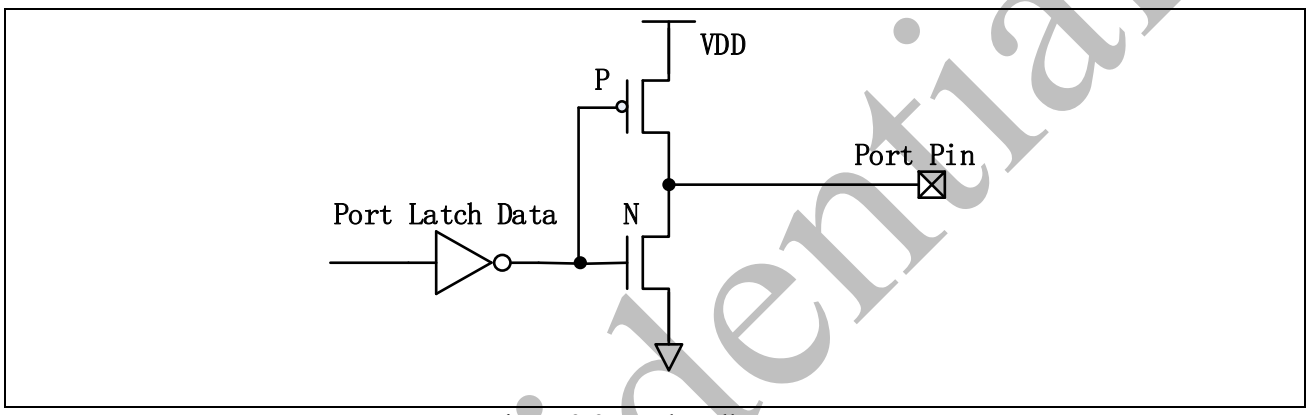


Figure 3-35 Push-Pull Output

3.9.5.3 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.

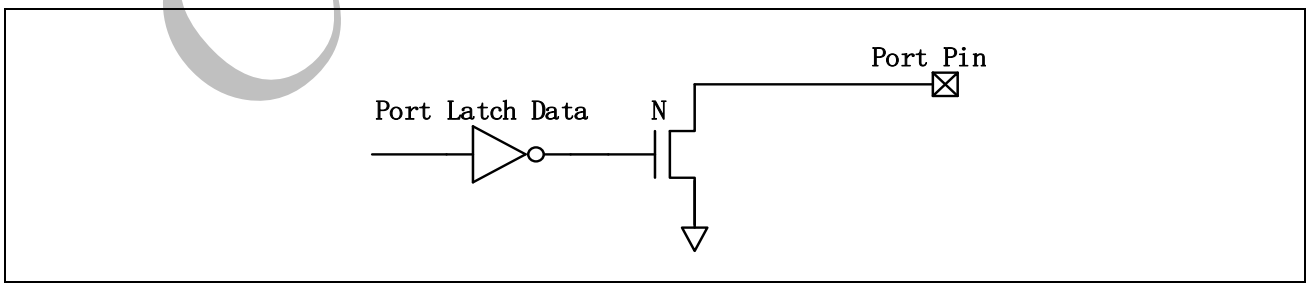


Figure 3-36 Open-Drain Output

3.9.5.4 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.

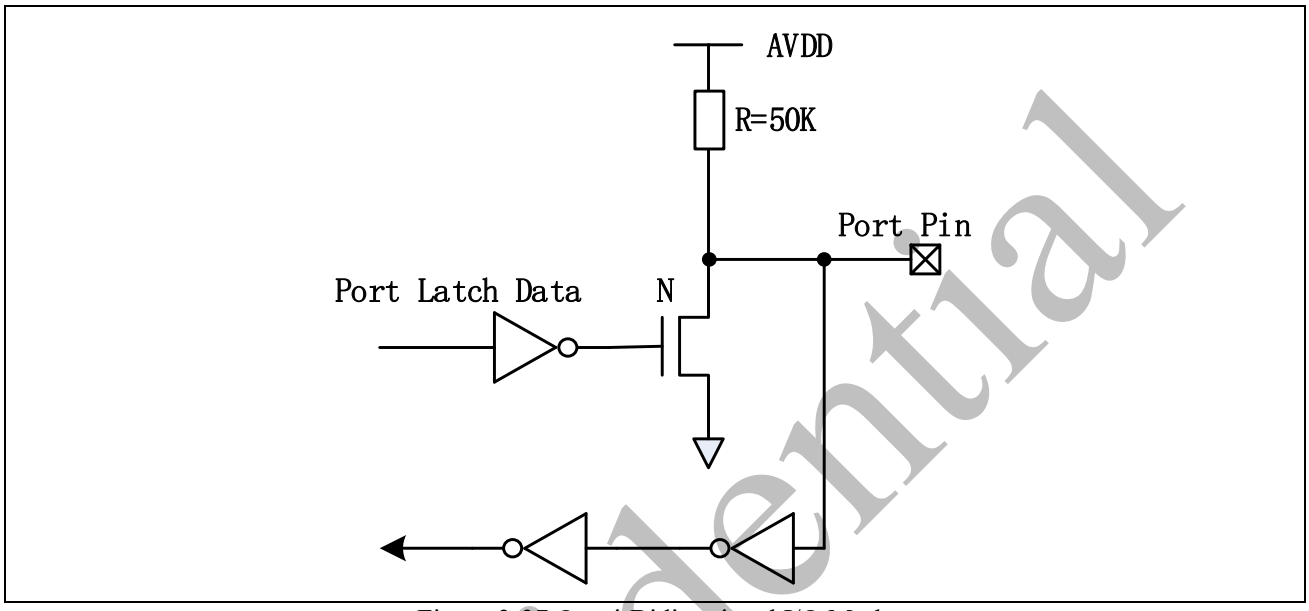


Figure 3-37 Quasi-Bidirectional I/O Mode

3.9.6 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.

3.9.7 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	0x00FF_0000
P0_DOUT	GP_BA+0x008	R/W	P0 Data Output Value	0x0000_0000
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	0x00FE_0000
P1_DOUT	GP_BA+0x048	R/W	P1 Data Output Value	0x0000_0000
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_0000
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	0x0077_0000
P3_DOUT	GP_BA+0x0C8	R/W	P3 Data Output Value	0x0000_0000
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
P4_MODE	GP_BA+0x100	R/W	P4 I/O Mode Control	0x0000_0000
P4_DINOFF	GP_BA+0x104	R/W	P4 Digital Input Path Disable Control	0x0000_0000
P4_DOUT	GP_BA+0x108	R/W	P4 Data Output Value	0x0000_0000

P4_DATMSK	GP_BA+0x10C	R/W	P4 Data Output Write Mask	0x0000_0000
P4_PIN	GP_BA+0x110	R	P4 Pin Value	0x0000_00XX
P4_DBEN	GP_BA+0x114	R/W	P4 De-bounce Enable Control	0x0000_0000
P4_INTTYPE	GP_BA+0x118	R/W	P4 Interrupt Mode Control	0x0000_0000
P4_INTEN	GP_BA+0x11C	R/W	P4 Interrupt Enable Control	0x0000_0000
P4_INTSRC	GP_BA+0x120	R/W	P4 Interrupt Source Flag	0x0000_0000
P5_MODE	GP_BA+0x140	R/W	P5 I/O Mode Control	0x0000_0000
P5_DINOFF	GP_BA+0x144	R/W	P5 Digital Input Path Disable Control	0x00F7_0000
P5_DOUT	GP_BA+0x148	R/W	P5 Data Output Value	0x0000_0000
P5_DATMSK	GP_BA+0x14C	R/W	P5 Data Output Write Mask	0x0000_0000
P5_PIN	GP_BA+0x150	R	P5 Pin Value	0x0000_00XX
P5_DBEN	GP_BA+0x154	R/W	P5 De-bounce Enable Control	0x0000_0000
P5_INTTYPE	GP_BA+0x158	R/W	P5 Interrupt Mode Control	0x0000_0000
P5_INTEN	GP_BA+0x15C	R/W	P5 Interrupt Enable Control	0x0000_0000
P5_INTSRC	GP_BA+0x160	R/W	P5 Interrupt Source Flag	0x0000_0000
GPIO_DBCTL	GP_BA+0x180	R/W	De-bounce Cycle Control	0x0000_0000
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	GPIO P0.2 Pin Data Input/Output	0x0000_0001
P03_PDIO	GP_BA+0x20C	R/W	GPIO P0.3 Pin Data Input/Output	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	GPIO P0.5 Pin Data Input/Output	0x0000_0001
P06_PDIO	GP_BA+0x218	R/W	GPIO P0.6 Pin Data Input/Output	0x0000_0001
P07_PDIO	GP_BA+0x21C	R/W	GPIO P0.7 Pin Data Input/Output	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	GPIO P1.6 Pin Data Input/Output	0x0000_0001
P17_PDIO	GP_BA+0x23C	R/W	GPIO P1.7 Pin Data Input/Output	0x0000_0001
P20_PDIO	GP_BA+0x240	R/W	GPIO P2.0 Pin Data Input/Output	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	GPIO P2.7 Pin Data Input/Output	0x0000_0001
P30_PDIO	GP_BA+0x260	R/W	GPIO P3.0 Pin Data Input/Output	0x0000_0001
P31_PDIO	GP_BA+0x264	R/W	GPIO P3.1 Pin Data Input/Output	0x0000_0001
P32_PDIO	GP_BA+0x268	R/W	GPIO P3.2 Pin Data Input/Output	0x0000_0001
P34_PDIO	GP_BA+0x270	R/W	GPIO P3.4 Pin Data Input/Output	0x0000_0001
P35_PDIO	GP_BA+0x274	R/W	GPIO P3.5 Pin Data Input/Output	0x0000_0001
P36_PDIO	GP_BA+0x278	R/W	GPIO P3.6 Pin Data Input/Output	0x0000_0001

P46_PDIO	GP_BA+0x298	R/W	GPIO P4.6 Pin Data Input/Output	0x0000_0001
P47_PDIO	GP_BA+0x29C	R/W	GPIO P4.7 Pin Data Input/Output	0x0000_0001
P50_PDIO	GP_BA+0x2A0	R/W	GPIO P5.0 Pin Data Input/Output	0x0000_0001
P51_PDIO	GP_BA+0x2A4	R/W	GPIO P5.1 Pin Data Input/Output	0x0000_0001
P52_PDIO	GP_BA+0x2A8	R/W	GPIO P5.2 Pin Data Input/Output	0x0000_0001
P53_PDIO	GP_BA+0x2AC	R/W	GPIO P5.3 Pin Data Input/Output	0x0000_0001
P54_PDIO	GP_BA+0x2B0	R/W	GPIO P5.4 Pin Data Input/Output	0x0000_0001
P55_PDIO	GP_BA+0x2B4	R/W	GPIO P5.5 Pin Data Input/Output	0x0000_0001
P56_PDIO	GP_BA+0x2B8	R/W	GPIO P5.6 Pin Data Input/Output	0x0000_0001
P57_PDIO	GP_BA+0x2BC	R/W	GPIO P5.7 Pin Data Input/Output	0x0000_0001

3.9.8 GPIO Register Description

3.9.8.1 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
P4_MODE	GP_BA+0x100	R/W	P4 I/O Mode Control	0x0000_0000
P5_MODE	GP_BA+0x140	R/W	P5 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.2 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	0x00FF_0000
P1_DINOFF	GP_BA+0x044	R/W	P1 Digital Input Path Disable Control	0x00FE_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	0x0077_0000
P4_DINOFF	GP_BA+0x104	R/W	P4 Digital Input Path Disable Control	0x0000_00C0
P5_DINOFF	GP_BA+0x144	R/W	P5 Digital Input Path Disable Control	0x00F7_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). **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
[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.
[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.

3.9.8.3 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_0000
P1_DOUT	GP_BA+0x048	R/W	P1 Data Output Value	0x0000_0000
P2_DOUT	GP_BA+0x088	R/W	P2 Data Output Value	0x0000_0000
P3_DOUT	GP_BA+0x0C8	R/W	P3 Data Output Value	0x0000_0000
P4_DOUT	GP_BA+0x108	R/W	P4 Data Output Value	0x0000_0000
P5_DOUT	GP_BA+0x148	R/W	P5 Data Output Value	0x0000_0000

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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.4 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
P4_DATMSK	GP_BA+0x10C	R/W	P4 Data Output Write Mask	0x0000_0000
P5_DATMSK	GP_BA+0x14C	R/W	P5 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.5 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
P4_PIN	GP_BA+0x110	R	P4 Pin Value	0x0000_00XX
P5_PIN	GP_BA+0x150	R	P5 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

3.9.8.6 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
P4_DBEN	GP_BA+0x114	R/W	P4 De-bounce Enable Control	0x0000_0000
P5_DBEN	GP_BA+0x154	R/W	P5 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.7 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
P4_INTTYPE	GP_BA+0x118	R/W	P4 Interrupt Mode Control	0x0000_0000
P5_INTTYPE	GP_BA+0x158	R/W	P5 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 HCLK 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.8 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
P4_INTEN	GP_BA+0x11C	R/W	P4 Interrupt Enable Control	0x0000_0000
P5_INTEN	GP_BA+0x15C	R/W	P5 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.9 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
P4_INTSRC	GP_BA+0x120	R/W	P4 Interrupt Source Flag	0x0000_0000
P5_INTSRC	GP_BA+0x160	R/W	P5 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.9.8.10 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_0000

Bits	Descriptions	
[31]	GPIO_RETENION	Used for low power. 0 = no use 1= all io control signals are controlled by gpio.
[30:5]	Reserved	Reserved.
[4]	DBCLKSRC	De-bounce Counter Clock Source Selection 0 = De-bounce counter clock source is HCLK. 1 = De-bounce counter clock source is 32 kHz internal low speed RC oscillator (LIRC).
[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.

3.9.8.11 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	GPIO P0.2 Pin Data Input/Output	0x0000_0001
P03_PDIO	GP_BA+0x20C	R/W	GPIO P0.3 Pin Data Input/Output	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	GPIO P0.5 Pin Data Input/Output	0x0000_0001
P06_PDIO	GP_BA+0x218	R/W	GPIO P0.6 Pin Data Input/Output	0x0000_0001
P07_PDIO	GP_BA+0x21C	R/W	GPIO P0.7 Pin Data Input/Output	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	GPIO P1.6 Pin Data Input/Output	0x0000_0001
P17_PDIO	GP_BA+0x23C	R/W	GPIO P1.7 Pin Data Input/Output	0x0000_0001
P20_PDIO	GP_BA+0x240	R/W	GPIO P2.0 Pin Data Input/Output	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	GPIO P2.7 Pin Data Input/Output	0x0000_0001
P30_PDIO	GP_BA+0x260	R/W	GPIO P3.0 Pin Data Input/Output	0x0000_0001
P31_PDIO	GP_BA+0x264	R/W	GPIO P3.1 Pin Data Input/Output	0x0000_0001
P32_PDIO	GP_BA+0x268	R/W	GPIO P3.2 Pin Data Input/Output	0x0000_0001
P34_PDIO	GP_BA+0x270	R/W	GPIO P3.4 Pin Data Input/Output	0x0000_0001
P35_PDIO	GP_BA+0x274	R/W	GPIO P3.5 Pin Data Input/Output	0x0000_0001
P36_PDIO	GP_BA+0x278	R/W	GPIO P3.6 Pin Data Input/Output	0x0000_0001
P46_PDIO	GP_BA+0x298	R/W	GPIO P4.6 Pin Data Input/Output	0x0000_0001
P47_PDIO	GP_BA+0x29C	R/W	GPIO P4.7 Pin Data Input/Output	0x0000_0001
P50_PDIO	GP_BA+0x2A0	R/W	GPIO P5.0 Pin Data Input/Output	0x0000_0001
P51_PDIO	GP_BA+0x2A4	R/W	GPIO P5.1 Pin Data Input/Output	0x0000_0001
P52_PDIO	GP_BA+0x2A8	R/W	GPIO P5.2 Pin Data Input/Output	0x0000_0001
P53_PDIO	GP_BA+0x2AC	R/W	GPIO P5.3 Pin Data Input/Output	0x0000_0001
P54_PDIO	GP_BA+0x2B0	R/W	GPIO P5.4 Pin Data Input/Output	0x0000_0001
P55_PDIO	GP_BA+0x2B4	R/W	GPIO P5.5 Pin Data Input/Output	0x0000_0001
P56_PDIO	GP_BA+0x2B8	R/W	GPIO P5.6 Pin Data Input/Output	0x0000_0001
P57_PDIO	GP_BA+0x2BC	R/W	GPIO P5.7 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, n=3, n=7 are reserved. Max. n=7 for port 4, n=0,.5 are reserved. Max. n=7 for port 5

3.10 Universal Serial Bus(USB)

3.10.1 Overview

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

3.10.2 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 DP(1.5kΩ)/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.

3.10.3 Block Diagram

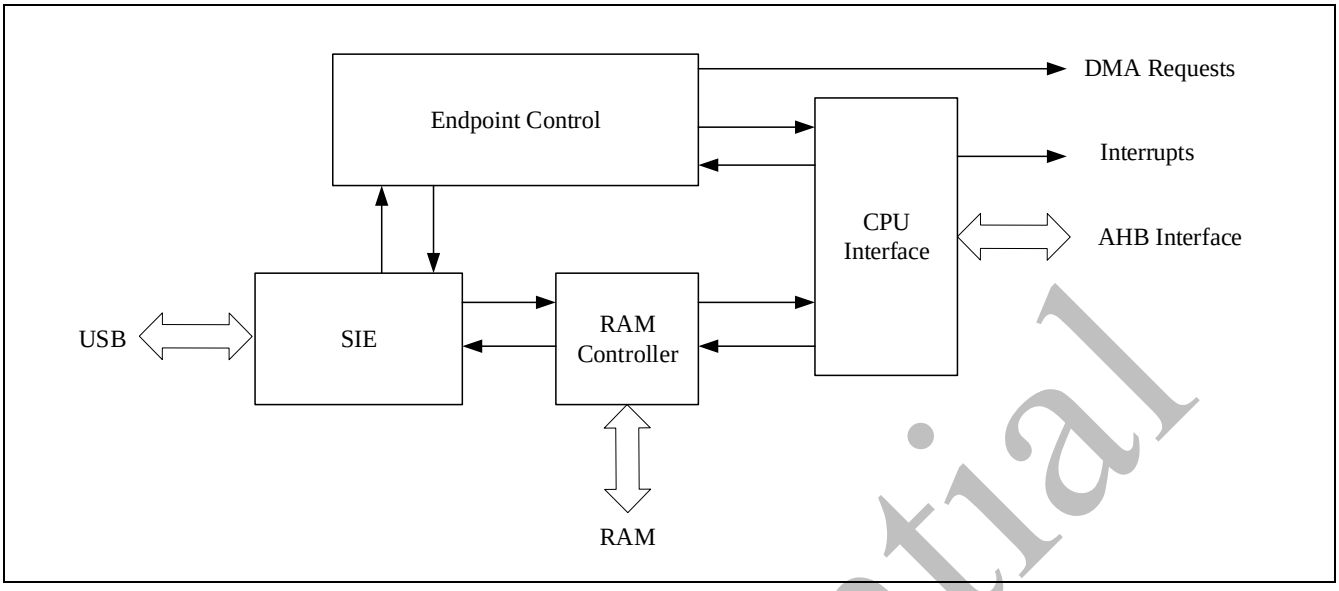


Figure 3-38 USB Block Diagram

3.10.4 Functional Description

3.10.4.1 Support BULK/ISOCHRONOUS transactions

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

3.10.4.2 USB DMA Operation

DMA MODE0 : OUT/RX ENDPOINTS

For operation in DMA Mode 0, A OUT endpoint or a Rx endpoint should be programmed as follows:

- The relevant interrupt enable bit in the IntrOutE/IntrInE register should be set to 1.
- The DMAEnab bit (D5) of the appropriate OutCSR2 register should be set to 0.

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

- ADDR : Memory address to store packet
- COUNT : Size of packet (determined by OutCount / RxCount registers)
- CNTL : DMA Enable (D0) = 1, Direction (D1) = 0, DMA Mode (D2) = 0, Interrupt Enable (D3) = 1

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

the OutPktRdy/RxPktRdy bit.

DMA MODE0 : IN/TX ENDPOINTS

For operation in DMA Mode 0, a IN-EndPoint or a TX-Endpoint should be programmed as follows:

- The relevant interrupt enable bit in the IntrIn1E/IntrOutE register should be set to 1.
- The DMAEnab bit (D4) of the appropriate InCSR2 register should be set to 0.

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
- CNTL : DMA Enable (D0) = 1, Direction (D1) = 1, DMA Mode (D2) = 0, Interrupt Enable (D3) = 1

The DMA controller will then request bus mastership and transfer the packet to USB FIFO. It will interrupt the processor when it has completed the transfer and the processor can set the InPktRdy/TxPktRdy bit.

DMA MODE 1 : OUT/RX ENDPOINTS

For operation in DMA Mode 1 with a OUT-EndPoint or a IN-EndPoint, the DMA Controller should be programmed as follows:

- ADDR : Memory address of buffer to store transfer
- COUNT : Maximum size of data buffer
- CNTL : DMA Enable (D0) = 1, Direction (D1) = 0, DMA Mode (D2) = 1, Interrupt Enable (D3) = 1, Max. Packet Size (D14 – 8) = the value set in the corresponding OutMaxP/InMaxP register

and the OUT/ IN endpoint should be programmed as follows:

- The relevant interrupt enable bit in the IntrOutnE/IntrRxnE register should be set to 1.
- The AutoClear (D7), DMAEnab (D5) and DMAMode (D4) bits of the appropriate OutCSR2/RxCSR2 register should be set to 1.

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 OutPktRdy/RxPktRdy bit in the appropriate OutCSR1/INCSR1 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 OutCount/Count0 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.

DMA MODE1 : IN/TX ENDPOINTS

For operation in DMA Mode 1 with a IN-EndPoint or a OUT-EndPoint, the DMA controller should be programmed as follows:

- ADDR : Memory address of data block to send
- COUNT : Size of data block
- CNTL : DMA Enable (D0) = 1, Direction (D1) = 1, DMA Mode (D2) = 1, Interrupt Enable (D3) = 1.Max. Packet Size (D14 – 8) = the value set in the corresponding USB InMaxP /OUTMaxP register

and the IN /OUT endpoint should be programmed as follows:

- The relevant interrupt enable bit in the IntrIn1E /IntrOut1E register should be set to 1 (so that any errors will cause an interrupt to be generated).
- The AutoSet (D7), DMAEnab (D4) and DMAMode (D2) bits of the appropriate InCSR2/OUTCSR2 register should be 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 InPktRdy/OutPktRdy bit in the appropriate InCSR1/ OUTCSR1 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 InPktRdy/OutPktRdy bit will not have been set. So the processor should set the InPktRdy /OutPktRdy 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 InPktRdy/ OutPktRdy bit in order to send a null packet signifying the end of the transfer.

3.10.4.3 Support plug in/out interrupt

Enable DM pull-up resistor (ANA_MISC[28]=1), When the USB is plugged in,

{DP,DM}={1,0} will generate a PUPIF interrupt for more than 1ms. Reading the IntrUSB, if {DP,DM}={1,0}, it means that the USB is plugged in; If {DP,DM}={1,1} exceeds 1ms, PUPIF interrupt will be generated. Reading the IntrUSB, if {DP,DM}={1,1}, it means USB is plugged out.

3.10.5 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=0x400A_0000				
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	R/W	Control Status register for Endpoint 0 (Index register set to select Endpoint 0)	0x0000_0000
InCSR1			Control Status register 1 for IN endpoint (Index register set to select Endpoints 1 – 3)	
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
CNTL1	USB_BA+0x204	R/W	DMA Channel 1 Control register	0x0000_0000
ADDR1	USB_BA+0x208	R/W	DMA Channel 1 AHB Memory Address register	0x0000_0000

COUNT1	USB_BA+0x20C	R/W	DMA Channel 1 Byte Count register	0x0000_0000

3.10.6 USB Register Description

3.10.6.1 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.

3.10.6.2 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 MUSBFSFC 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.

3.10.6.3 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.

3.10.6.4 IntrIn2

Register	Offset	R/W	Description	Reset Value
IntrIn2	USB_BA+0x03	R	Reserved	0x00

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.10.6.5 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

3.10.6.6 IntrOut2

Register	Offset	R/W	Description	Reset Value
IntrOut2	USB_BA+0x05	R	Reserved	0x00

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.10.6.7 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 MUSBFSFC is in Suspend mode.
[0]	Suspend	Set when Suspend signaling is detected on the bus.

3.10.6.8 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.

3.10.6.9 IntrIn2E

Register	Offset	R/W	Description	Reset Value
IntrIn2E	USB_BA+0x08	R/W	Reserved	0xFF

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.10.6.10 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

3.10.6.11 IntrOut2E

Register	Offset	R/W	Description	Reset Value
IntrOut2E	USB_BA+0x0A	R/W	Reserved	0xFF

Bits	Descriptions	
[7:0]	Reserved	Reserved

3.10.6.12 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.

3.10.6.13 Frame1

Register	Offset	R/W	Description	Reset Value
Frame1	USB_BA+0x0C	R/W	Frame number register	0x00

Bits	Descriptions	
[7:0]	Frame1	Lower 8 bits of the last received frame number

3.10.6.14 Frame2

Register	Offset	R/W	Description	Reset Value
Frame2	USB_BA+0x0D	R/W	Frame number register	0x00

Bits	Descriptions	
[7:3]	Reserved	Reserved
[2:0]	Frame2	Upper 3 bits of the last received frame number

3.10.6.15 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.

3.10.6.16 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).

3.10.6.17 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	R. 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.

3.10.6.18 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.

3.10.6.19 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	R. 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.

3.10.6.20 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

3.10.6.21 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。

3.10.6.22 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.

3.10.6.23 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

3.10.6.24 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

3.10.6.25 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.

3.10.6.26 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

3.10.6.27 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.

3.10.6.28 CNTL1

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 packet size	Max packet 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: write-out endpoint 1: read-IN endpoint
[0]	Enable DMA	Enable DMA transfer.

3.10.6.29 ADDR1

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)

3.10.6.30 COUNT1

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

3.11 Enhanced PWM Generator (PWM)

3.11.1 Overview

The PAN108 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.

3.11.2 Features

- Support 3 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

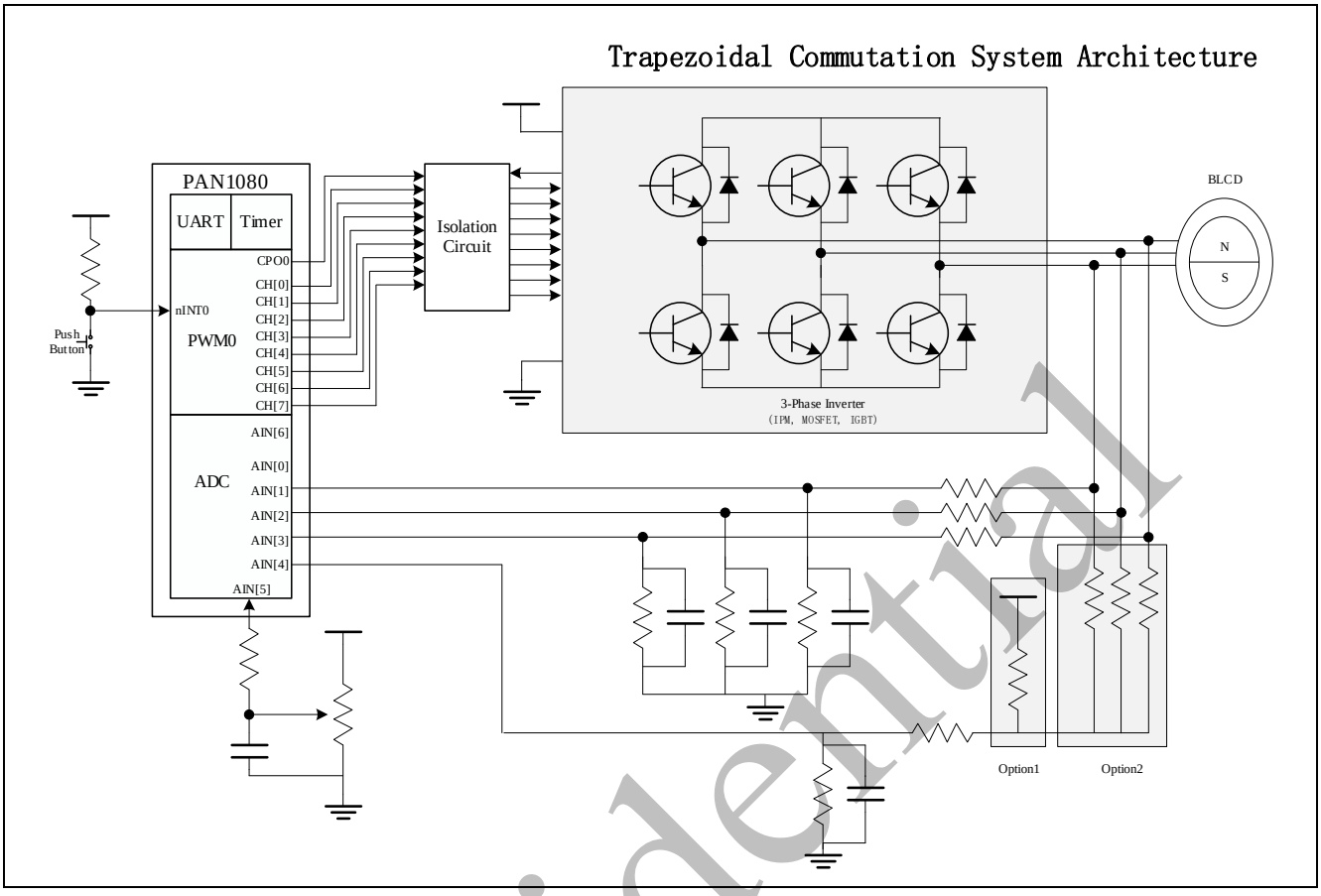


Figure 3-39 Application Circuit Diagram

3.11.3 Block Diagram

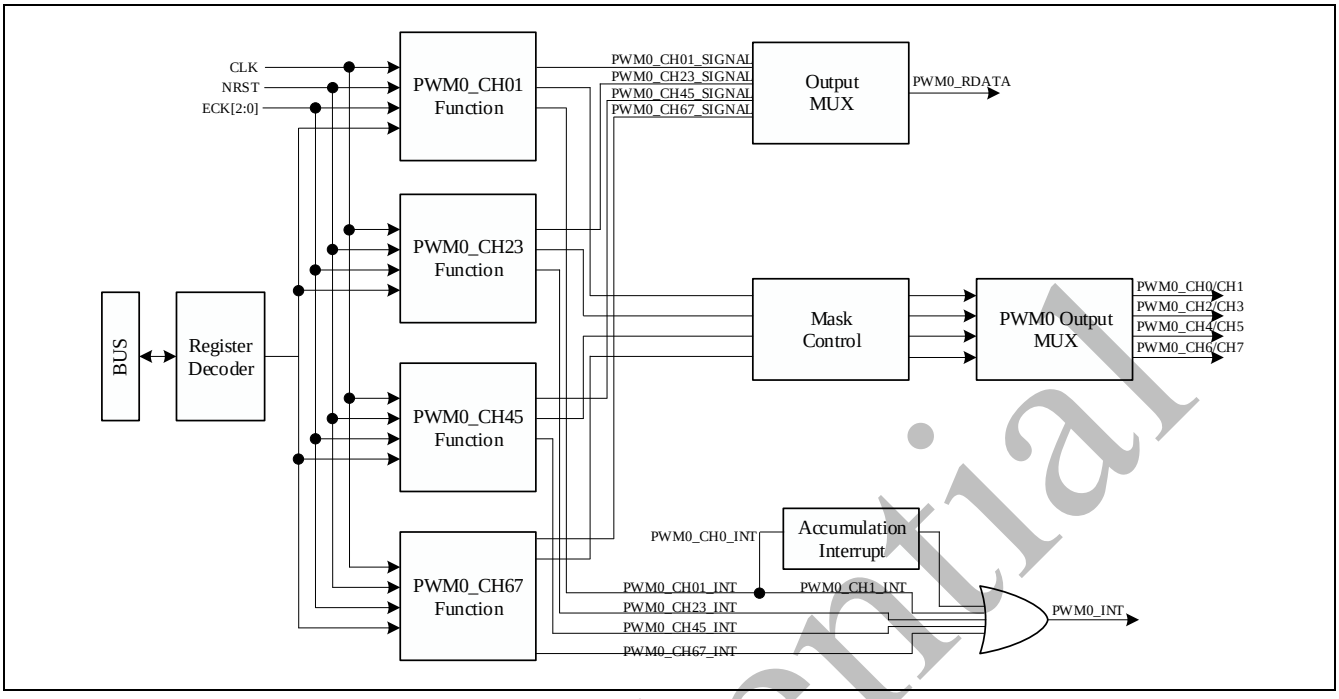


Figure 3-40 PWM0 Block Diagram

Figure 3-40 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).

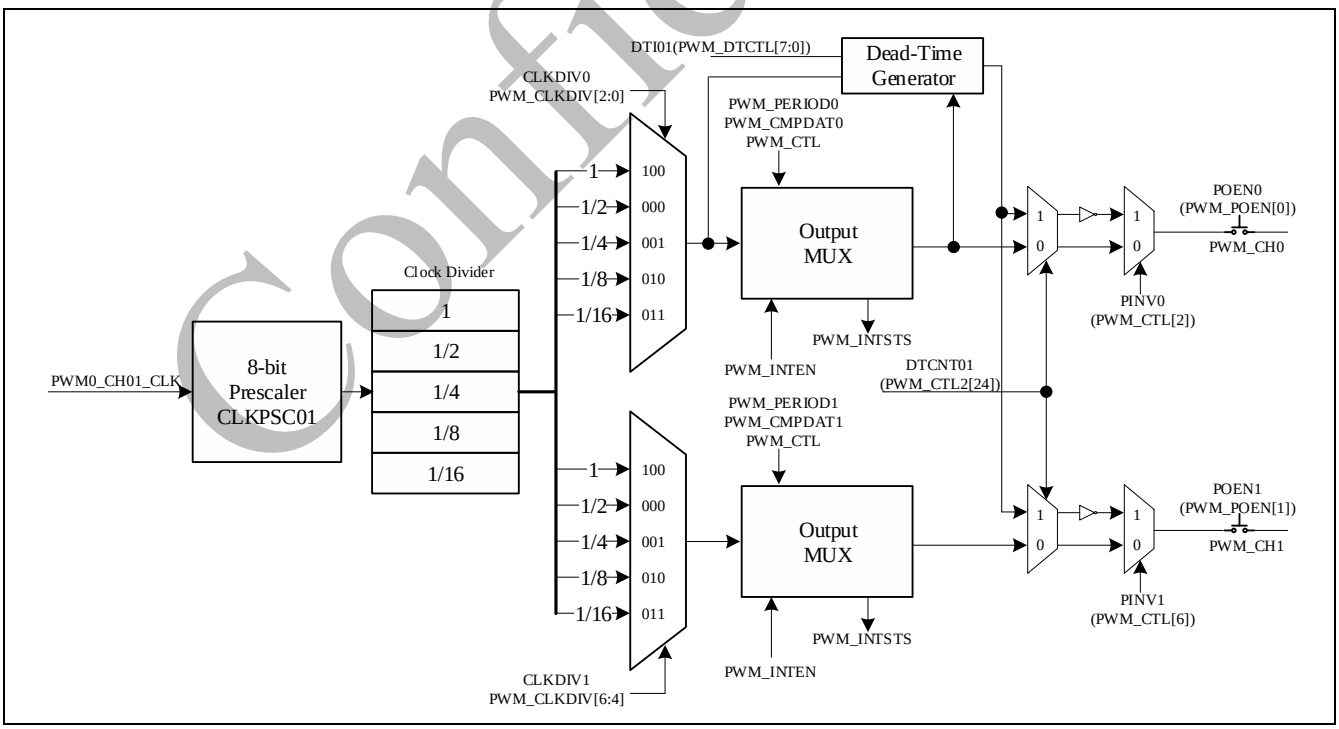


Figure 3-41 PWM0 Generator 0 Architecture Diagram

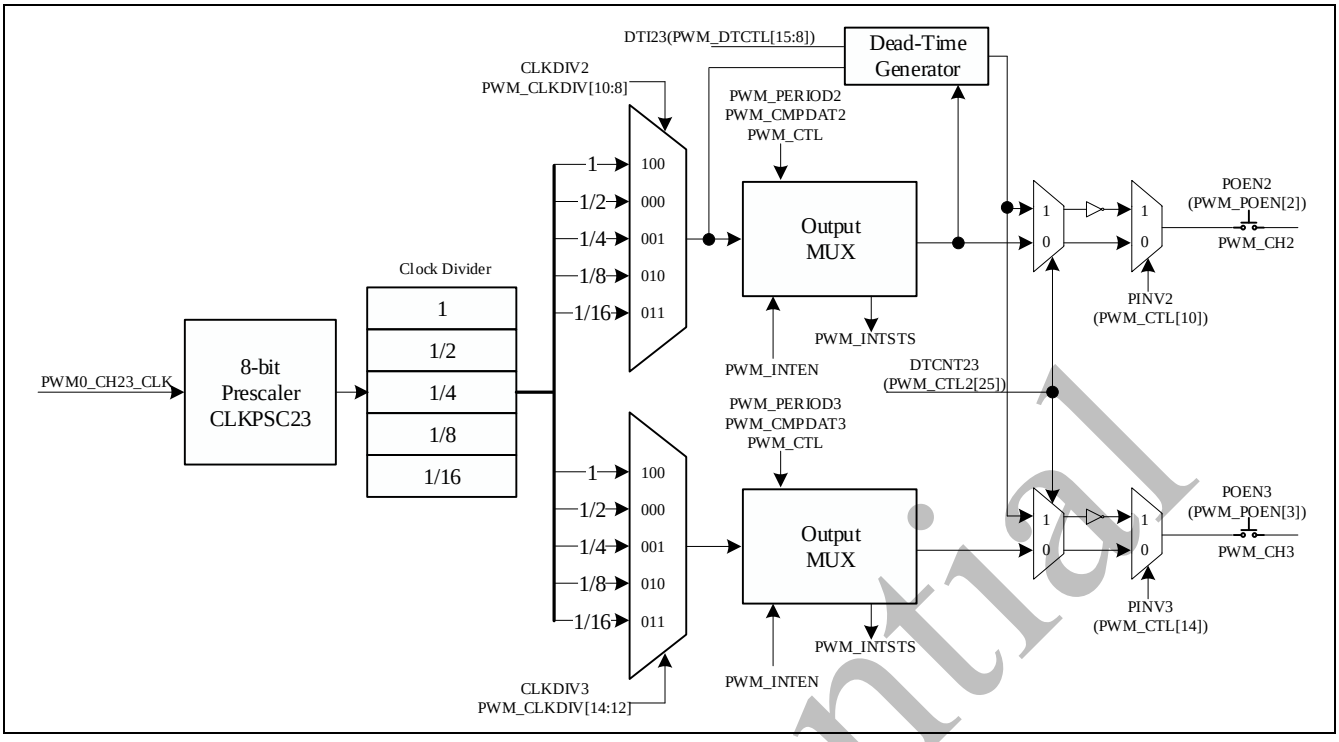


Figure 3-42 PWM0 Generator 2 Architecture Diagram

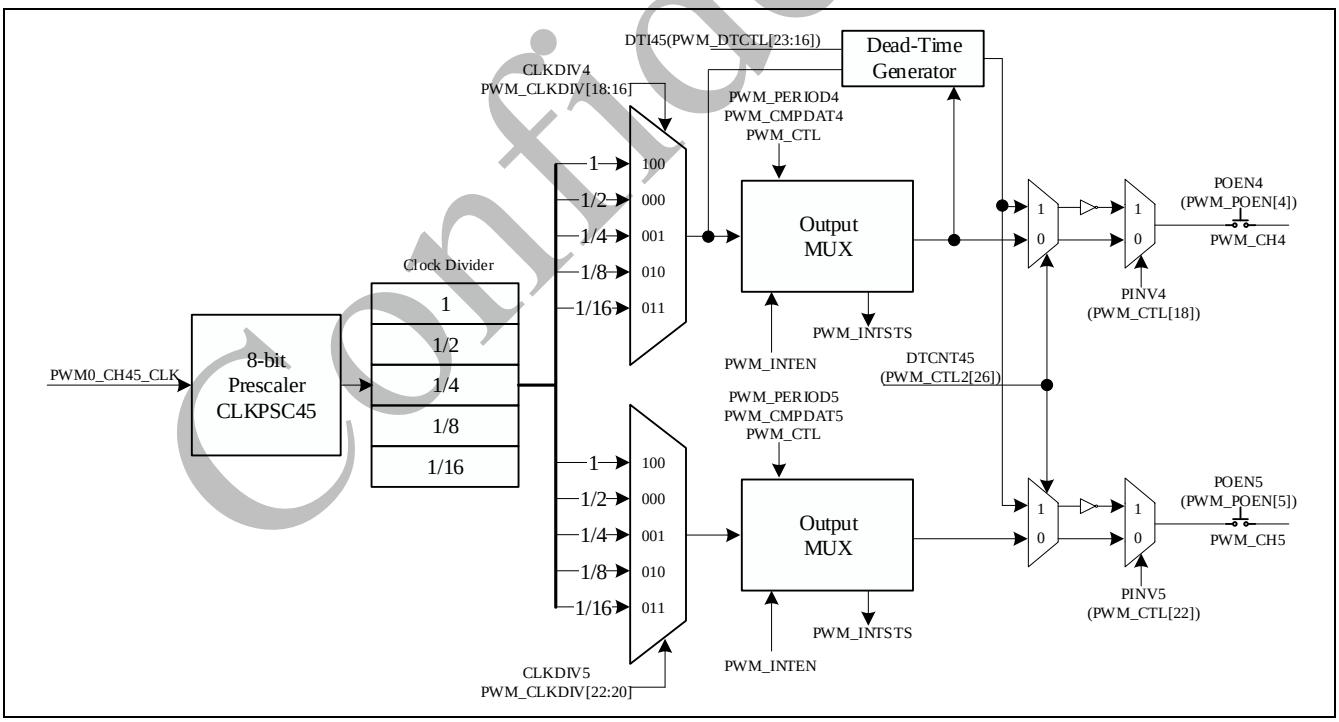


Figure 3-43 PWM0 Generator 4 Architecture Diagram

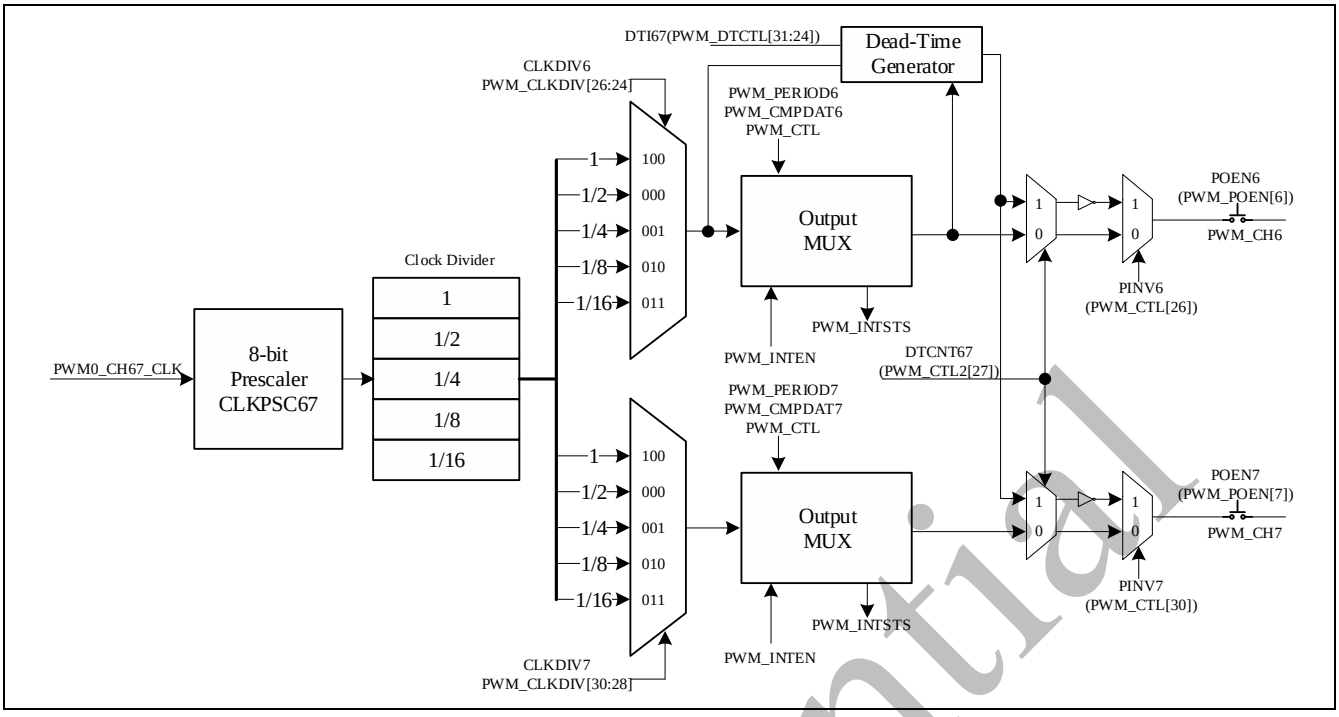


Figure 3-44 PWM0 Generator 6 Architecture Diagram

3.11.4 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.

3.11.5 Functional Description

3.11.5.1 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 and set PIF period interrupt flag.

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

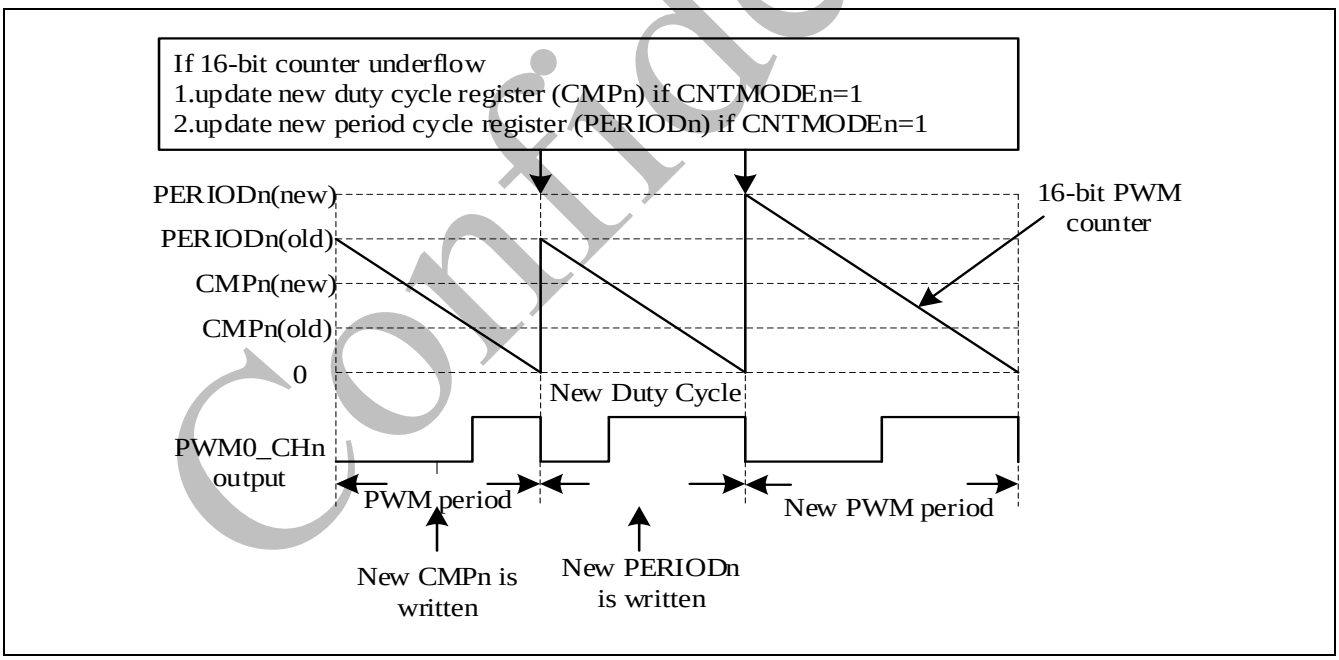


Figure 3-45 Edge-aligned Type PWM

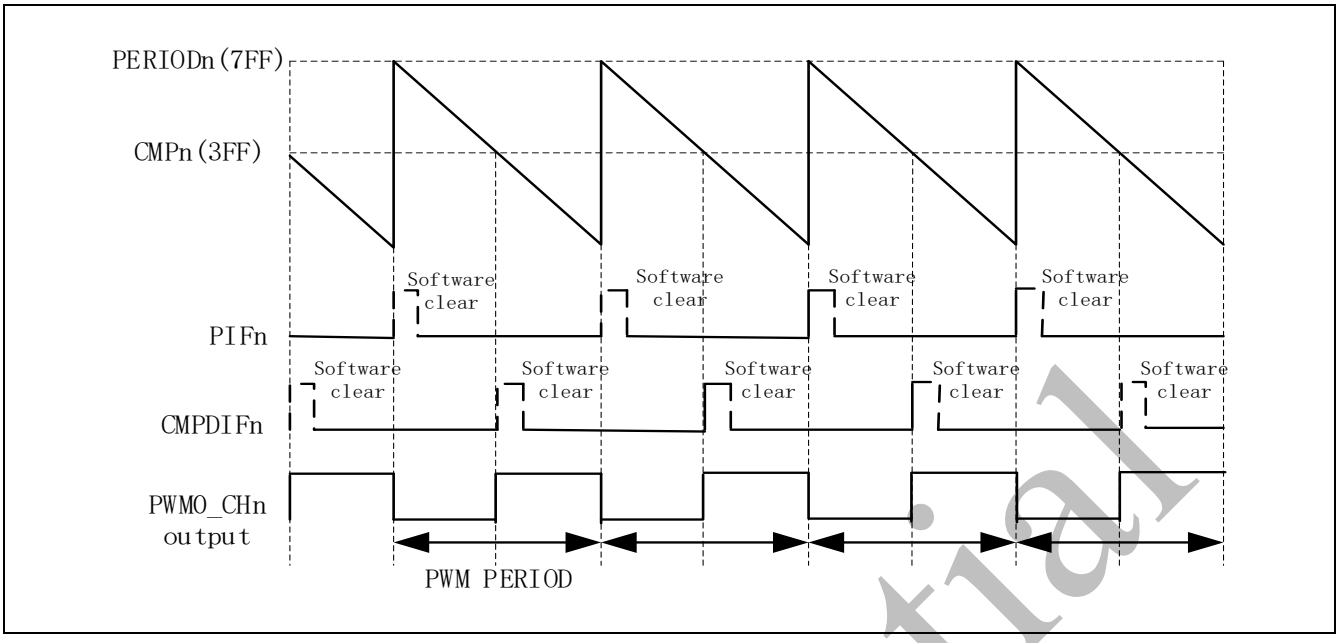


Figure 3-46 PWM Edge-aligned Waveform Timing Diagram

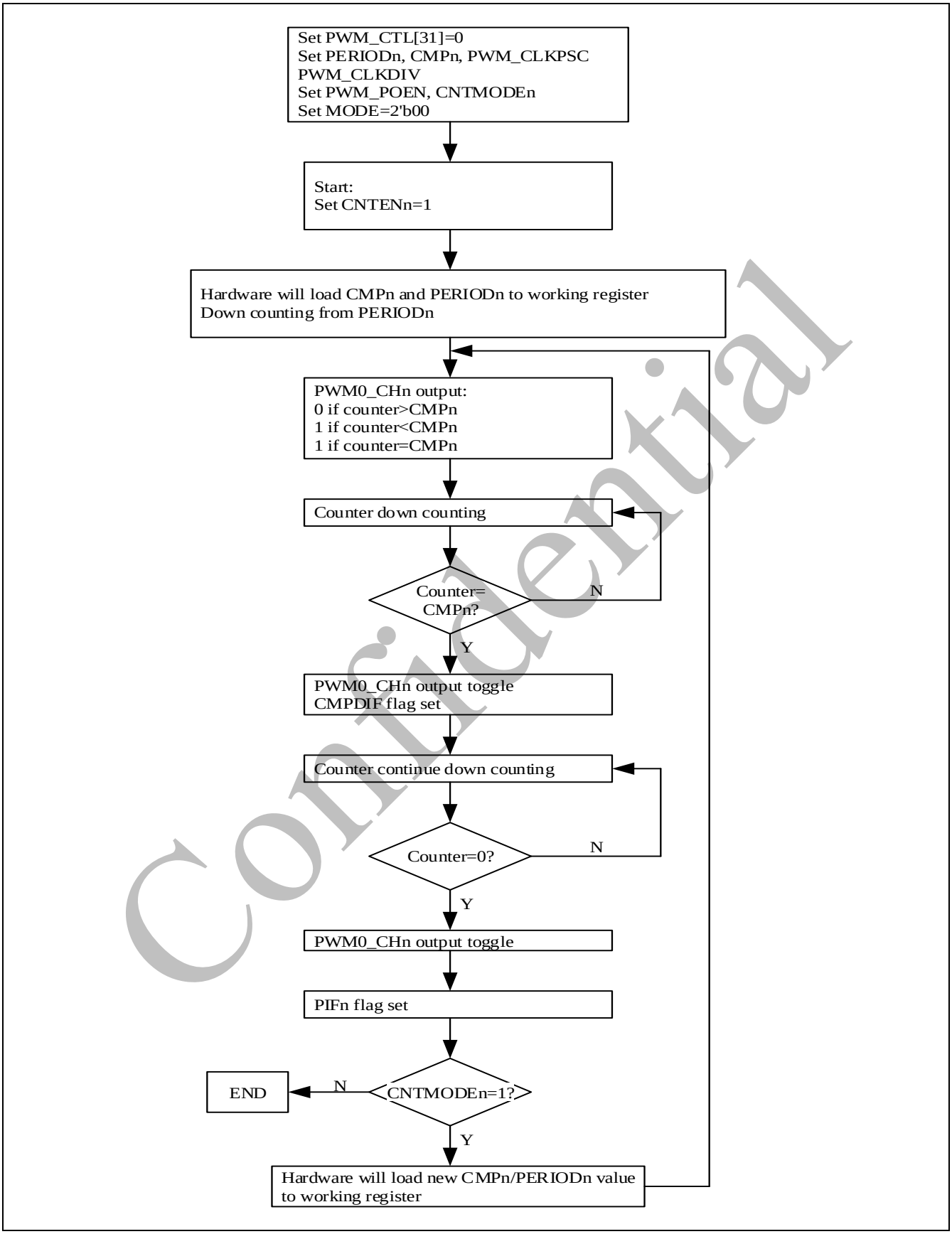


Figure 3-47 Edge-aligned Flow Diagram

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 3-49. The pulse width modulation follows the formula below and the legend of PWM counter Comparator is shown in Figure 3-48. 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

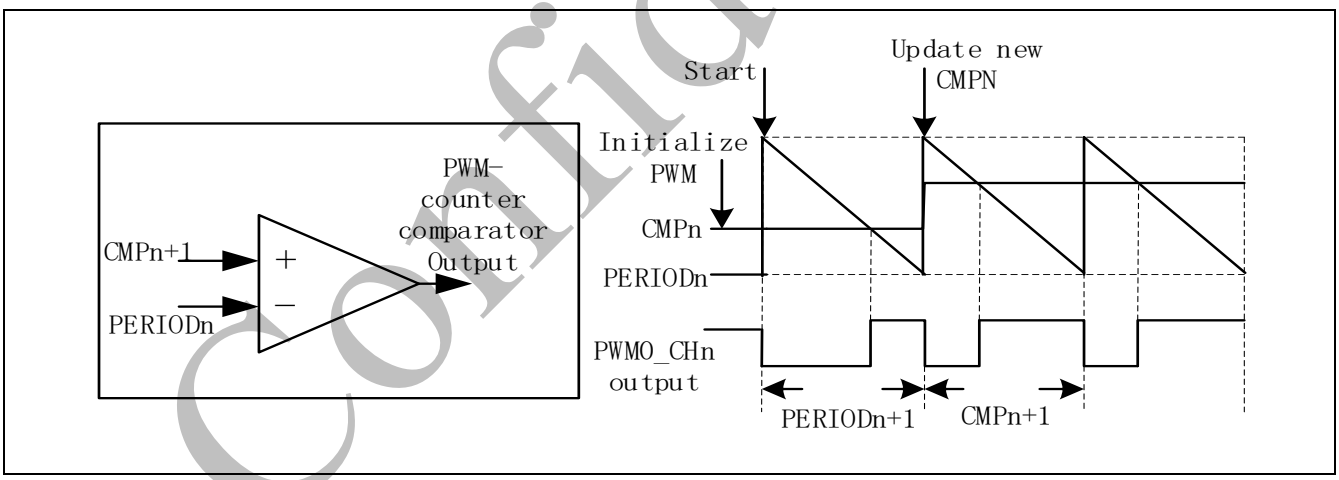


Figure 3-48 Legend of Internal Comparator Output of PWM Counter

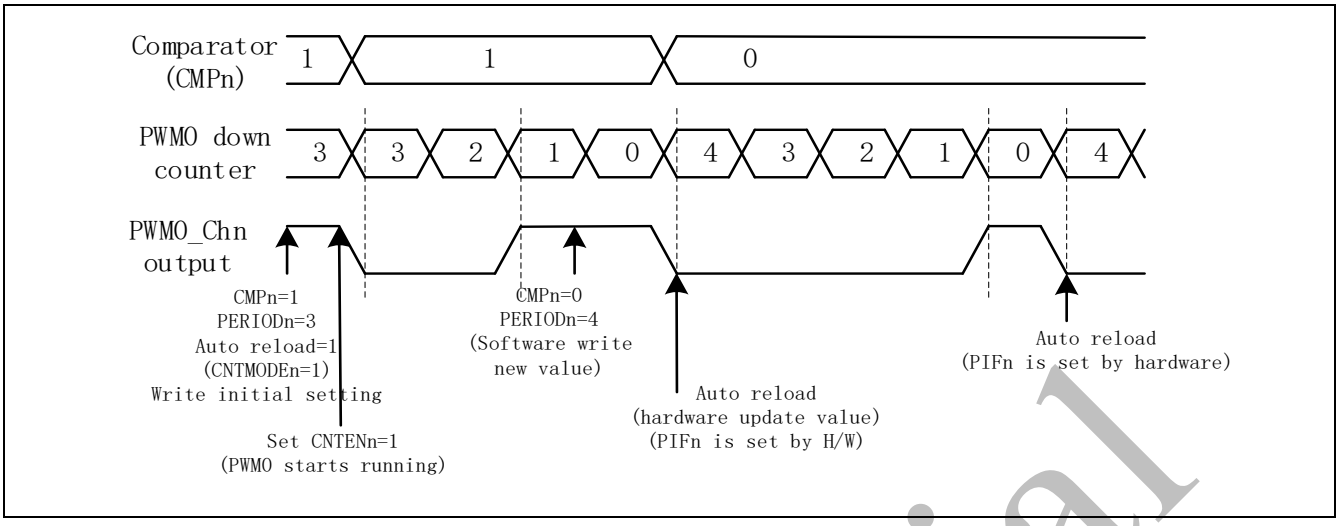


Figure 3-49 PWM Counter Operation Timing

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).

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 3-50, Figure 3-51, Figure 3-52 and Figure 3-53 show the Center-aligned PWM timing and operation flow.

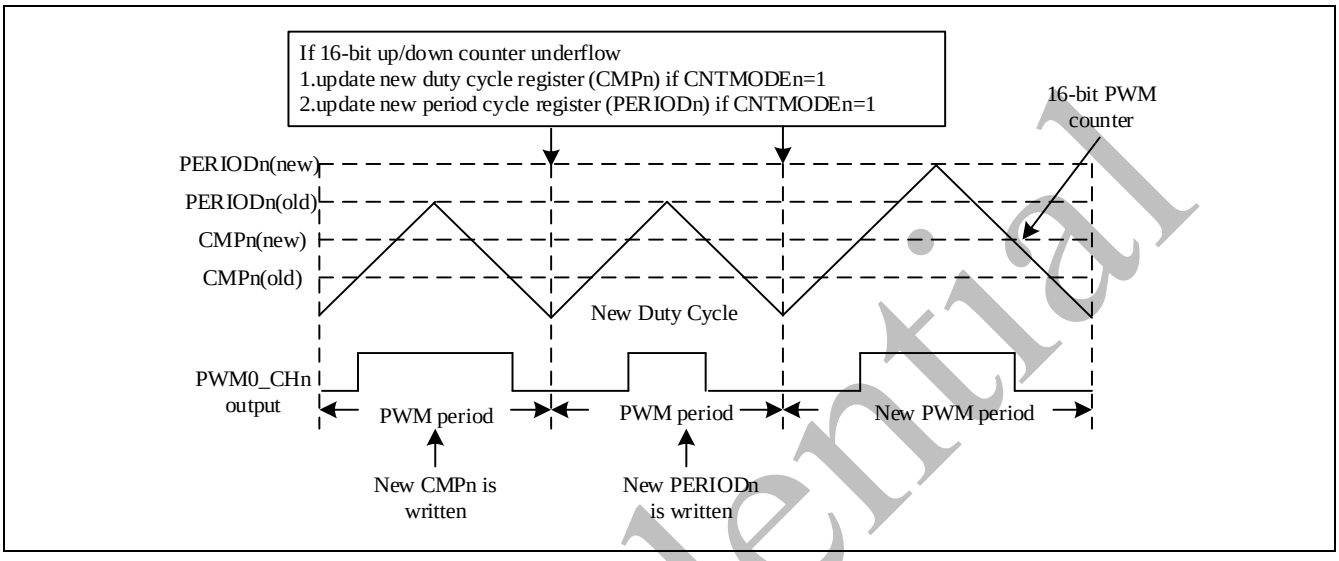


Figure 3-50 Center-aligned Type PWM

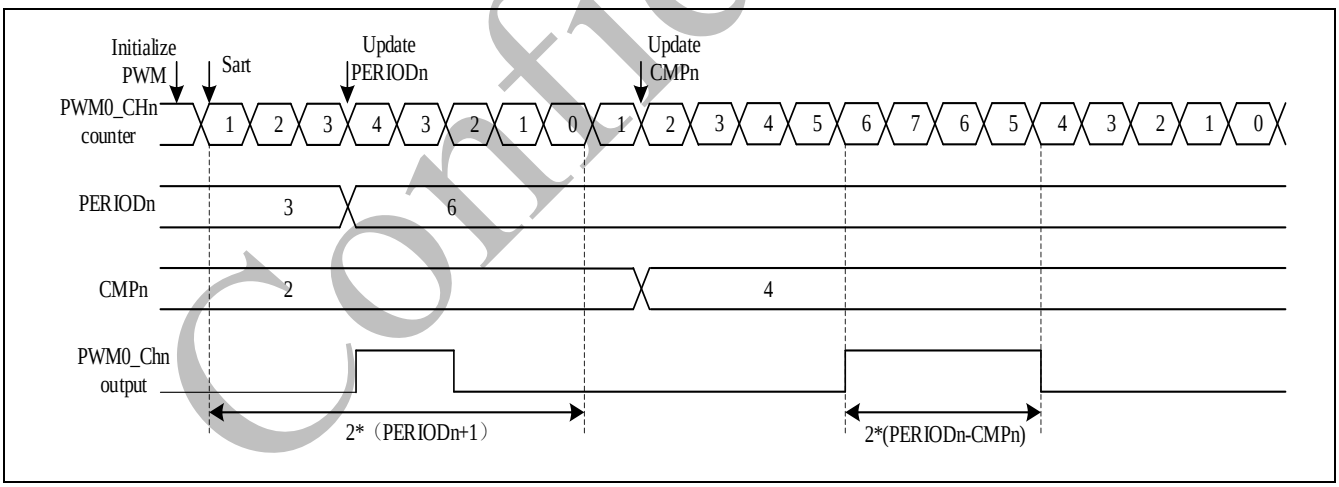


Figure 3-51 Center-aligned Type Operation Timing

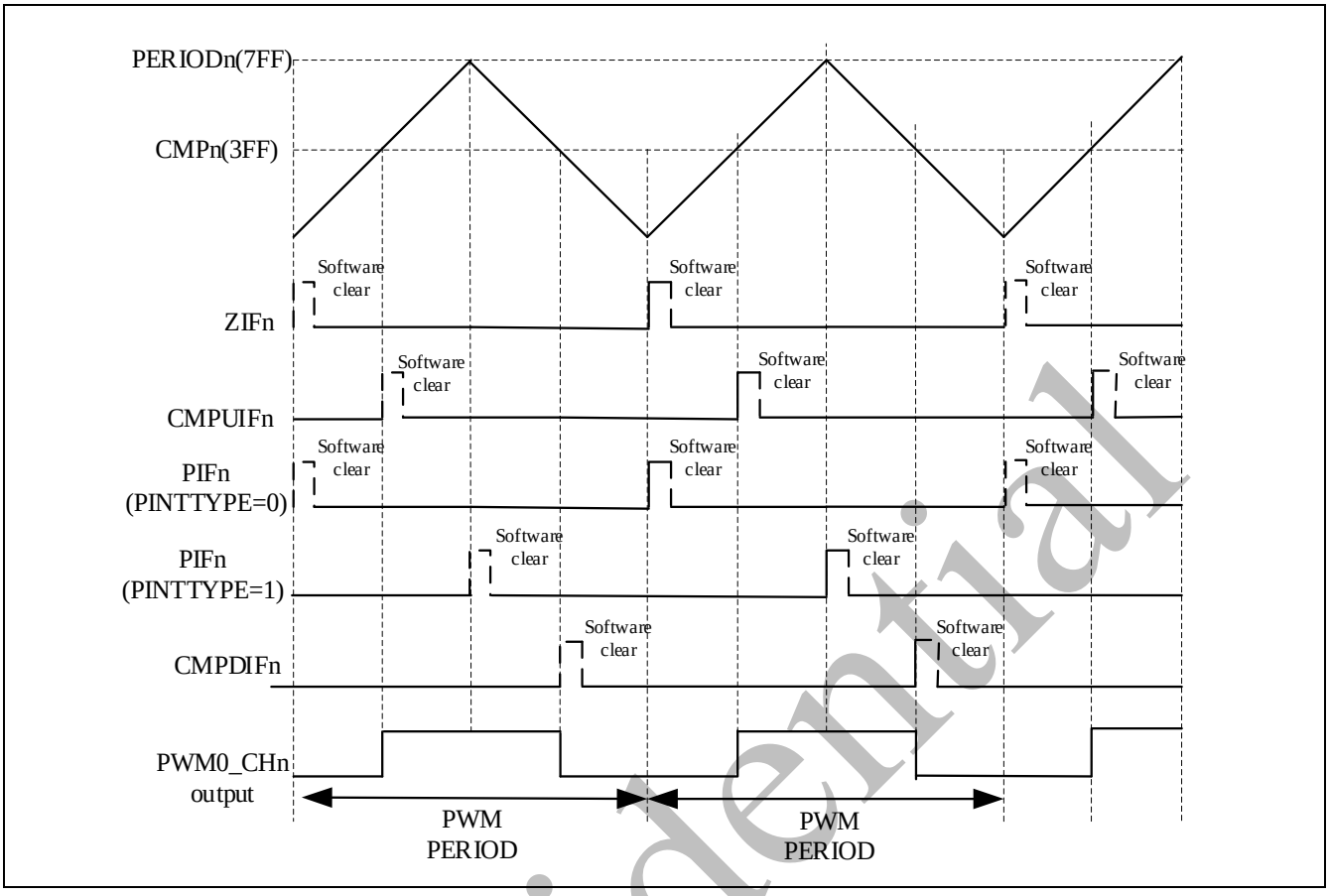


Figure 3-52 PWM Center-aligned Waveform Timing Diagram

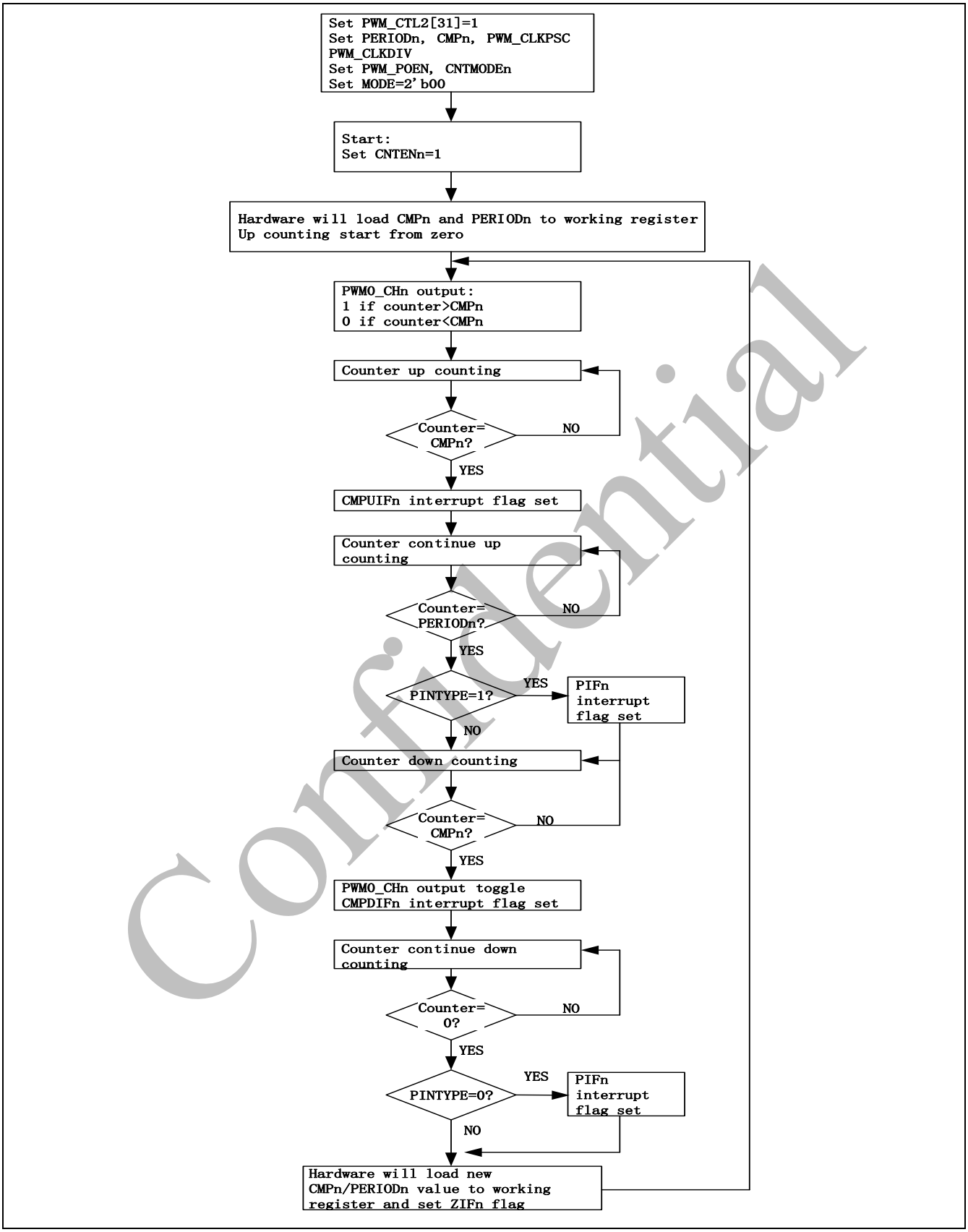


Figure 3-53 Center-aligned Flow Diagram

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).

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 3-54, Figure 3-55, Figure 3-56 show the Precise Center-aligned PWM timing and operation flow.

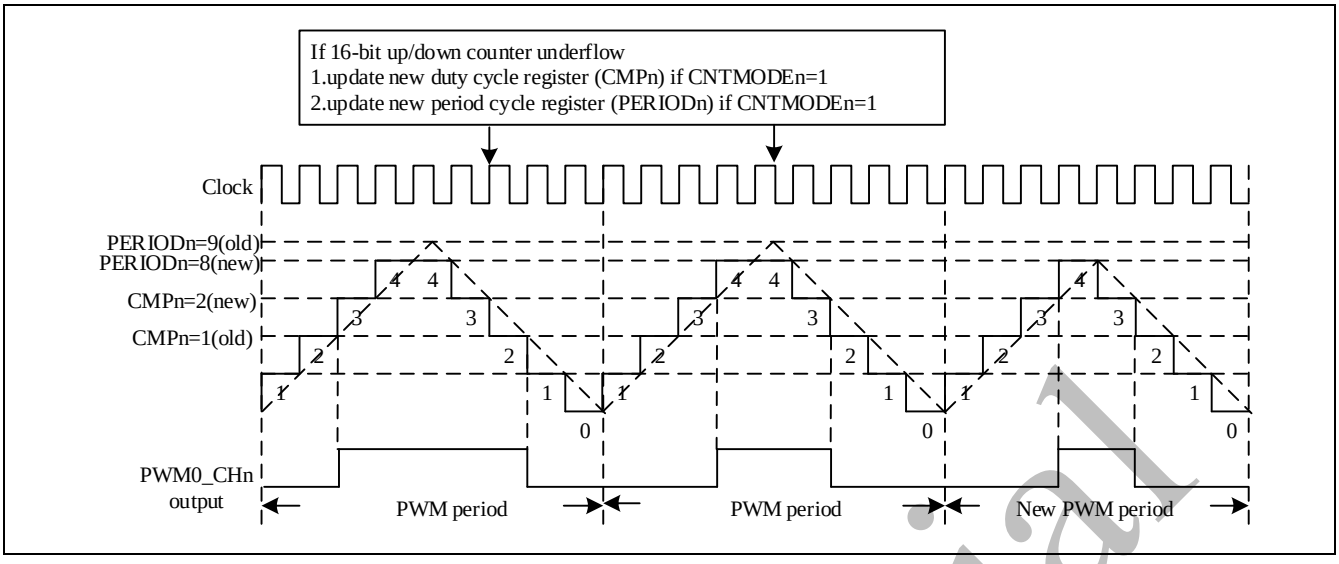


Figure 3-54 Precise Center-aligned Type PWM

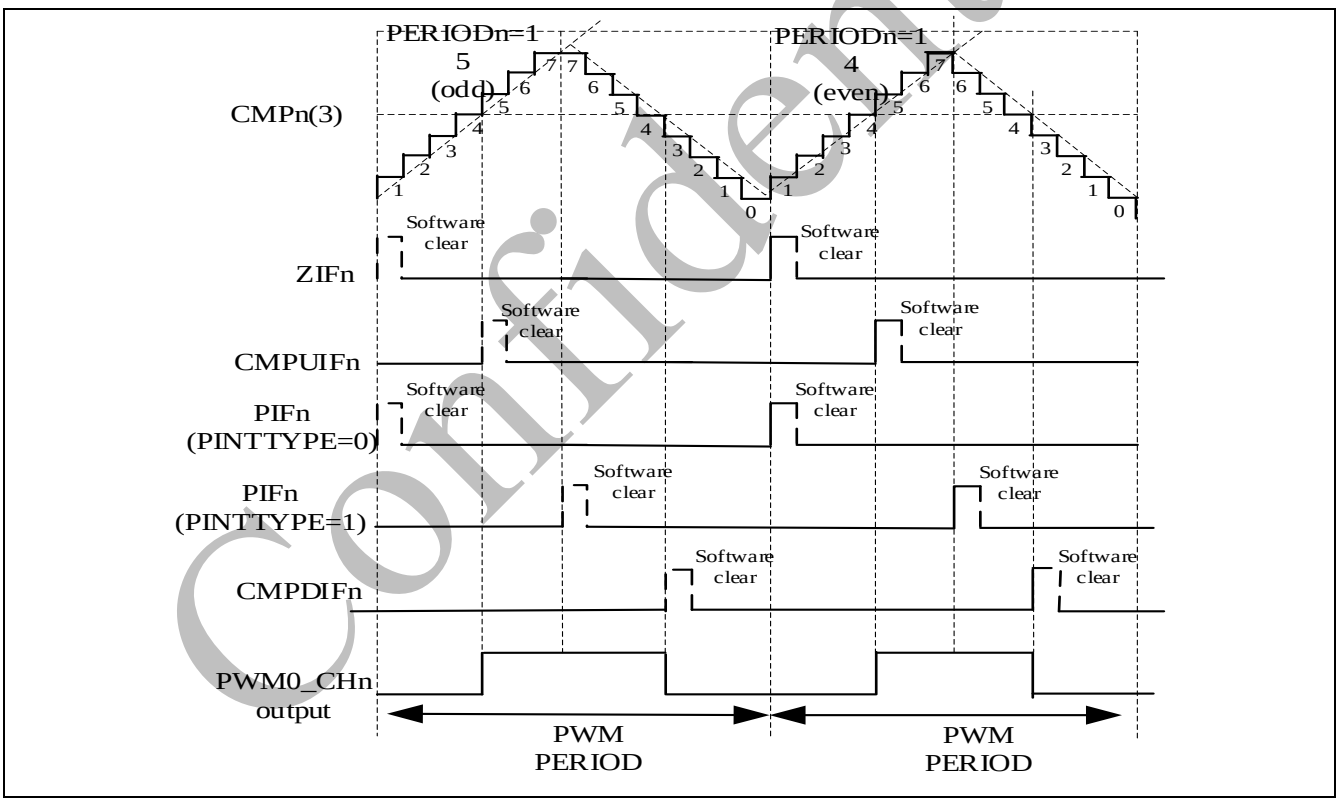


Figure 3-55 PWM Precise Center-aligned Waveform Timing Diagram

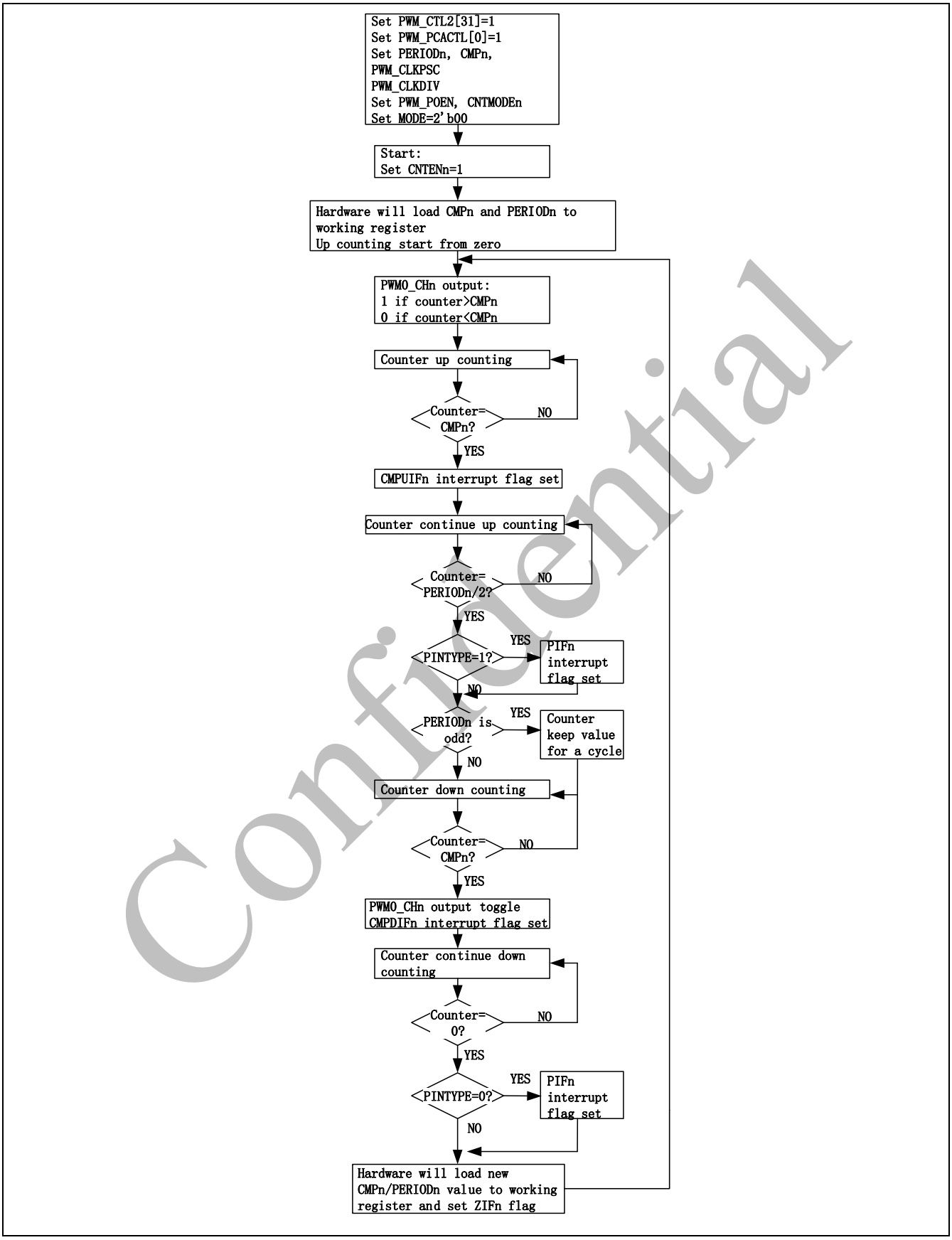


Figure 3-56 Precise Center-aligned Flow Diagram

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.

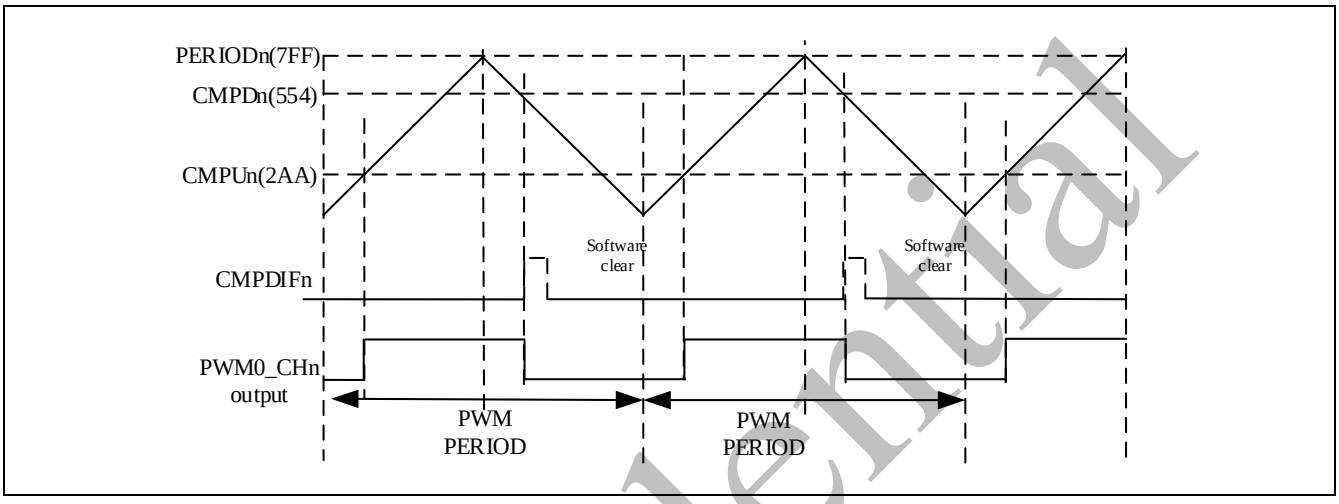


Figure 3-57 Asymmetric Mode Timing Diagram

3.11.5.2 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 3-58 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.

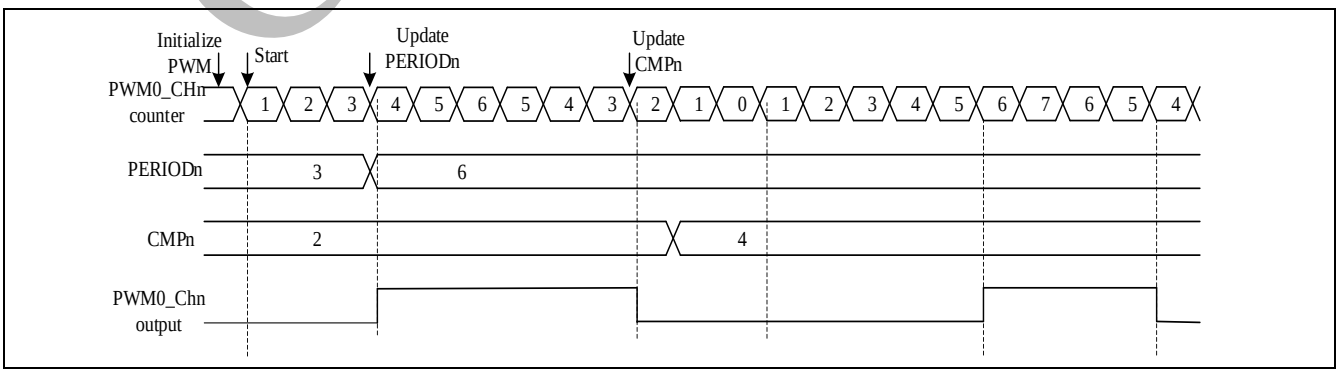


Figure 3-58 PWM Center Loading Timing Diagram

3.11.5.3 PWM Double Buffering and Auto-reload Operation

The PAN108 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. 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.

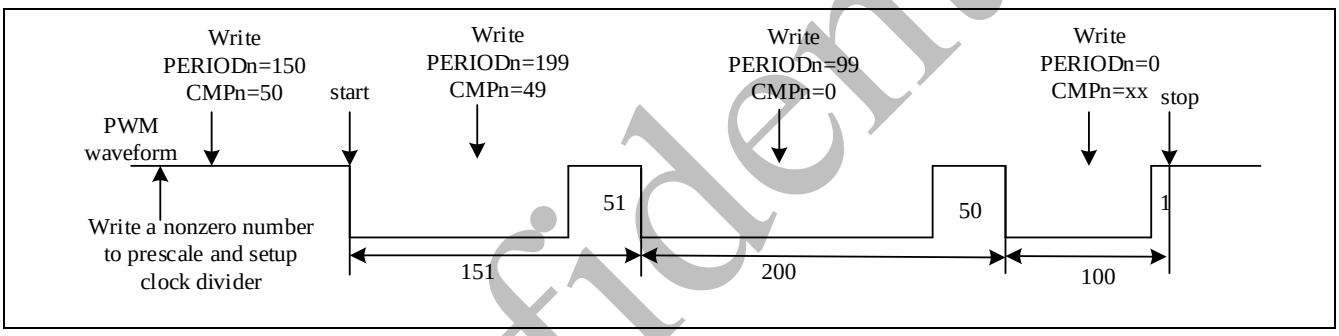


Figure 3-59 PWM Double Buffering Illustration

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.

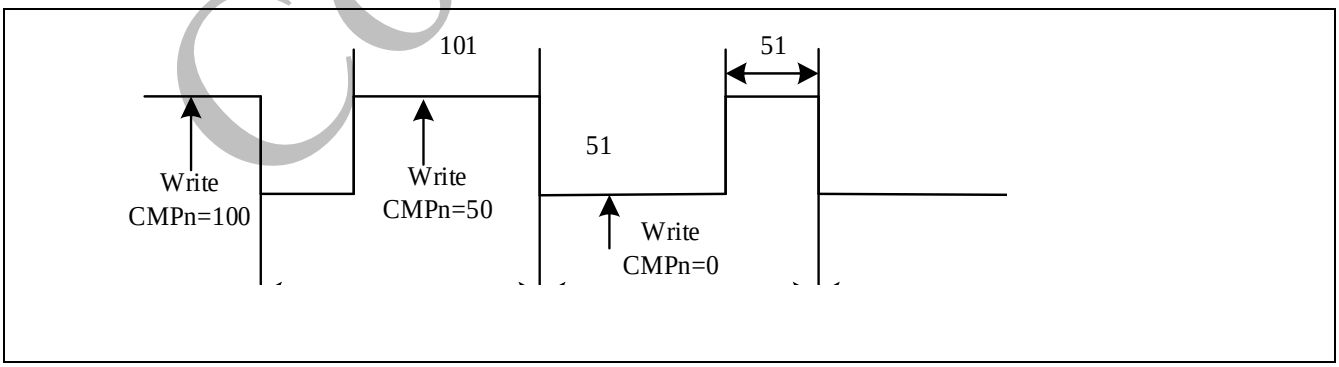


Figure 3-60 PWM Controller Output Duty Ratio

3.11.5.4 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.

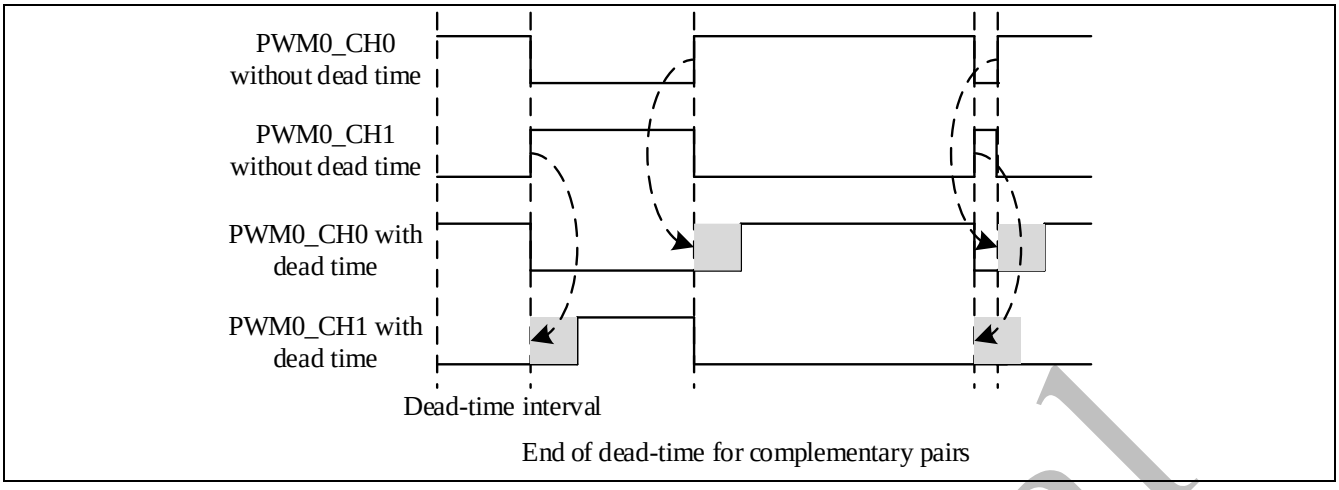


Figure 3-61 Dead-time Insertion

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.

3.11.5.5 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 3-62 shows the initial state before PWM starts with different polarity settings.

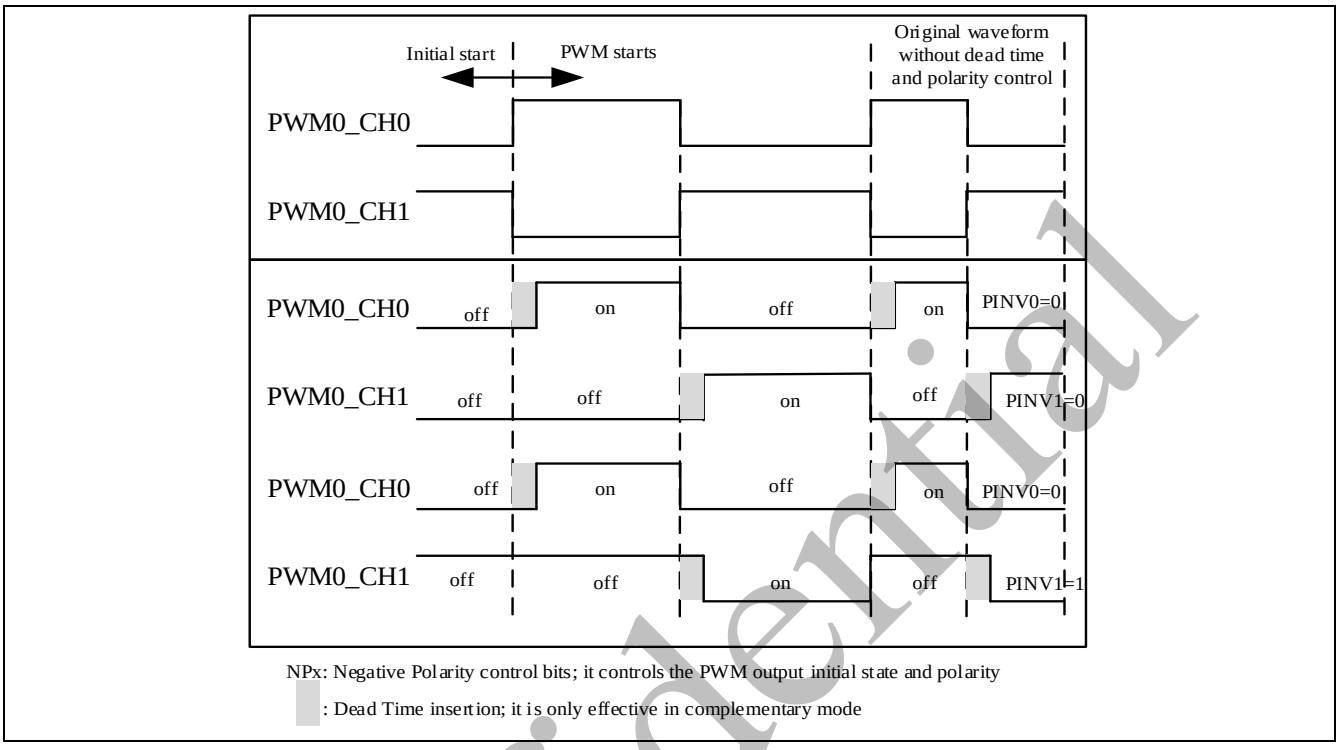


Figure 3-62 Initial State and Polarity Control with Rising Edge Dead-time Insertion

3.11.5.6 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 3-63 shows the architecture of Motor Control PWM interrupts.

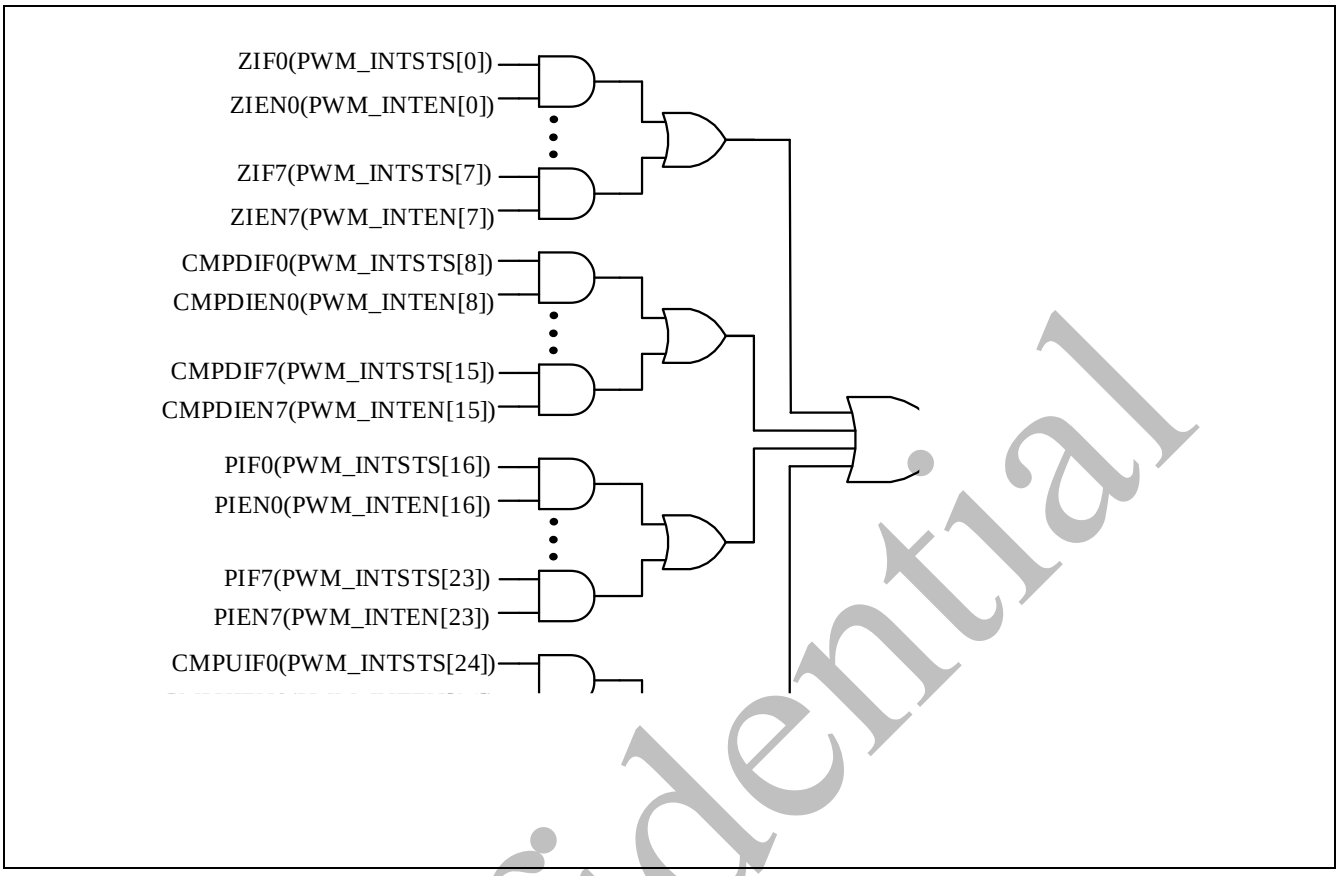


Figure 3-63 Motor Control PWM Interrupt Architecture

3.11.6 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

3.11.7 PWM Control Register Description

3.11.7.1 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.

3.11.7.2 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.

3.11.7.3 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.

3.11.7.4 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.

3.11.7.5 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.

3.11.7.6 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

3.11.7.7 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.

3.11.7.8 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.

3.11.7.9 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.

3.11.7.10 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.

3.11.7.11 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.

3.11.7.12 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.

3.11.7.13 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.

3.11.7.14 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.

3.11.7.15 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.

3.11.7.16 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.

3.11.8 Operation Steps

3.11.8.1 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, 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)

3.11.8.2 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.

3.12 Watchdog Timer (WDT)

3.12.1 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.

3.12.2 Features

- Selectable time-out interval (24 ~ 218) WDT_CLK cycles, up counter can be reset by register.
- WDT_CLK source can be configured to be APB_CLK/2048 or 32KHz clock.
- 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

3.12.3 Block Diagram

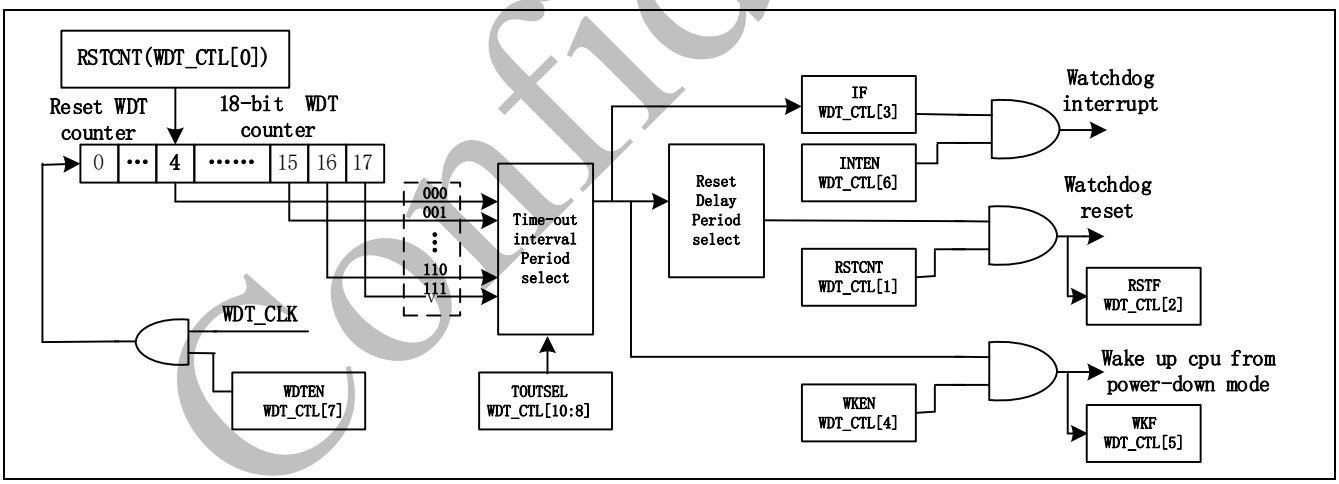


Figure 3-64 Watchdog Timer Block Diagram

3.12.4 Clock Control

The WDT clock control is shown in Figure 3-65.

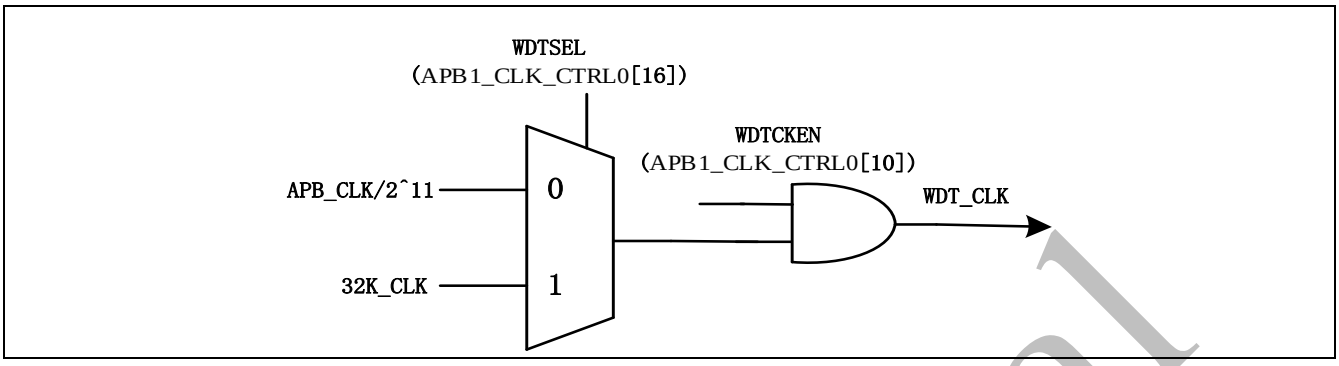


Figure 3-65 Watchdog Timer Clock Control

3.12.5 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]).

3.12.6 Functional Description

The WDT includes an 18-bit free running up counter with programmable time-out intervals. Table 3-16 shows the WDT time-out interval period selection and Figure 3-66 shows the WDT time-out interval and reset period timing.

3.12.6.1 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[10:8]). When the WDT up counter reaches the *TOUTSEL* (WDT_CTL[10:8]) setting, the WDT time-out interrupt will occur and then WDT time-out flag TOF (WDT_CTL[16]) will be set to 1 immediately.

3.12.6.2 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[10:8]). When the WDT up counter reaches the TOUTSEL (WDT_CTL[10:8]) 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.

3.12.6.3 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 18-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 3-66 Watchdog Timer Time-out Interval and Reset Period Timing. The reset signal will keep low 1 apb 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 3-16 Watchdog Timer Time-out Interval Period Selection

TOUTSEL	Time-Out Interval Period TTIS	Reset Delay Period TRSTD
000	24 * TWDT	(3/18/130/1026) * TWDT
001	26 * TWDT	(3/18/130/1026) * TWDT
010	28 * TWDT	(3/18/130/1026) * TWDT
011	210 * TWDT	(3/18/130/1026) * TWDT
100	212 * TWDT	(3/18/130/1026) * TWDT
101	214 * TWDT	(3/18/130/1026) * TWDT
110	216 * TWDT	(3/18/130/1026) * TWDT
111	218 * TWDT	(3/18/130/1026) * TWDT

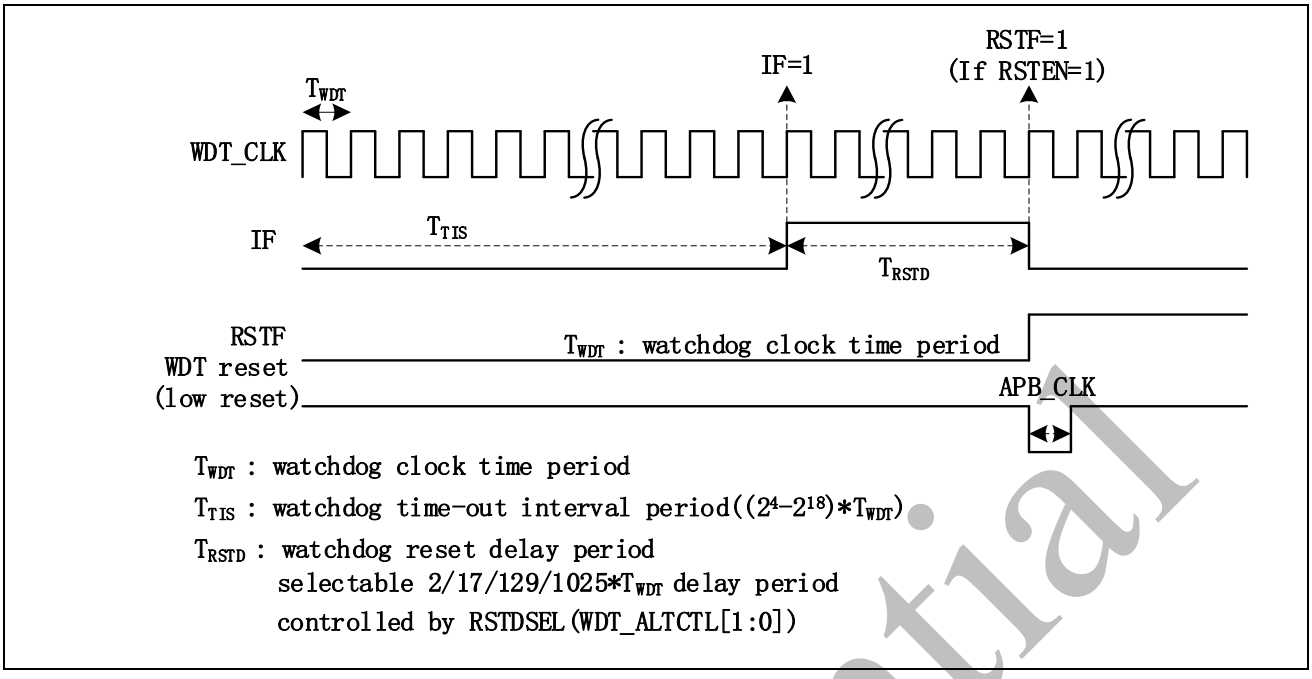


Figure 3-66 Watchdog Timer Time-out Interval and Reset Period Timing

3.12.6.4 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.

3.12.7 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_0700
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000

3.12.8 WDT Register Description

3.12.8.1 WDT Control Register (WDT_CTL)

Register	Offset	R/W	Description	Reset Value
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0700

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.
[15:11]	Reserved	Reserved.
[10:8]	TOUTSEL	WDT Time-out Interval Selection (Write Protect) These three bits select the time-out interval period for the WDT. 000 = 2^4 * WDT_CLK. 001 = 2^6 * WDT_CLK. 010 = 2^8 * WDT_CLK. 011 = 2^10 * WDT_CLK. 100 = 2^12 * WDT_CLK. 101 = 2^14 * WDT_CLK. 110 = 2^16 * WDT_CLK. 111 = 2^18 * 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 18-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.

3.12.8.2 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.

3.13 Window Watchdog Timer (WWDT)

3.13.1 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.

3.13.2 Feature

- 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

3.13.3 Block Diagram

The WWDT block diagram is shown in Figure 3-67.

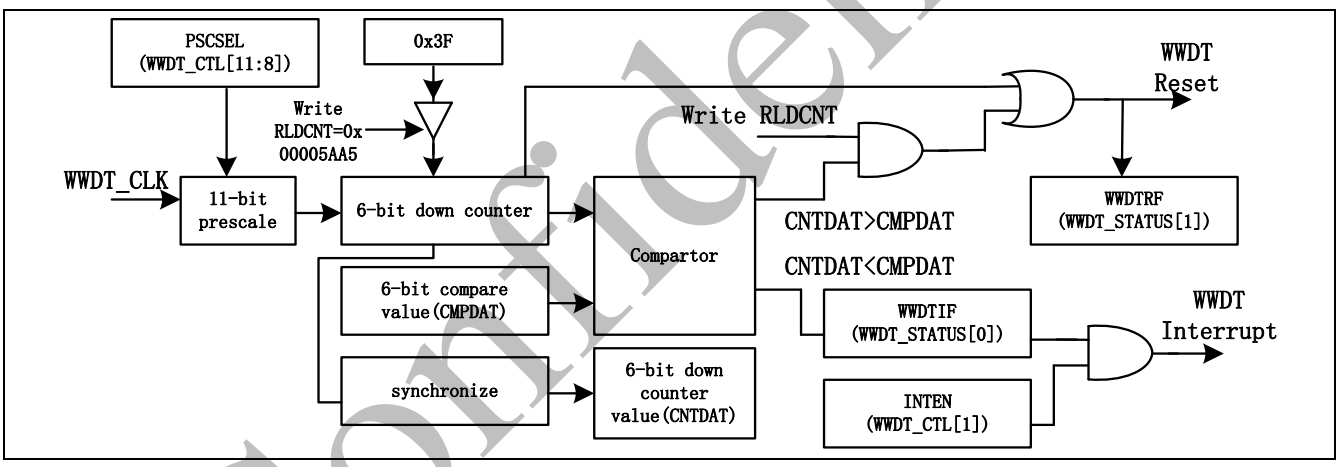


Figure 3-67 WWDT Block Diagram

3.13.4 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 3-68.

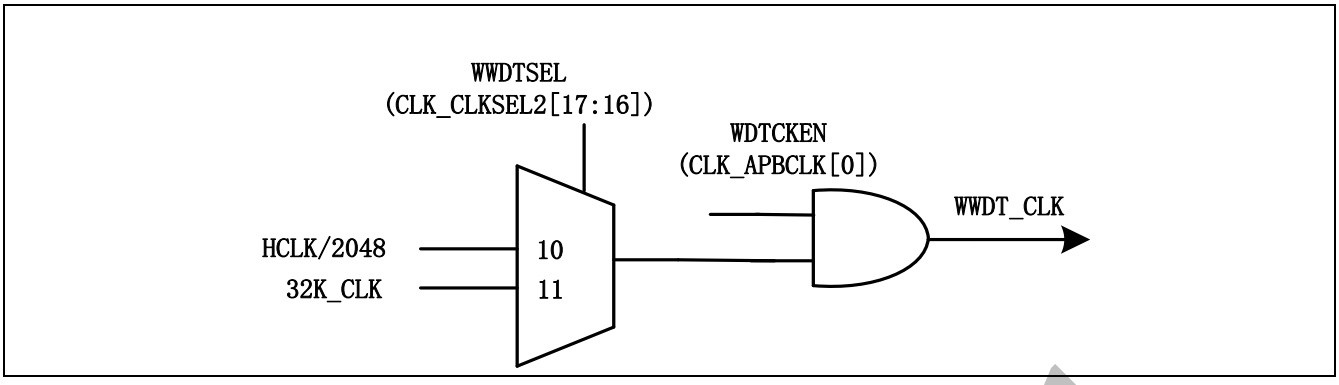


Figure 3-68 WWDT Clock Control

3.13.5 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 3-17.

Table 3-17 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

3.13.5.1 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.

3.13.5.2 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.

3.13.5.3 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.

3.13.5.4 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.

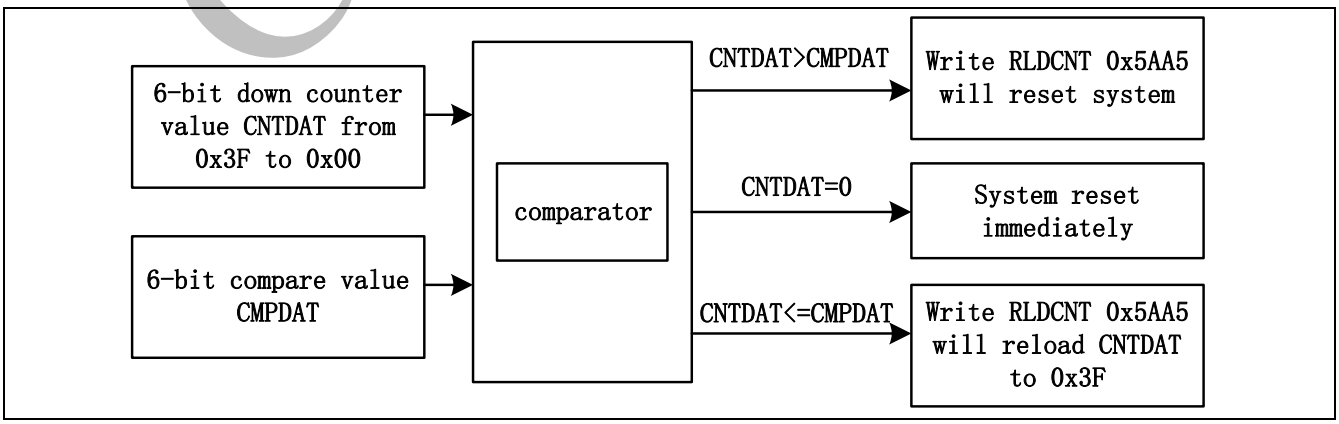


Figure 3-69 WWDT Reset and Reload Behavior

3.13.5.5 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 3-18 CMPDAT Setting Limitation

PSCSEL	Prescale	Value
0000	1	0x03 ~ 0x3F
0001	2	0x02 ~ 0x3F
others	others	0x00 ~ 0x3F

3.13.6 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

3.13.7 WWDT Register Description

3.13.7.1 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.

3.13.7.2 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.

3.13.7.3 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.

3.13.7.4 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.

3.14 I2C Serial Interface Controller (I2C)

3.14.1 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 1.6 Mb/s).

Depending on specific device implementation DMA capability can be available for reduced CPU overload.

3.14.2 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;DAT)
- Multiple masters arbitration

3.14.3 Block Diagram

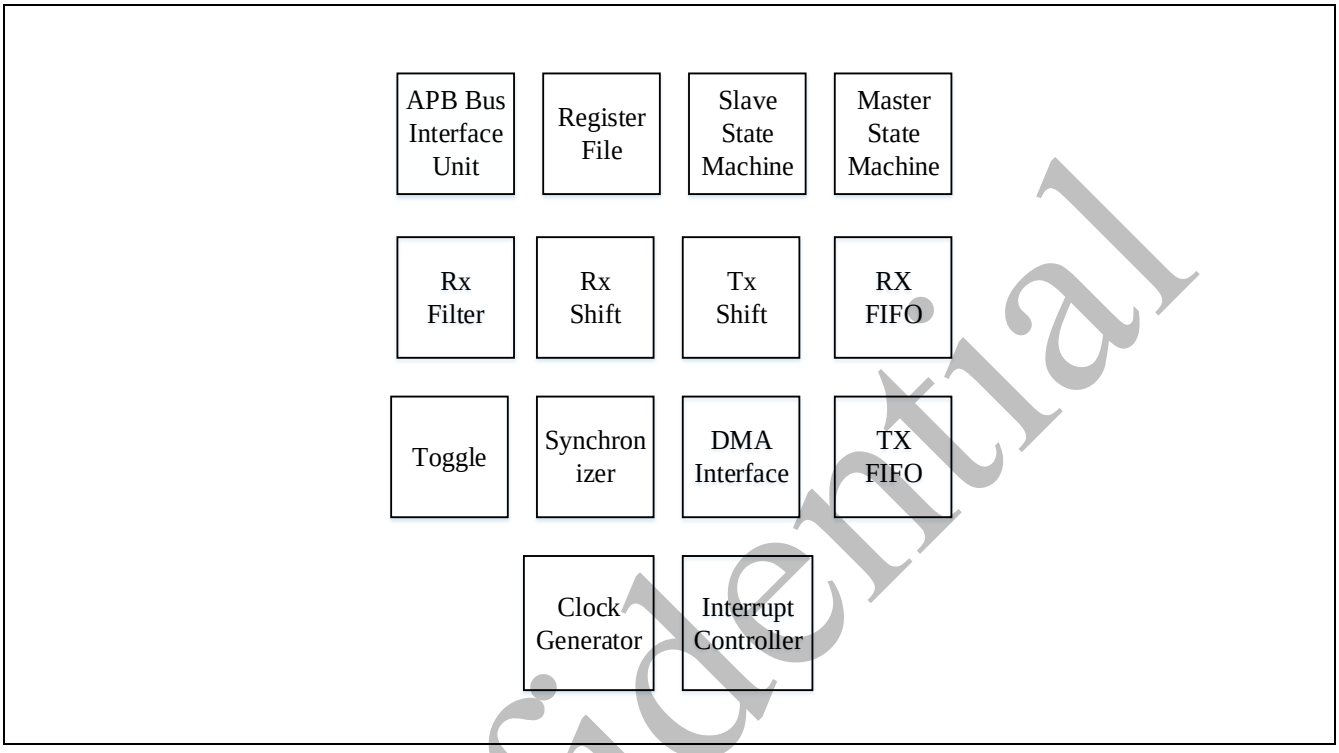


Figure 3-70 I2C Controller Block Diagram

3.14.4 Functional Description

This chapter describes the functional behavior of I2C in more detail.

3.14.4.1 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 3-71.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 3-72.

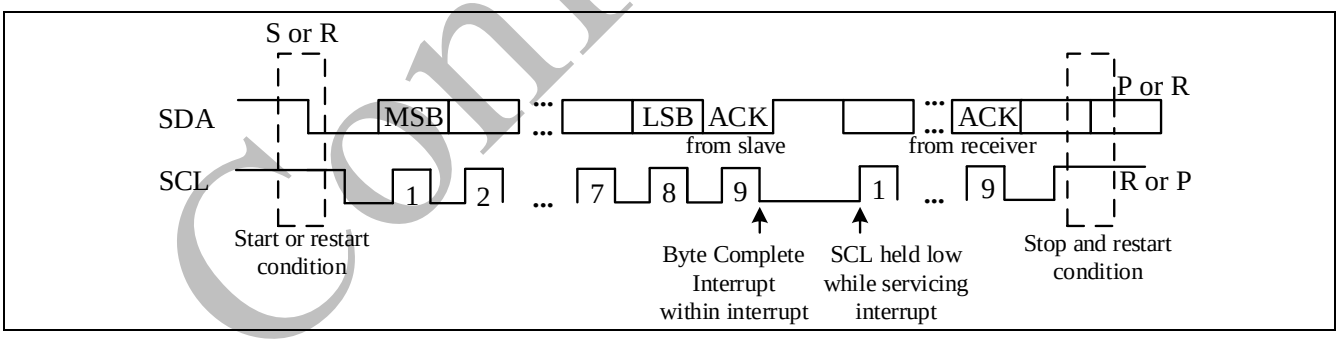


Figure 3-71 Data transfer on the I2C Bus for Master Transmitter

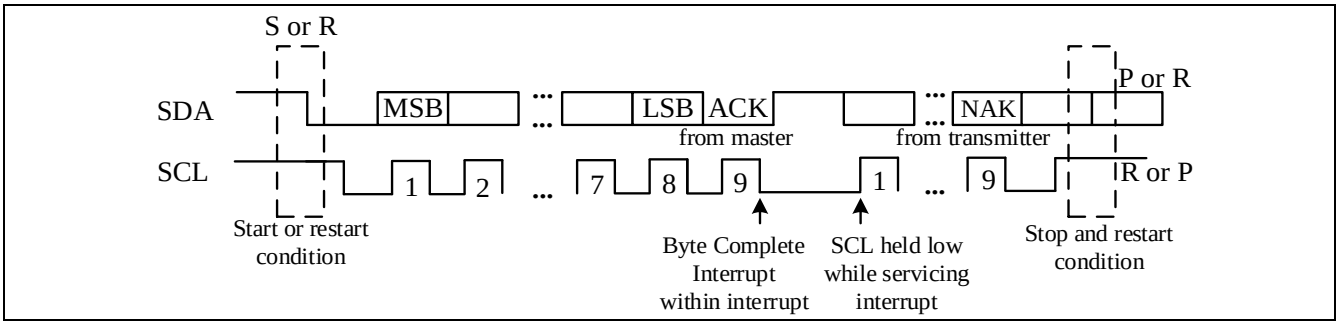


Figure 3-72 Data transfer on the I2C Bus for Master Receiver

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 IC_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 IC_ENABLE 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, IC_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, IC_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.

3.14.4.2 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 3-73 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.

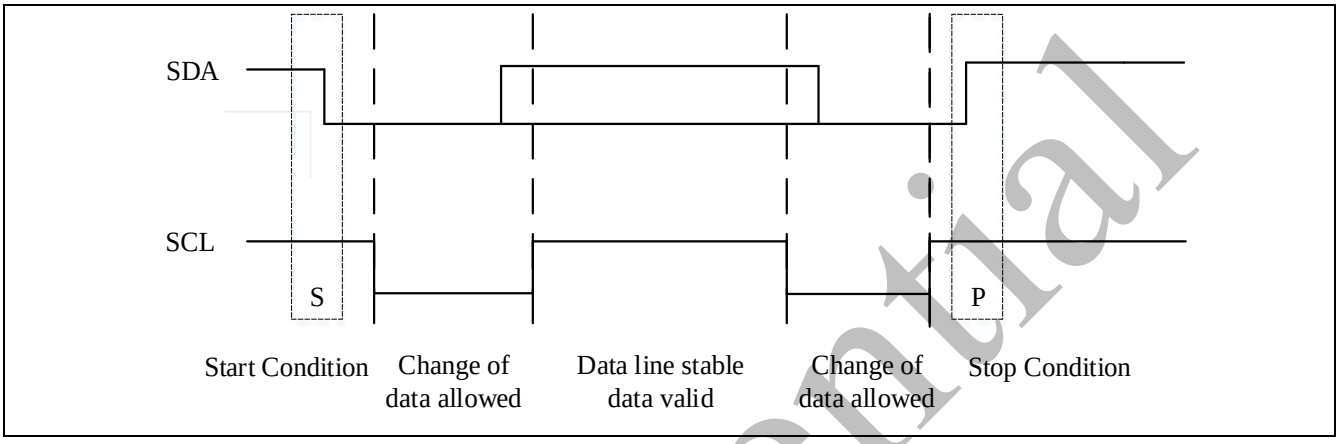


Figure 3-73 START and STOP Condition

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 3-74. 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.

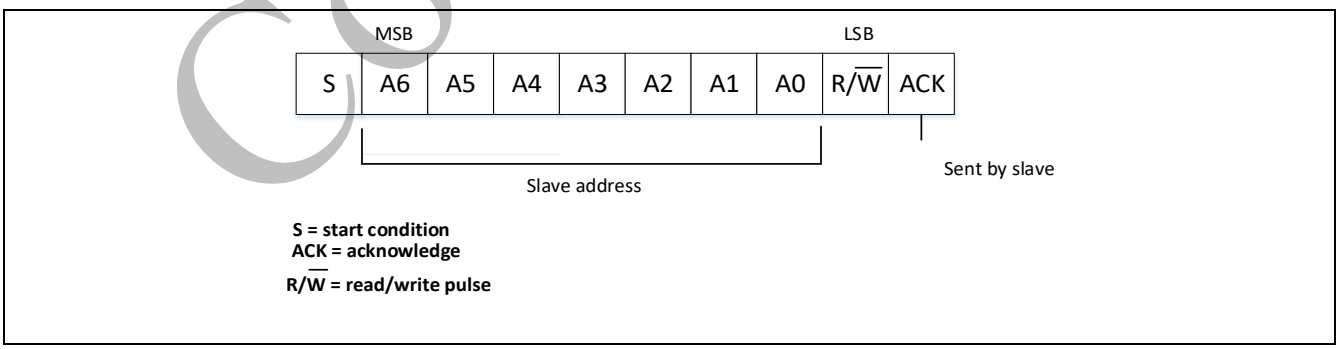


Figure 3-74 7-bit Address Format

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 3-75 shows the 10-bit address format.

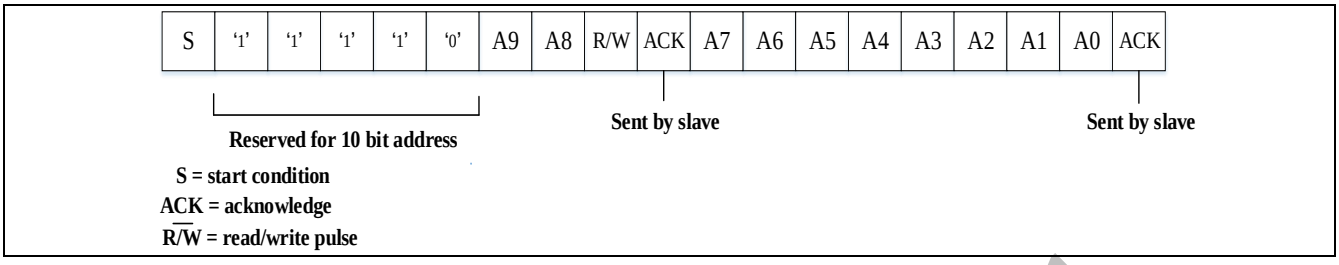


Figure 3-75 10-bit Address Format

Table 3-19 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 3-76, then the slave-receiver responds to the master-transmitter with an acknowledge pulse after every byte of data is received.

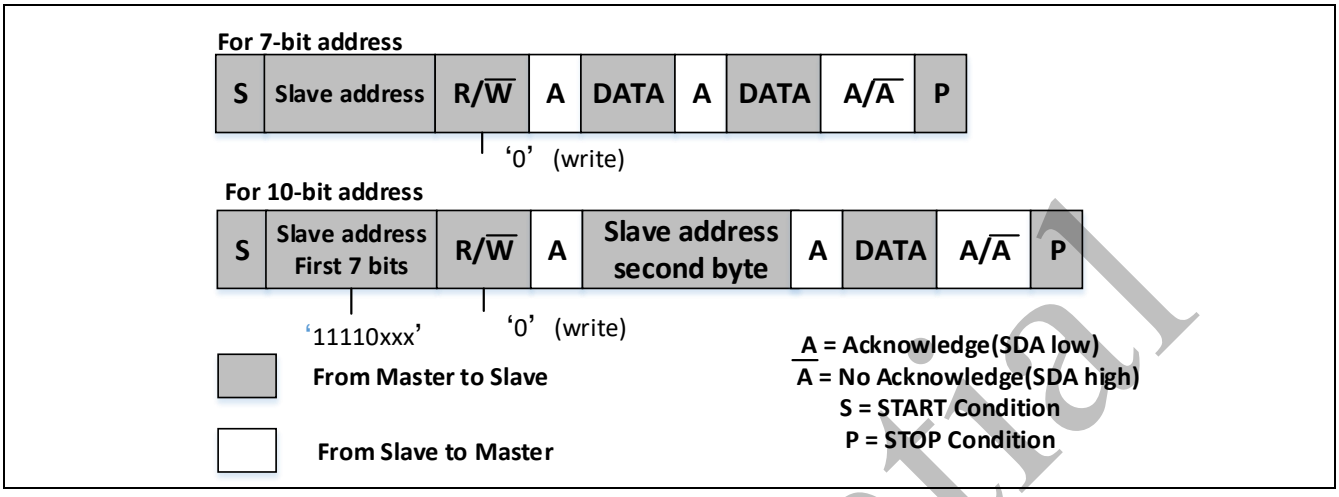


Figure 3-76 Master-Transmitter Protocol

Master-Receiver and Slave-Transmitter

If the master is receiving data as shown in Figure 3-77, 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.

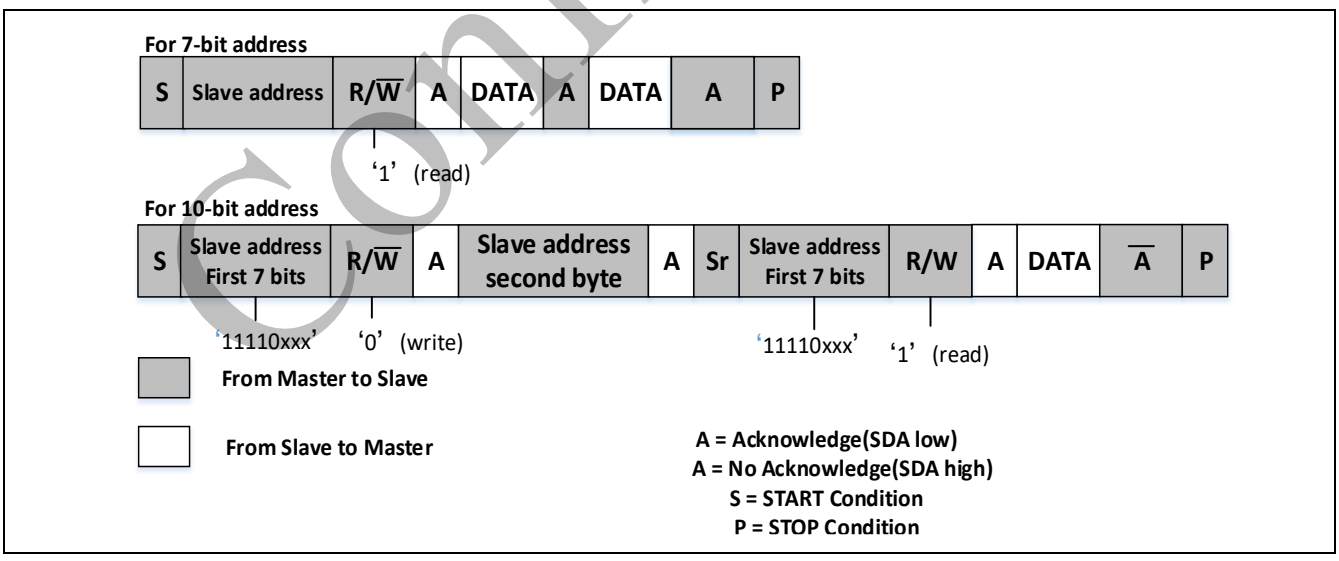


Figure 3-77 Master-Receiver Protocol

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 3-78. This allows the processor that is polling the bus to under-sample the address phase until 0 is detected. Once the microcontroller detects a 0, it switches from the under sampling rate to the correct rate of the master.

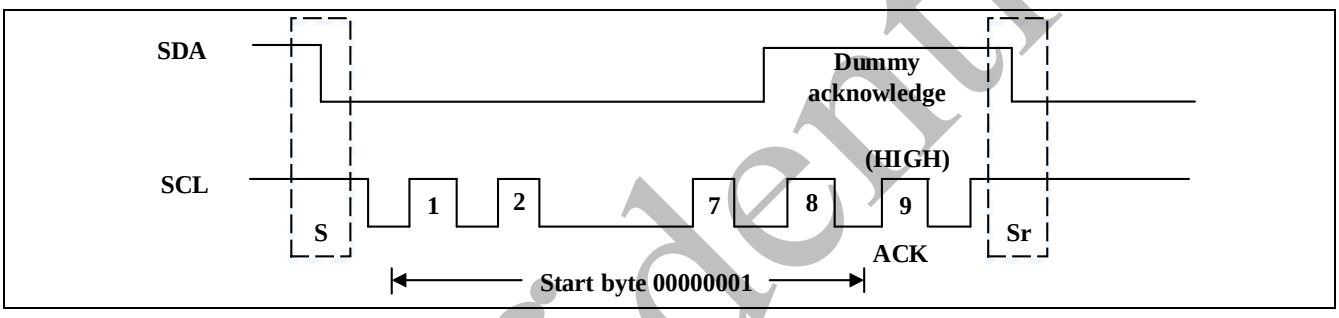


Figure 3-78 START BYTE Transfer

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.

3.14.4.3 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 IC_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 3-79 IC_DATA_CMD Register if IC_EMPTYFIFO_HOLD_MASTER_EN = 1

3.14.4.4 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 microcontroller) 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 3-80 illustrates the timing of when two masters are arbitrating on the bus.

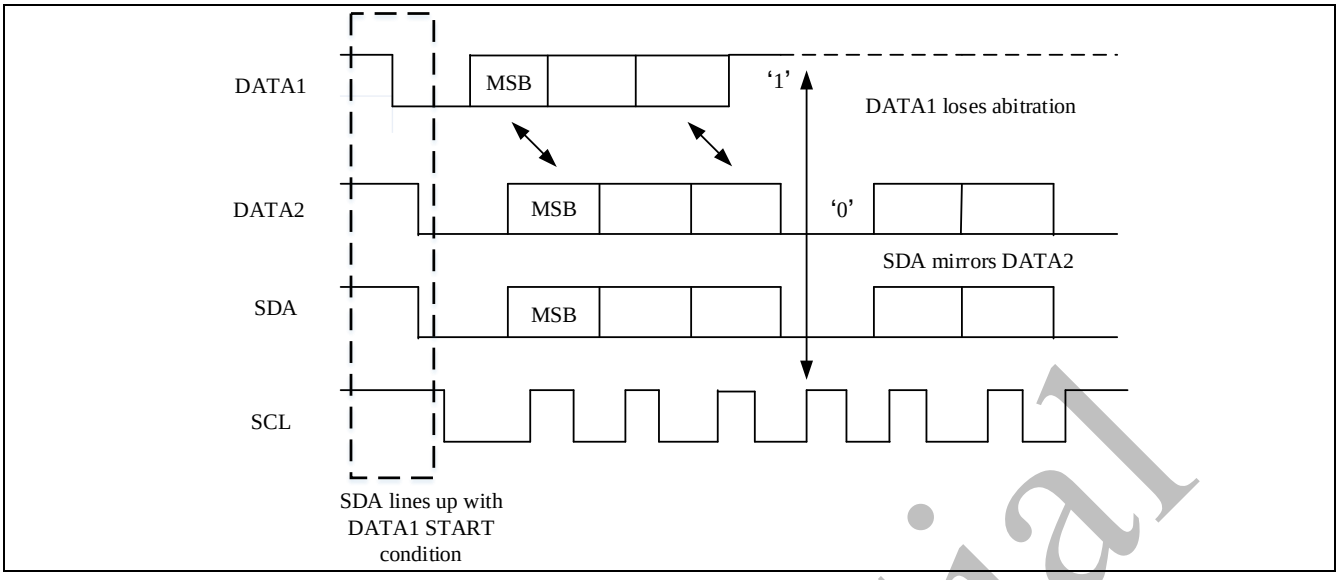


Figure 3-80 Multiple Master Arbitration

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, IC_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.

3.14.4.5 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 3-81. Optionally, slaves may hold the SCL line low to slow down the timing on the I2C bus.

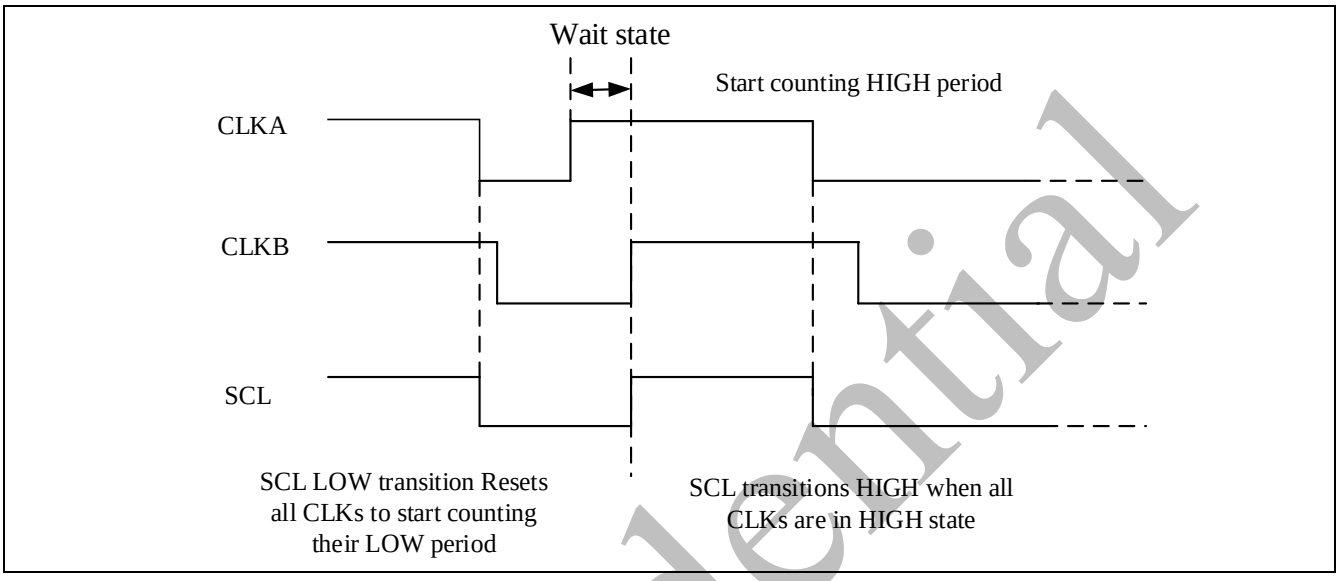


Figure 3-81 Multi-Master Clock Synchronization

3.14.4.6 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 IC_ENABLE register.
2. Write to the IC_SAR register (bits 9:0) to set the slave address. This is the address to which the I2C responds.
3. Write to the IC_CON 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 (IC_SLAVE_DISABLE) and a '0' to bit 0 (MASTER_MODE).
4. Enable the I2C by writing a '1' in bit 0 of the IC_ENABLE register.

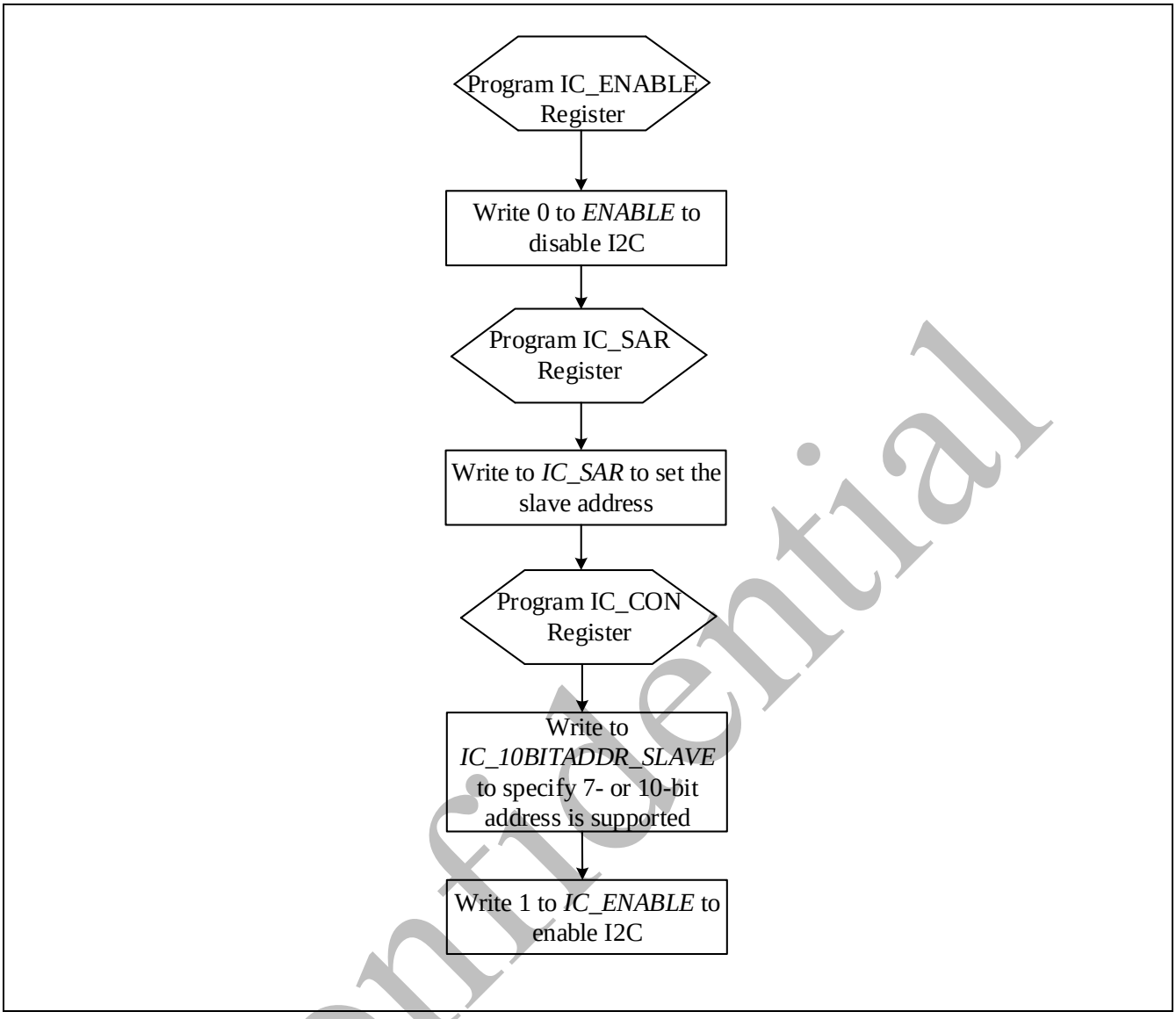


Figure 3-82 Initial Configuration

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 IC_SAR 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 IC_RAW_INTR_STAT 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 IC_INTR_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 IC_RAW_INTR_STAT register.

a. Reads that indicate IC_RAW_INTR_STAT[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 IC_RAW_INTR_STAT register) to flush the old data from the TX FIFO.

   If the TX_ABRT interrupt has been masked, due to of IC_INTR_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 IC_RAW_INTR_STAT 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 IC_RAW_INTR_STAT[5] register.
5. Software writes to the IC_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 IC_RAW_INTR_STAT register before proceeding. If the RD_REQ and/or TX_ABRT interrupts have been masked, then clearing of the IC_RAW_INTR_STAT 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 IC_SAR 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 (IC_RAW_INTR_STAT[2] register). If the RX_FULL interrupt has been masked, due to setting IC_INTR_MASK[2] register to 0 or setting IC_TX_TL to a value larger than 0, then it is recommended that a timing routine be implemented for periodic reads of the IC_STATUS register. Reads of the IC_STATUS register, with bit 3 (RFNE) 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 IC_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 IC_INTR_STAT register being set to 0, then it is recommended that a timing routine be used to activate periodic reads of the IC_RAW_INTR_STAT register. Reads of IC_RAW_INTR_STAT 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 IC_10BITADDR_MASTER bit is set (either bit 4 of IC_CON register or bit 12 of IC_TAR 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 IC_ENABLE register.
2. Write to the IC_CON 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 IC_TAR 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 IC_10BITADDR_MASTER bit field (bit 12).
4. Only applicable for high-speed mode transfers. Write to the IC_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 IC_ENABLE register.
6. Now write the transfer direction and data to be sent to the IC_DATA_CMD register. If the IC_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 IC_TAR or IC_10BITADDR_MASTER Update

The I2C supports dynamic updating of the IC_TAR (bits 9:0) and IC_10BITADDR_MASTER (bit 12) bit fields of the IC_TAR register. In order to perform a dynamic update of the IC_TAR register, the I2C_DYNAMIC_TAR_UPDATE configuration parameter must be set to Yes (1). You can dynamically write to the IC_TAR 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 IC_TAR should be re-programmed only if the following conditions are met:

- I2C is not enabled (IC_ENABLE[0]=0);

OR

- I2C is enabled (IC_ENABLE[0]=1); AND
- I2C is NOT engaged in any Master (tx, rx) operation (IC_STATUS[5]=0);

AND

- I2C is enabled to operate in Master mode (IC_CON[0]=1);

AND

- there are NO entries in the Tx FIFO (IC_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 (IC_DATA_CMD). The CMD 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 IC_DATA_CMD register, and a 1 should be written to the CMD 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 IC_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 IC_ENABLE_STATUS is added to allow software to unambiguously determine when the hardware has completely shut down in response to bit 0 of the IC_ENABLE 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 IC_ENABLE register to 0.
6. Read the IC_ENABLE_STATUS register and test the IC_EN bit (bit 0). Increment POLL_COUNT by one. If POLL_COUNT ≥ MAX_T_POLL_COUNT, exit with the relevant error code.
7. If IC_ENABLE_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 IC_ENABLE 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 (IC_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 IC_ENABLE register (ABORT) to 1.
4. Wait for the M_TX_ABRT interrupt.

5. Read the IC_TX_ABRT_SOURCE register to identify the source as ABRT_USER_ABRT.

3.14.4.7 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 ic_clk cycles before they are sampled internally. There is one separate counter for each signal (SCL and SDA). The number of ic_clk cycles can be programmed by the user and should be calculated taking into account the frequency of ic_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 3-83 illustrates the behavior described above.

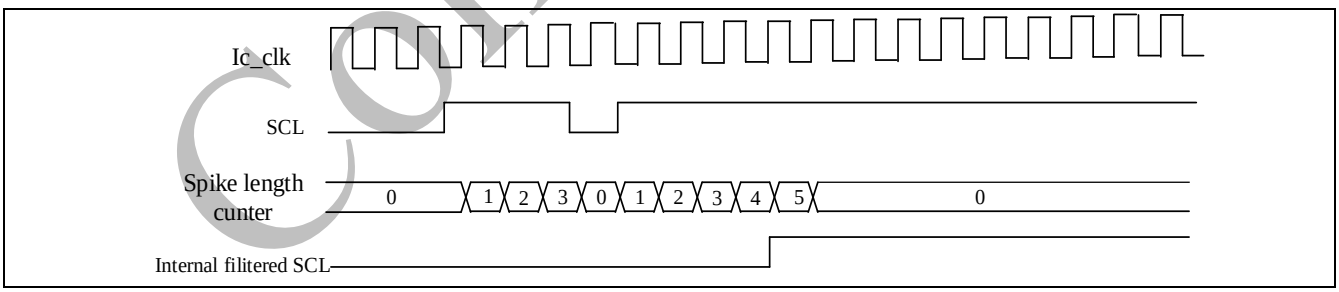


Figure 3-83 Spike Suppression Example

The count limit value used in this example is 5 and was calculated for a 10 ns ic_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 IC_FS_SPKLEN holds the maximum spike length for SS and FS/FM+ modes

- Register IC_HS_SPKLEN 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.

3.14.4.8 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. ic_clk frequency should greater than or equal to 32 MHz
3. Program the IC_CON register [2:1] = 2'b10 for fast mode or fast mode plus.
4. Program IC_FS_SCL_LCNT and IC_FS_SCL_HCNT registers to meet the fast mode plus SCL.
5. Program the IC_FS_SPKLEN register to suppress the maximum spike of 50ns.
6. Program the IC_SDA_SETUP register to meet the minimum data setup time (tSU; DAT)

3.14.4.9 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:

- IC_SS_SCL_LCNT and IC_FS_SCL_LCNT register values must be larger than IC_FS_SPKLEN + 7.
- IC_SS_SCL_HCNT and IC_FS_SCL_HCNT register values must be larger than IC_FS_SPKLEN + 5.
- If the component is programmed to support HS, IC_HS_SCL_LCNT register value must be larger than IC_HS_SPKLEN + 7.
- If the component is programmed to support HS, IC_HS_SCL_HCNT register value must be larger than IC_HS_SPKLEN + 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:
    - IC_FS_SPKLEN if the component is operating in SS or FS
    - IC_HS_SPKLEN if the component is operating in HS.
  - This filtering includes metastability 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 ic_clk cycles is incurred. Consequently, the minimum SCL low time of which the I2C is capable is nine (9) ic_clk periods (7 + 1 + 1), while the minimum SCL high time is thirteen (13) ic_clk periods (6 + 1 + 3 + 3).

3.14.4.10 SDA Hold Time

The I2C protocol specification requires 300ns of hold time on the SDA signal (tHD;DAT) 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 (IC_SDA_HOLD) to enable dynamic adjustment of the SDA hold-time. The IC_SDA_HOLD register can be programmed only when the I2C is disabled (IC_ENABLE[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.

IC_SDA_RX_HOLD can be used to alter the internal hold time which I2C applies to the incoming SDA line. Each value in the IC_SDA_RX_HOLD register represents a unit of one ic_clk period. The minimum value of IC_SDA_RX_HOLD 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 3-84 shows the I2C as receiver with IC_SDA_RX_HOLD programmed to greater than or equal to 3.

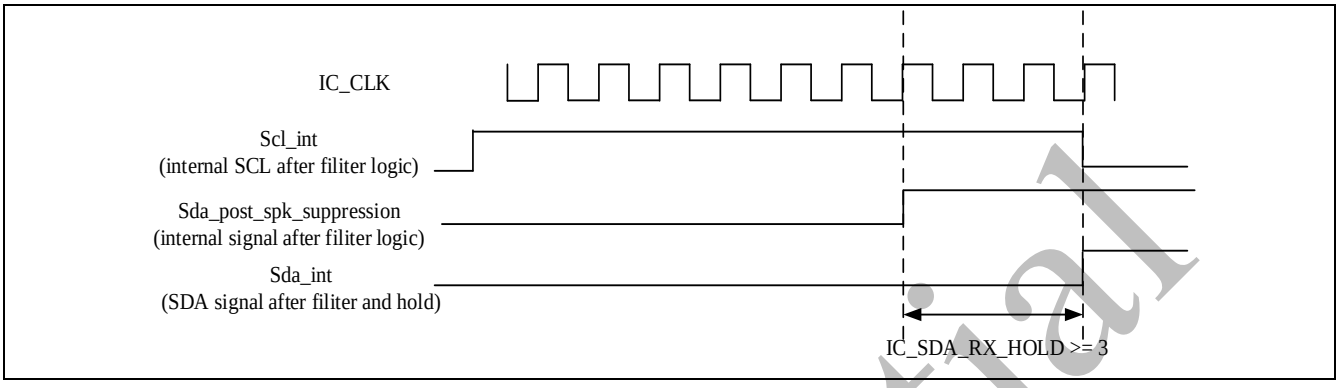


Figure 3-84 I2C receiver with IC_SDA_RX_HOLD

If IC_SDA_RX_HOLD is greater than 3, I2C does not hold SDA beyond 3 ic_clk cycles, because SCL goes LOW internally.

The maximum values of IC_SDA_RX_HOLD that can be programmed in the register for the respective speed modes are derived from the equations show in Table 3-20.

Table 3-20 Maximum Values for IC_SDA_RX_HOLD

Speed Mode	Maximum IC_SDA_RX_HOLD Value
Standard Mode	IC_SS_SCL_HCNT – IC_FS_SPKLEN – 3
Fast Mode or Fast Mode Plus	IC_FS_SCL_HCNT – IC_FS_SPKLEN – 3
High Speed	Min {IC_FS_SCL_HCNT – IC_FS_SPKLEN – 3 , IC_HS_SCL_LCNT – IC_HS_SPKLEN – 3}

SDA Hold Timings in Transmitter

The IC_SDA_TX_HOLD register can be used to alter the timing of the generated SDA signal by the I2C. Each value in the IC_SDA_TX_HOLD register represents a unit of one ic_clk period. When the I2C is operating in Master Mode, the minimum tHD:DAT timing is one ic_clk period. Therefore even when IC_SDA_TX_HOLD has a value of zero, the I2C will drive SDA one ic_clk cycle after driving SCL to logic 0. For all other values of IC_SDA_TX_HOLD, the following is true:

- Drive on SDA occurs IC_SDA_TX_HOLD ic_clk cycles after driving SCL to logic 0
  - When the I2C is operating in Slave Mode, the minimum tHD:DAT timing is SPKLEN + 7 ic_clk periods, where SPKLEN is:
- IC_FS_SPKLEN if the component is operating in standard mode, fast mode, or fast mode

plus

- IC_HS_SPKLEN if the component is operating in high speed mode
  - This delay allows for synchronization and spike suppression on the SCL sample. Therefore, even when IC_SDA_TX_HOLD has a value less than SPKLEN + 7, the I2C drives SDA SPKLEN + 7 ic_clk cycles after SCL has transitioned to logic 0. For all other values of IC_SDA_TX_HOLD, the following is true:
- Drive on SDA occurs IC_SDA_TX_HOLD ic_clk cycles after SCL has transitioned to logic 0
  - Figure 3-85 shows the tHD:DAT timing generated by the I2C operating in Master Mode when IC_SDA_TX_HOLD = 3.

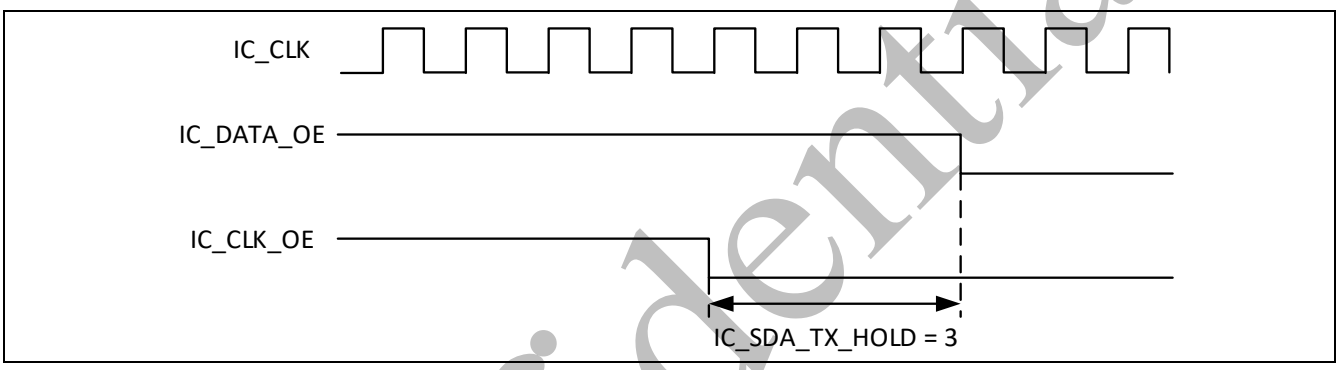


Figure 3-85 I2C Master Implementing tHD;DAT with IC_SDA_HOLD = 3

3.14.4.11 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 3-86.

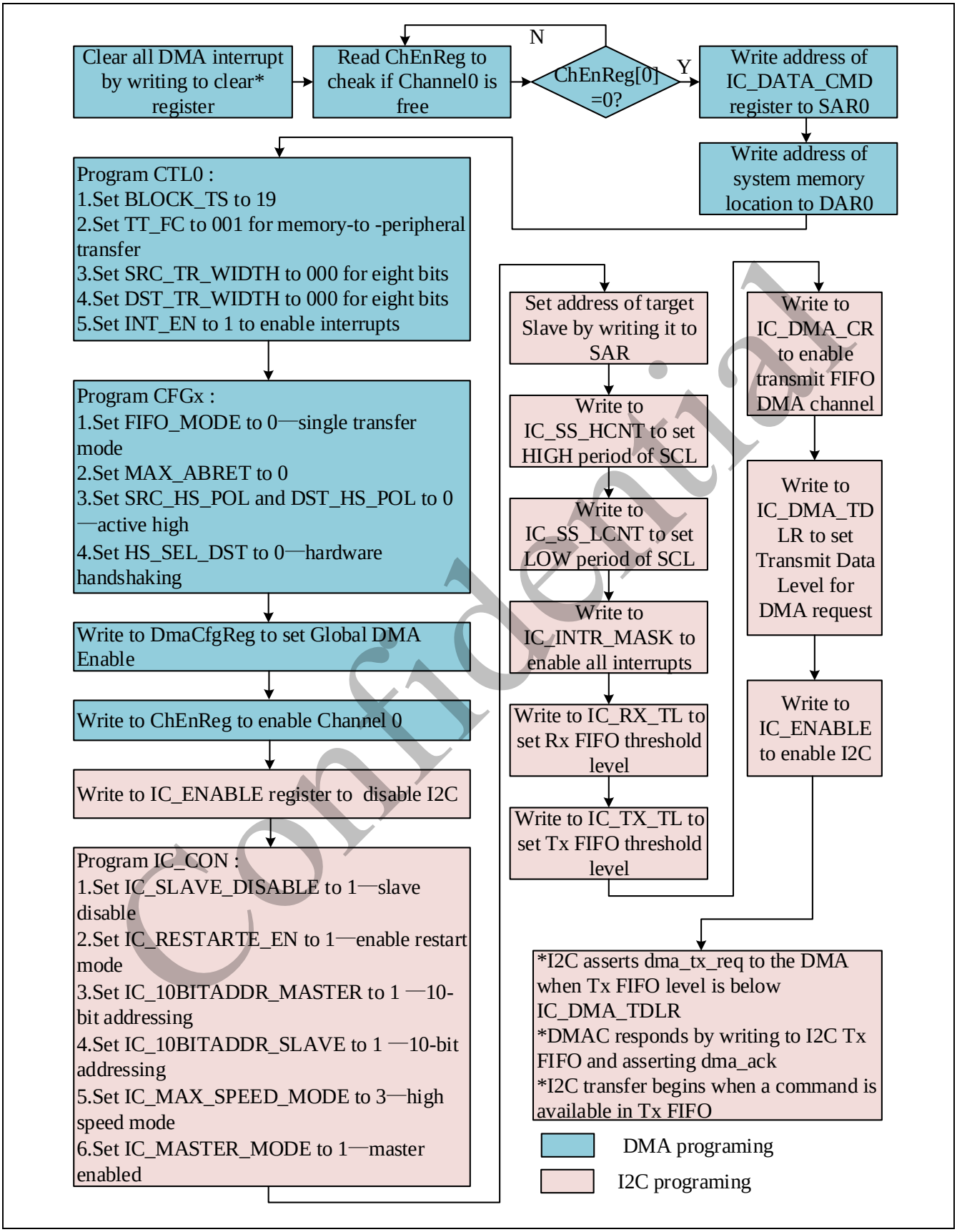


Figure 3-86 Flowchart for DMA and I2C Programing

3.14.5 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				
IC_CON	I2C_BA+ 0×00	R/W	I2C Control Register	0×0000_007F
IC_TAR	I2C_BA+ 0×04	R/W	I2C Target Address Register	0×0000_1055
IC_SAR	I2C_BA+0×08	R/W	I2C Slave Address Register	0×0000_0055
IC_HS_MADDR	I2C_BA+0×0C	R/W	I2C High Speed Master Mode Code Address Register	0×0000_0001
IC_DATA_CMD	I2C_BA+0×10	R/W	I2C Rx/Tx Data Buffer and Command Register	0×0000_0000
IC_SS_SCL_HCNT	I2C_BA+0×14	R/W	Standard Speed I2C Clock SCL High Count Register	0×0000_0190
IC_SS_SCL_LCNT	I2C _BA+0×18	R/W	Standard Speed I2C Clock SCL Low Count Register	0×0000_01D6
IC_FS_SCL_HCNT	I2C_BA+0×1C	R/W	Fast Speed I2C Clock SCL High Count Register	0×0000_003C
IC_FS_SCL_LCNT	I2C_BA+0×20	R/W	Fast Speed I2C Clock SCL Low Count Register	0×0000_0082
IC_HS_SCL_HCNT	I2C_BA+0×24	R/W	High Speed I2C Clock SCL High Count Register	0×0000_0006
IC_HS_SCL_LCNT	I2C_BA+0×28	R/W	High Speed I2C Clock SCL Low Count Register	0×0000_0010
IC_INTR_STAT	I2C_BA+0×2C	R	I2C Interrupt Status Register	0×0000_0000
IC_INTR_MASK	I2C_BA+0×30	R/W	I2C Interrupt Mask Register	0×0000_08FF
IC_RAW_INTR_STAT	I2C_BA+0×34	R	I2C Raw Interrupt Status Register	0×0000_0000
IC_RX_TL	I2C_BA+0×38	R/W	I2C Receive FIFO Threshold Register	0×0000_0000
IC_TX_TL	I2C_BA+0×3C	R/W	I2C Transmit FIFO Threshold Register	0×0000_0000
IC_CLR_INTR	I2C_BA+ 0×40	R	Clear Combined and Individual Interrupt Register	0×0000_0000
IC_CLR_RX_UNDER	I2C_BA+0×44	R	Clear RX_UNDER Interrupt	0×0000_0000
IC_CLR_RX_OVER	I2C_BA+0×48	R	Clear RX_OVER Interrupt	0×0000_0000
IC_CLR_TX_OVER	I2C_BA+0×4C	R	Clear TX_OVER Interrupt	0×0000_0000
IC_CLR_RD_REQ	I2C_BA+0×50	R	Clear RD_REQ Interrupt	0×0000_0000
IC_CLR_TX_ABRT	I2C_BA+0×54	R	Clear TX_ABRT Interrupt	0×0000_0000
IC_CLR_RX_DONE	I2C_BA+0×58	R	Clear RX_DONE Interrupt	0×0000_0000
IC_CLR_ACTIVITY	I2C_BA+0×5C	R	Clear ACTIVITY Interrupt	0×0000_0000
IC_CLR_STOP_DET	I2C_BA+0×60	R	Clear STOP_DET Interrupt	0×0000_0000
IC_CLR_START_DET	I2C_BA+0×64	R	Clear START_DET Interrupt	0×0000_0000
IC_CLR_GEN_CALL	I2C_BA+0×68	R	Clear GEN_CALL Interrupt	0×0000_0000
IC_ENABLE	I2C_BA+0×6C	R/W	I2C Enable Register	0×0000_0000
IC_STATUS	I2C_BA+0×70	R	I2C Status Register	0×0000_0006

IC_TXFLR	I2C_BA+0×74	R	I2C Transmit Abort Status Register	0×0000_0000
IC_RXFLR	I2C_BA+0×78	R	Receive FIFO Level Register	0×0000_0000
IC_SDA_HOLD	I2C_BA+0×7C	R/W	SDA Hold Time Length Register	0×0000_0001
IC_TX_ABRT_SOURCE	I2C_BA+0×80	R	I2C Transmit Abort Status Register	0×0000_0000
IC_SLV_DATA_NACK_ONLY	I2C_BA+0×84	R/W	Generate SLV_DATA_NACK Register	0×0000_0000
IC_DMA_CR	I2C_BA+0×88	R/W	DMA Control Register for transmit and receive	0×0000_0000
IC_DMA_TDLR	I2C_BA+0×8C	R/W	DMA Transmit Data Level	0×0000_0000
IC_DMA_RDLR	I2C_BA+0×90	R/W	DMA Receive Data Level	0×0000_0000
IC_SDA_SETUP	I2C_BA+0×94	R/W	I2C SDA Setup Register	0×0000_0064
IC_ACK_GENERAL_CALL	I2C_BA+0×98	R/W	I2C ACK General Call Register	0×0000_0001
IC_ENABLE_STATUS	I2C_BA+0×9C	R	I2C Enable Status Register	0×0000_0000
IC_FS_SPKLEN	I2C_BA+0×A0	R/W	I2C SS and FS Spike Suppression Limit Register	0×0000_0005
IC_HS_SPKLEN	I2C_BA+0×A4	R/W	I2C HS Spike Suppression Limit Register	0×0000_0001
IC_CLR_RESTART_DET	I2C_BA+0×A8	R	Clear RESTART_DET Interrupt Register	0×0000_0000
IC_SCL_STUCK_AT_LOW_TIMEOUT	I2C_BA+0×AC	R/W	I2C SCL Stuck at Low Timeout	0×FFFF_FFFF
IC_CLR_SCL_STUCK_DET	I2C_BA+0×B4	R	Clear SCL Stuck at Low Detect Interrupt Register	0×0000_0000

3.14.6 Operation of Interrupt Registers

Table 3-21 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 3-21 Operation of the Interrupt Register

Interrupt Bit Fields	Set by Hardware/ Cleared by Software	Set and Cleared by Hardware
MST_ON_HOLD	×	√
RESTART_DET	√	×
GEN_CALL	√	×
START_DET	√	×
STOP_DET	√	×
ACTIVITY	√	×
RX_DONE	√	×
TX_ABRT	√	×
RD_REQ	√	×
TX_EMPTY	×	√
TX_OVER	√	×
RX_FULL	×	√
RX_OVER	√	×
RX_UNDER	√	×

3.14.7 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 ic_clk domain. Therefore, there is delay when the HCLK register reflects the activity that occurred on the ic_clk side.

Some registers may be written only when the I2C is disabled, programmed by the IC_ENABLE 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.

3.14.7.1 IC_CON

This register can be written only when the I2C is disabled, which corresponds to IC_ENABLE[0] being set to 0. Writes at other times have no effect.

Register	Offset	R/W	Description	Reset Value
IC_CON	I2C_BA+0×00	R/W	I2C Control Register	0×0000_007F

Bits	Description	
[31:8]	Reserved	Reserved.
[7]	STOP_DET_IFADDRESSED	In slave mode: 1'b1 – issues the STOP_DET interrupt only when it is addressed. 1'b0 – issues the STOP_DET irrespective of whether it's addressed or not. Dependencies: This register bit value is applicable in the slave mode only (MASTER_MODE = 1'b0) Note: During a general call address, this slave does not issue the STOP_DET interrupt if STOP_DET_IF_ADDRESSED = 1'b1, even if the slaveresponds to the general call address by generating ACK. The STOP_DET interrupt is generated only when thetransmitted address matches the slave address (SAR).
[6]	IC_SLAVE_DISABLE	This bit controls whether I2C has its slave disabled, which means once the presetn signal is applied, then this bit takes on the value of the configuration parameter IC_SLAVE_DISABLE. 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, theslave 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 aslave. 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]	IC_RESTART_EN	Determines whether RESTART conditions may be sent whenacting as a master. Some older slaves do not supporthandling 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 asubsequent START condition, split operations are brokendown into multiple I2C transfers. If the aboveoperations are performed, it will result in setting bit 6 (TX_ABRT) of the IC_RAW_INTR_STAT register.
[4]	IC_10BITADDR_MASTER_rd_only	The function of this bit is handled by bit 12 of IC_TAR register, and becomes a read-only copy called IC_10BITADDR_MASTER_rd_only. 0 = 7-bit addressing 1 = 10-bit addressing
[3]	IC_10BITADDR_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 transactionsthat involve 10-bit addressing; for 7-bit addressing, only thelower 7 bits of the IC_SAR register are compared. 1 = 10-bit addressing. The I2C responds to only10-bit addressing transfers that match the full 10 bits of theIC_SAR register.
[2:1]	SPEED	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 programmedby 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_MODE	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'.

3.14.7.2 IC_TAR

Register	Offset	R/W	Description	Reset Value
IC_TAR	I2C_BA+0×04	R/W	I2C Target Address Register	0×0000_1055

Bits	Description	
[31:13]	Reserved	Reserved.
[12]	IC_10BITADDR_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]	SPECIAL	This bit indicates whether software performs a Device-ID, General Call or START BYTE command. 0 = ignore bit 10 GC_OR_START and use IC_TAR normally 1 = perform special I2C command as specified in GC_OR_START bit
[10]	GC_OR_START	If bit 11 (SPECIAL) 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 IC_RAW_INTR_STAT register. The I2C remains in General Call mode until the SPECIAL bit value (bit 11) is cleared. 1 = START BYTE
[9:0]	IC_TAR	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 IC_TAR and IC_SAR are the same, loopback exists but the FIFOs are shared between master and slave, so full loopback is not feasible. Only one direction loopback mode is supported (simplex), not duplex. A master cannot transmit to itself; it can transmit to only slave.

3.14.7.3 IC_SAR

Register	Offset	R/W	Description	Reset Value
IC_SAR	I2C_BA+0×08	R/W	I2C Slave Address Register	0×0000_0055

Bits	Descriptions	
[31:10]	Reserved	Reserved.
[9:0]	IC_SAR	The IC_SAR holds the slave address when the I2C is operating as a slave. For 7-bit addressing, only IC_SAR [6:0] is used. This register can be written only when the I2C interface is disabled, which corresponds to IC_ENABLE [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 IC_SAR or IC_TAR to a reserved value.

3.14.7.4 IC_HS_MADDR

Register	Offset	R/W	Description	Reset Value
IC_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]	IC_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 IC_ENABLE[0] being set to 0. Writes at other times have no effect.

3.14.7.5 IC_DATA_CMD

Register	Offset	R/W	Description	Reset Value
IC_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 IC_RESTART_EN is 1, a RESTART is issued before the data issent/received (according to the value of CMD), regardless of whether ornot the transfer direction is changing from the previous command. 0 = If IC_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 immediatelytries to start a new transfer by issuing a START and arbitrating for thebus. 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 thecurrent transfer by sending/receiving data bytes according to the value ofthe CMD bit. If the Tx FIFO is empty, the master holds the SCL line lowand stalls the bus until a new command is available in the Tx FIFO.
[8]	CMD	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 IC_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 IC_RAW_INTR_STAT register), unless bit 11 (SPECIAL) in the IC_TAR 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]	DAT	This register contains the data to be transmitted or received on the I2C bus. If you are writing to this register and want to perform a read, bits 7:0 (DAT) are ignored by the I2C. However, when you read this register, these bits return the value of data received on the I2C interface.

3.14.7.6 IC_SS_SCL_HCNT

Register	Offset	R/W	Description	Reset Value
IC_SS_SCL_HCNT	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]	IC_SS_SCL_HCNT	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 IC_ENABLE[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 conditionwhen this counter reaches a value of IC_SS_SCL_HCNT + 10.

3.14.7.7 IC_SS_SCL_LCNT

Register	Offset	R/W	Description	Reset Value
IC_SS_SCL_LCNT	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]	IC_SS_SCL_LCNT	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 IC_ENABLE[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. .

3.14.7.8 IC_FS_SCL_HCNT

Register	Offset	R/W	Description	Reset Value
IC_FS_SCL_HCNT	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]	IC_FS_SCL_HCNT	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 IC_ENABLE[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.

3.14.7.9 IC_FS_SCL_LCNT

Register	Offset	R/W	Description	Reset Value
IC_FS_SCL_LCNT	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]	IC_FS_SCL_LCNT	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 IC_ENABLE[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.

3.14.7.10 IC_HS_SCL_HCNT

Register	Offset	R/W	Description	Reset Value
IC_HS_SCL_HCNT	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]	IC_HS_SCL_HCNT	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 IC_ENABLE[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.

3.14.7.11 IC_HS_SCL_LCNT

Register	Offset	R/W	Description	Reset Value
IC_HS_SCL_LCNT	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]	IC_HS_SCL_LCNT	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, theSCL 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 IC_ENABLE[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.

3.14.7.12 IC_INTR_STAT

Register	Offset	R/W	Description	Reset Value
IC_INTR_STAT	I2C_BA+0×2C	R	I2C Interrupt Status Register	0×0000_0000

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[13]	R_MST_ON_HOLD	See *IC_RAW_INTR_STAT* for a detailed description of this bit.
[12]	Reserved	Reserved
[11]	R_GEN_CALL	See *IC_RAW_INTR_STAT* for a detailed description of this bit.
[10]	R_START_DET	
[9]	R_STOP_DET	
[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	

3.14.7.13 IC_INTR_MASK

Register	Offset	R/W	Description	Reset Value
IC_INTR_MASK	I2C_BA+0×30	R/W	I2C Interrupt Mask Register	0×0000_08FF

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[13]	M_MST_ON_HOLD	This bit masks the R_MST_ON_HOLD interrupt bit in the *IC_INTR_STAT* register.(Read-Only)
[12]	Reserved	Reserved
[11]	M_GEN_CALL	These bits mask their corresponding interrupt status bits in the *IC_INTR_STAT* register.
[10]	M_START_DET	
[9]	M_STOP_DET	
[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	

3.14.7.14 IC_RAW_INTR_STAT

Register	Offset	R/W	Description	Reset Value
IC_RAW_INTR_STAT	I2C_BA+0×34	R	I2C raw Interrupt Status Register	0×0000_0000

Bits	Descriptions	
[31:14]	Reserved	Reserved.
[13]	MST_ON_HOLD	Indicates whether a master is holding the bus and the Tx FIFO is empty.
[12]	Reserved	Reserved
[11]	GEN_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 *IC_CLR_GEN_CALL* register. I2C stores the received data in the Rx buffer.
[10]	START_DET	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_DET	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 IC_CON[7]=1'b1 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is generated only if the slave is addressed. **Note:** During a general call address, this slave does not issue a STOP_DET interrupt if STOP_DET_IF_ADDRESSED=1'b1, even if the slave responds to the general call address by generating ACK. The STOP_DET interrupt is generated only when the transmitted address matches the slave address (SAR). If IC_CON[7]=1'b0 (STOP_DET_IFADDRESSED), the STOP_DET interrupt is issued irrespective of whether it is being addressed. In Master Mode: The STOP_DET 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 *IC_CLR_ACTIVITY* register Reading the *IC_CLR_INTR* 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 IC_TX_ABRT_SOURCE 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 IC_TX_ABRT_SOURCE register. The Tx FIFO remains in this flushed state until the register

		IC_CLR_TX_ABRT is read. Once this read is performed, theTx 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 IC_DATA_CMD register. This bit is set to 0 just after the processor reads the IC_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 IC_CON 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 IC_TX_TL 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 IC_TX_TL 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 IC_ENABLE[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 ic_en=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 IC_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 ic_en goes to 0, this interrupt is cleared.
[2]	RX_FULL	Set when the receive buffer reaches or goes above the RX_TL threshold in the IC_RX_TL register. It is automatically cleared by hardware when buffer level goes below the threshold. If the module is disabled (IC_ENABLE[0]=0), the RX FIFO is flushed and held in reset; therefore the RX FIFO is not full. So this bit is cleared once IC_ENABLE[0] is set to 0, regardless of the activity that continues.
[1]	RX_OVER	Set if the receive buffer is completely filled to IC_RX_BUFFER_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 (IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en 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 IC_DATA_CMD register. If the module is disabled(IC_ENABLE[0]=0), this bit keeps its level until the master or slave state machines go into idle, and when ic_en goes to 0, this interrupt is cleared.

3.14.7.15 IC_RX_TL

Register	Offset	R/W	Description	Reset Value
IC_RX_TL	I2C_BA+0×38	R/W	I2C Receive FIFO Threshold Register	0×0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	RX_TL	Receive FIFO Threshold Level Controls the level of entries (or above) that triggers the RX_FULL interrupt (bit 2 in *IC_RAW_INTR_STAT* 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.

3.14.7.16 IC_TX_TL

Register	Offset	R/W	Description	Reset Value
IC_TX_TL	I2C_BA+0×3C	R/W	I2C Transmit FIFO Threshold Register	0×0000_0000

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	TX_TL	Transmit FIFO Threshold Level Controls the level of entries (or below) that trigger the TX_EMPTY interrupt (bit 4 in *IC_RAW_INTR_STAT* 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.

3.14.7.17 IC_CLR_INTR

Register	Offset	R/W	Description	Reset Value
IC_CLR_INTR	I2C_BA+0×40	R	Clear Combined and Individual Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	DAT	Read this register to clear the combined interrupt, all individual interrupts, and the *IC_TX_ABRT_SOURCE* register. This bit does not clear hardware clearable interrupts but software clearable interrupts. Refer to Bit 9 of the *IC_TX_ABRT_SOURCE* register for an exception to clearing IC_TX_ABRT_SOURCE.

3.14.7.18 IC_CLR_RX_UNDER

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register.

3.14.7.19 IC_CLR_RX_OVER

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register.

3.14.7.20 IC_CLR_TX_OVER

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register.

3.14.7.21 IC_CLR_RD_REQ

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register.

3.14.7.22 IC_CLR_TX_ABRT

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register, and the *IC_TX_ABRT_SOURCE* 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 *IC_TX_ABRT_SOURCE* register for an exception to clearing IC_TX_ABRT_SOURCE.

3.14.7.23 IC_CLR_RX_DONE

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register.

3.14.7.24 IC_CLR_ACTIVITY

Register	Offset	R/W	Description	Reset Value
IC_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 *IC_RAW_INTR_STAT* register.

3.14.7.25 IC_CLR_STOP_DET

Register	Offset	R/W	Description	Reset Value
IC_CLR_STOP_DET	I2C_BA+0×60	R	Clear STOP_DET Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_STOP_DET	Read this register to clear the STOP_DET interrupt (bit 9) of the *IC_RAW_INTR_STAT* register.

3.14.7.26 IC_CLR_START_DET

Register	Offset	R/W	Description	Reset Value
IC_CLR_START_DET	I2C_BA+0×64	R	Clear START_DET Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_START_DET	Read this register to clear the STOP_DET interrupt (bit 9) of the *IC_RAW_INTR_STAT* register.

3.14.7.27 IC_CLR_GEN_CALL

Register	Offset	R/W	Description	Reset Value
IC_CLR_GEN_CALL	I2C_BA+0×68	R	Clear GEN_CALL Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	CLR_GEN_CALL	Read this register to clear the GEN_CALL interrupt (bit 11) of *IC_RAW_INTR_STAT* register.

3.14.7.28 IC_ENABLE

Register	Offset	R/W	Description	Reset Value
IC_ENABLE	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 ENABLE 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]	ENABLE	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 IC_INTR_STAT 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.

3.14.7.29 IC_STATUS

Register	Offset	R/W	Description	Reset Value
IC_STATUS	I2C_BA+0×70	R	I2C Status Register	0×0000_0006

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[6]	SLV_ACTIVITY	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]	MST_ACTIVITY	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: IC_STATUS[0]—that is, ACTIVITY bit—is the OR of SLV_ACTIVITY and MST_ACTIVITY bits.
[4]	RFF	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]	RFNE	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]	TFE	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]	TFNF	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]	ACTIVITY	I2C Activity Status.

3.14.7.30 IC_TXFLR

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 IC_RAW_INTR_STAT 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
IC_TXFLR	I2C_BA+0×74	R	I2C Transmit FIFO Level Register	0×0000_0000

Bits	Descriptions	
[31:4]	Reserved	Reserved.
[3:0]	TXFLR	Transmit FIFO Level. Contains the number of valid data entries in the transmit FIFO.

3.14.7.31 IC_RXFLR

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 IC_TX_ABRT_SOURCE

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
IC_RXFLR	I2C_BA+0×78	R	I2C Receive FIFO Level Register	0×0000_0000

Bits	Descriptions	
[31:4]	Reserved	Reserved.
[3:0]	RXFLR	Receive FIFO Level. Contains the number of valid data entries in thereceive FIFO.

3.14.7.32 IC_SDA_HOLD

Writes to this register succeed only when IC_ENABLE[0]=0.

The values in this register are in units of ic_clk period. The value programmed in IC_SDA_TX_HOLD 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 (IC_SDA_TX_HOLD) 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 ic_clk cycles.

Register	Offset	R/W	Description	Reset Value
IC_SDA_HOLD	I2C_BA+0×7C	R/W	I2C SDA Hold Time Length Register	0×0001_0001

Bits	Descriptions	
[31:24]	Reserved	Reserved.
[23:16]	IC_SDA_RX_HOLD	Sets the required SDA hold time in units of ic_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]	IC_SDA_TX_HOLD	Sets the required SDA hold time in units of ic_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).

3.14.7.33 IC_TX_ABRT_SOURCE

This register has 32 bits that indicate the source of the TX_ABRT bit. Except for Bit 9, this register is cleared whenever the IC_CLR_TX_ABRT register or the IC_CLR_INTR register is read. To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; RESTART must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (IC_TAR[11]), or the GC_OR_START bit must be cleared (IC_TAR[10]).

Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT 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
IC_TX_ABRT_SOURCE	I2C_BA+0×80	R	I2C Transmit Abort Source Register	0×0000_0000

Bits	Descriptions	
[31]	TX_FLUSH_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]	ABRT_USER_ABRT	This is a master-mode-only bit. Master has detected the transfer abort (*IC_ENABLE* [1]). Role of I2C: Master-Transmitter
[15]	ABRT_SLVRD_INTX	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 CMD (bit 8) of *IC_DATA_CMD* register. Role of I2C:Slave-Transmitter
[14]	ABRT_SLV_ARBLOST	1 = Slave lost the bus while transmitting datato a remote master. *IC_TX_ABRT_SOURCE* [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]	ABRT_SLVFLUSH_TXFIFO	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]	ARB_LOST	1 = Master has lost arbitration, or if *IC_TX_ABRT_SOURCE* [14] is also set, then the slave transmitter has lost arbitration. Role of I2C:Slave-Transmitter Master-Transmitter
[11]	ABRT_MASTER_DIS	1 = User tries to initiate a Master operation with the Master mode disabled. Role of I2C:Master-Receiver

		Master-Transmitter
[10]	ABRT_10B_RD_NORSTRT	1 = The restart is disabled (IC_RESTART_EN bit (*IC_CON* [5]) = 0) and the master sends a read command in10-bit addressing mode. Role of I2C:Master-Receiver
[9]	ABRT_SBYTE_NORSTRT	To clear Bit 9, the source of the ABRT_SBYTE_NORSTRT must be fixed first; restart must be enabled (IC_CON[5]=1), the SPECIAL bit must be cleared (*IC_TAR* [11]), or the GC_OR_START bit must be cleared(IC_TAR[10]). Once the source of the ABRT_SBYTE_NORSTRT is fixed, then this bit can be cleared in the same manner as other bits in this register. If the source of the ABRT_SBYTE_NORSTRT 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 (IC_RESTART_EN bit (IC_CON[5]) = 0) and the user is trying to send a START Byte. Role of I2C: Master
[8]	ABRT_HS_NORSTRT	1 = The restart is disabled (IC_RESTART_EN bit (IC_CON[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]	ABRT_SBYTE_ACKDET	1 = Master has sent a START Byte and the START Byte was acknowledged (wrong behavior). Role of I2C:Master
[6]	ABRT_HS_ACKDET	1 = Master is in High Speed mode and the High Speed Master code was acknowledged (wrong behavior). Role of I2C: Master
[5]	ABRT_GCALL_READ	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 (IC_DATA_CMD[9] is set to 1). Role of I2C: Master-Transmitter
[4]	ABRT_GCALL_NOACK	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]	ABRT_TXDATA_NOACK	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]	ABRT_10ADDR2_NOACK	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 Master-Receiver
[1]	ABRT_10ADDR1_NOACK	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 Master-Receiver
[0]	ABRT_7B_ADDR_NOACK	1 = Master is in 7-bit addressing mode and the address sent was not acknowledged by any slave. Role of I2C: Master-Transmitter Master-Receiver

3.14.7.34 IC_SLV_DATA_NACK_ONLY

Register	Offset	R/W	Description	Reset Value
IC_SLV_DATA_NACK_ONLY	I2C_BA+0×84	R/W	Generate Slave Data NACK Register	0×0000_0000

Bits	Description		
[31:1]	Reserved	N/A	Reserved.
[0]	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

3.14.7.35 IC_DMA_CR

Register	Offset	R/W	Description	Reset Value
IC_DMA_CR	I2C_BA+0×88	R/W	DMA Control Register	0×0000_0000

Bits	Descriptions	
[31:2]	Reserved	Reserved.
[1]	TDMAE	Transmit DMA Enable. This bit enables/disables the transmit FIFO DMA channel. 0 = Transmit DMA disabled 1 = Transmit DMA enabled
[0]	RDMAE	Receive DMA Enable. This bit enables/disables the receive FIFO DMA channel. 0 = Receive DMA disabled 1 = Receive DMA enabled

3.14.7.36 IC_DMA_TDLR

Register	Offset	R/W	Description	Reset Value
IC_DMA_TDLR	I2C_BA+0×8C	R/W	DMA Transmit Data Level Register	0×0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2:0]	DMATDL	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.

3.14.7.37 IC_DMA_RDLR

Register	Offset	R/W	Description	Reset Value
IC_DMA_RDLR	I2C_BA+0×90	R/W	DMA Receive Data Level Register	0×0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2:0]	DMARDL	Receive Data Level. This bit field controls the level at which a DMA request is made by the receive logic. The watermark level = DMARDL+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 RDMAE = 1. For instance, when DMARDL is 0, then dma_rx_req is asserted when 1 or more data entries are present in the receive FIFO.

3.14.7.38 IC_SDA_SETUP

Register	Offset	R/W	Description	Reset Value
IC_SDA_SETUP	I2C_BA+0×94	R/W	I2C SDA Setup Register	0×0000_0064

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	SDA_SETUP	SDA Setup. It is recommended that if the required delay is 1000ns, then for an ic_clk frequency of 10 MHz, IC_SDA_SETUP should be programmed to a value of 11. Note: Writes to this register succeed only when IC_ENABLE [0] = 0.

3.14.7.39 IC_ACK_GENERAL_CALL

Register	Offset	R/W	Description	Reset Value
IC_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_GEN_CALL	ACK General Call. When set to 1, I2C responds with a ACK (by asserting ic_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.

3.14.7.40 IC_ENABLE_STATUS

The register is used to report the I2C hardware status when IC_ENABLE[0] is set from 1 to 0; that is, when I2C is disabled.

- If IC_ENABLE[0] has been set to 1, bits 2:1 are forced to 0, and bit 0 is forced to 1.
- If IC_ENABLE[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
IC_ENABLE_STATUS	I2C_BA+0×9C	R	I2C Enable Status Register	0×0000_0000

Bits	Descriptions	
[31:3]	Reserved	Reserved.
[2]	SLV_RX_DATA_LOST	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 IC_ENABLE[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 IC_ENABLE[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 IC_EN (bit 0) is read as 0.
[1]	SLV_DISABLED_WHILE_BUSY	Slave Disabled While Busy (Transmit, Receive). This bit indicates if a potential or active Slave operation has been aborted due tosetting bit 0 of the IC_ENABLE register from 1 to 0. This bit is setwhen the CPU writes a 0 to bit 0 of IC_ENABLE while: ■ I2C is receiving the address byte of the Slave-Transmitteroperation from a remote master; OR, ■ Address and data bytes ofthe Slave-Receiver operation from a remote master. When read as 1, I2C is deemed to have forced a NACKduring any part of an I2C transfer, irrespective of whether the I2Caddress matches the slave address set in I2C (IC_SARregister) OR if the transfer is completed before bit 0 of IC_ENABLEis 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 IC_ENABLE 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 *IC_EN* (bit 0) is read as 0.
[0]	IC_EN	ic_en Status.

		This bit always reflects the value driven on the output port ic_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 SLV_RX_DATA_LOST (bit 2) and SLV_DISABLED_WHILE_BUSY (bit 1).

3.14.7.41 IC_FS_SPKLEN

Register	Offset	R/W	Description	Reset Value
IC_FS_SPKLEN	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]	IC_FS_SPKLEN	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_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 IC_ENABLE[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.

3.14.7.42 IC_HS_SPKLEN

Register	Offset	R/W	Description	Reset Value
IC_HS_SPKLEN	I2C_BA+0×A4	R/W	I2C HS Spike Suppression Limit Register	0×0000_0001

Bits	Descriptions	
[31:8]	Reserved	Reserved.
[7:0]	IC_HS_SPKLEN	This register must be set before any I2C bus transaction can take place to ensure stable operation. This register sets the duration, measured in ic_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 IC_ENABLE[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.

3.14.7.43 IC_CLR_RESTART_DET

Register	Offset	R/W	Description	Reset Value
IC_CLR_RESTART_DET	I2C_BA+0×A8	R	Clear RESTART_DET Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved.
[0]	RESTART	Read this register to clear the RESTART_DET interrupt (bit 12) of the *IC_RAW_INTR_STAT* register.

3.14.7.44 IC_SCL_STUCK_AT_LOW_TIMEOUT

Register	Offset	R/W	Description	Reset Value
IC_SCL_STUCK_AT_LOW_TIMEOUT	I2C_BA+0×AC	R/W	I2C SCL Stuck at Low Timeout	0×FFFF_FFFF

Bits	Descriptions	
[31:0]	IC_SCL_STUCK_LOW_TIMEOUT	I2C generates the interrupt to indicate SCL stuck at low if it detects the SCL stuck at low for the IC_SCL_STUCK_LOW_TIMEOUT in units of ic_clk period.

3.14.7.45 IC_CLR_SCL_STUCK_DET

Register	Offset	R/W	Description	Reset Value
IC_CLR_SCL_STUCK_DET	I2C_BA+0×B4	R	Clear SCL Stuck at Low Detect Interrupt Register	0×0000_0000

Bits	Descriptions	
[31:1]	Reserved	Reserved
[0]	IC_SDA_STUCK_AT_LOW_TIMEOUT	Read this register to clear the SCL_STUCK_DET interrupt (bit 14) of the *IC_RAW_INTR_STAT* register.

3.15 Inter-IC Sound (I2S)

3.15.1 Overview

The I2S bus can only handle audio data transmissions. The I2S protocol requires a minimum of three wires—data (sd), word select (ws), and serial clock (sclk)—keeping the design simple and the pin count minimal. I2S supports the standard I2S frame format for transmitting and receiving data —the MSB of a word is sent one sclk cycle after a word select change.

3.15.2 Features

- APB data bus widths of 32 bits
- Full duplex communication due to the independence of transmitter and receiver
- Master or slave mode of operation
- FIFO depth of 8 words
- Audio data resolutions of 12, 16, 20, 24, and 32 bits
- Support single stereo channel to transmit or receive
- Support DMA

3.15.3 Block Diagram

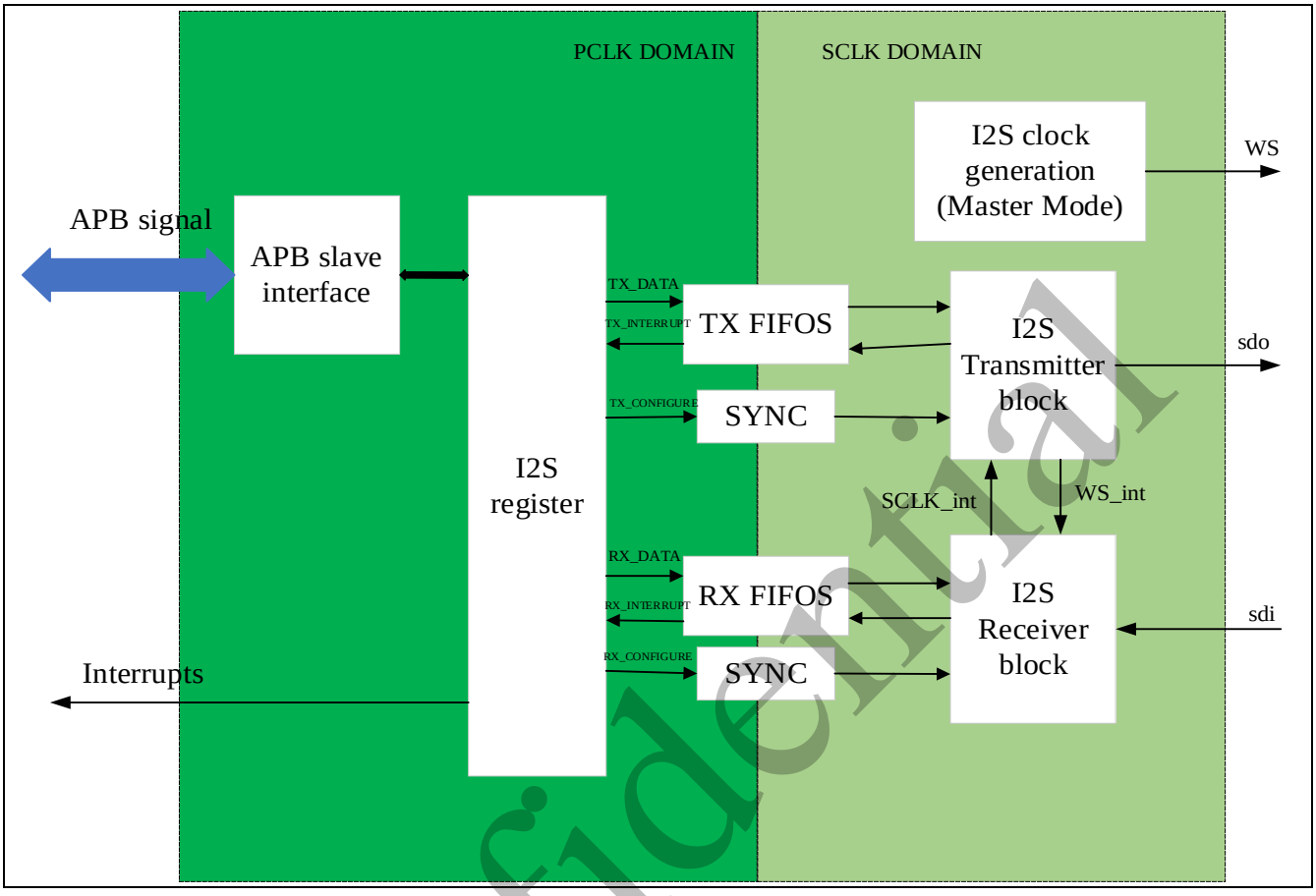


Figure 3-87 I2S Block Diagram

3.15.4 Functional Description

3.15.4.1 I2S Enable

You must enable the I2S component before any data can be received or transmitted into the FIFO. To enable the component, set the I2S Enable (IEN) bit of the I2S Enable Register (IER) to 1. When you disable the device, it acts as a global disable. To disable I2S, set IER[0] to 0. After disable, the following events occur:

- TX and RX FIFO are cleared, and read/write pointers are reset
- Any data in the process of being transmitted or received is lost
- All other programmable enables (such as transmitter/receiver block enables and individual TX/RX channel enables) in the component are overridden
- Generation of master mode clock signals ws is disabled (for instance, they are held low)

When I2S master is enabled, the device always starts in the left stereo data cycle (ws = 0), and one sclk cycle later transitions to the right stereo data cycle (ws = 1). This allows for half a frame of sclks to write data to the TX FIFO and to ensure that any connected slave receivers do not miss the start of the data frame (for instance, the ws 1-to-0 transition) once the sclk restarts. (ws of I2S slave is externally supplied.)

On reset, the IER[0] is set to 0 (disable).

3.15.4.2 I2S as Transmitter

The I2S component supports only one stereo I2S transmit (TX) channel. This channel operates in master mode or slave mode. Stereo data pairs (such as, left and right audio data) written to a TX channel via the APB bus are shifted out serially on the appropriate serial data out line (sdo). The shifting is timed with respect to the serial clock (sclk) and the word select line (ws). The basic usage flow (I2S as Transmitter) is shown in Figure 3-88.

Transmitter Block Enable

The Transmitter Block Enable (TXEN) bit of the I2S Transmitter Enable Register (ITER) globally turns on and off all of the configured TX channel. To enable the transmitter block, set ITER[0] to 1. To disable the block, set ITER[0] to 0. When the transmitter block is disabled, the following events occur:

- Outgoing data is lost and the channel outputs are held low;
- Data in the TX FIFO are preserved and the FIFO can be written to; and
- Any previous programming (like changes in word size, threshold levels, and so on) of the

TX channels is preserved;

- Any individual TX channel enables are overridden.

When the transmitter block is enabled, if there is data in the TX FIFO, the channel resumes transmission on the next left stereo data cycle (such as when the ws line goes low).

When the block is disabled, you can perform any of the following procedures:

- Program (or further program) TX channel registers
- Flush the TX FIFO by programming the Transmitter FIFO Reset bit of the Transmitter FIFO Flush Register (TXFFR[0] = 1)
- Flush an individual channel's TX FIFO by programming the Transmit Channel FIFO Reset (TXCHFR) bit of the Transmit FIFO Flush Register (TFF[0] = 1)

On reset, the ITER[0] is set to 0 (disable)

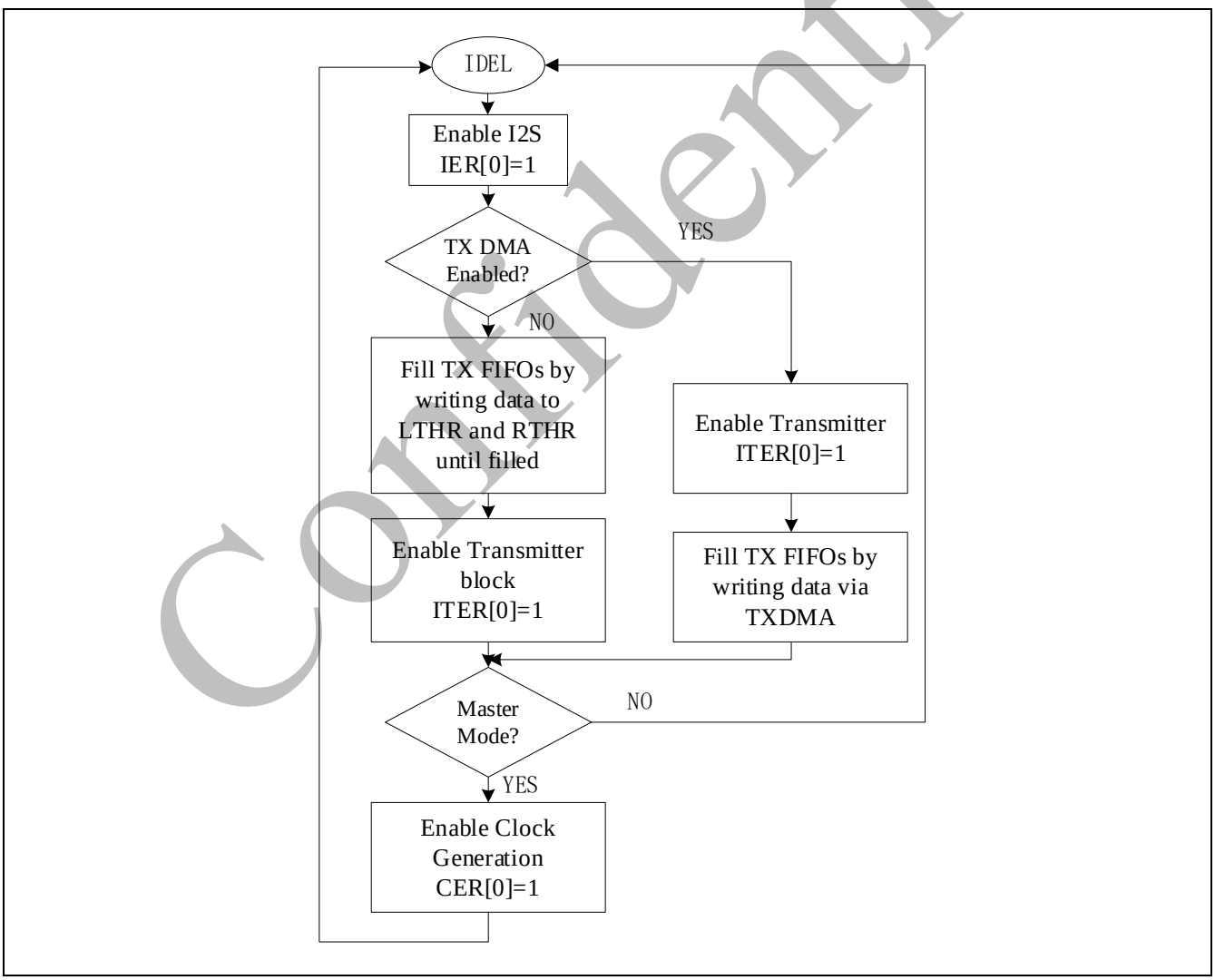


Figure 3-88 Transmitter Block Flow Chart

Transmitter Channel Enable

Each transmit channel has its own enable/disable that can be set independently of the other channels to allow the reprogramming of a channel and to flush the channel's TX FIFO while other TX channels are transmitting. This enable/disable is controlled by bit 0 of the Transmitter Enable Register (TER). When a TX channel is disabled, the following occurs:

- Outgoing stereo data is lost;
- Channel output is held low;
- Data in the TX FIFO is preserved, and the FIFO can be written to; and
- Any previous programming of the TX channel's registers is preserved, and the registers can be further reprogrammed.

When a TX channel is disabled, you can flush the channel's TX FIFO by programming the Transmit Channel FIFO Reset (TXCHFR) bit of the Transmit FIFO Flush (TFF[0] = 1). When the TX channel is enabled, if there is data in the TX FIFO, the channel resumes transmission on the next left stereo data cycle (such as, when the ws line goes low).

On reset, the TFF[0] is set to 1 (enable).

Transmit Channel Audio Data Resolution

TX Channel is initially configured with a 32-bit audio resolution, it can be programmed to support a resolution of 12, 16, 20, 24, or 32 bits.

Reprogramming of the audio resolution ensures that the MSB of the data is still transmitted first if the resolution of the data to be sent is reduced. Changes to the resolution are programmed via the Word Length (WLEN) bits of the Transmitter Configuration Registers (TCR[2:0]). The channel must be disabled prior to any resolution changes. On reset or if an invalid resolution is selected, the TX channel's audio data resolution defaults back to 32-bit audio resolution.

Transmit Channel FIFO

Each Transmit Channel has two FIFO banks for left and right stereo data. The FIFO width is 32. There are several ways to clear the TX FIFO and reset the read/write pointers as described as follows:

- on reset
- by disabling I2S (IER[0] = 0)
- by flushing the transmitter block (TXFFR[0] = 1)
- by flushing an individual TX channel (TFF[0] = 1)

You must disable the transmitter block/channel before the transmitter block and individual channel FIFO can be flushed.

The TX FIFO Empty Threshold Trigger Level parameter sets the default trigger threshold level for the TX FIFO. The trigger level can be set to any value in the range of 0 to FIFO_DEPTH – 1. When this level is reached, a transmit channel empty interrupt is generated. This level can be reprogrammed during operation by writing to the Transmit Channel Empty Trigger (TXCHET) bits of the Transmit FIFO Configuration Register (TFCR[3:0]).

You must disable the TX channel prior to changing the trigger level.

Transmit Channel Interrupts

TX channel generates two interrupts: TX FIFO Empty and Data Overrun.

- TX FIFO Empty interrupt – This interrupt is asserted when the empty trigger threshold level for the TX FIFO is reached. A TX FIFO Empty interrupt is cleared by writing data to the TX FIFO to bring its level above the empty trigger threshold level for the channel.
- Data Overrun interrupt –This interrupt is asserted when an attempt is made to write to a full TX FIFO (any data being written is lost while data in the FIFO is preserved). A Data Overrun interrupt is cleared by reading the Transmit Channel Overrun (TXCHO) bit [0] of the Transmit Overrun Register (TOR).

Writing to a Transmit Channel

The stereo data pairs to be transmitted by a TX channel are written to the TX FIFO via the Left Transmit Holding Register (LTHR) and the Right Transmit Holding Register (RTHR). All stereo data pairs must be written using the following two stage process:

1. Write left stereo data to LTHR
2. Write right stereo data to RTHR

When TX DMA is enabled, data to be transmitted by TX channels are written to the TX FIFO via the TXDMA register rather than through LTHR and RTHR.

When I2S is enabled, if the TX FIFO is empty and data is not written to the FIFO before the next left cycle, the channel outputs zeros for a full frame (left and right cycle). Transmission only commences if there is data in the TX FIFO prior to the transition to the left data cycle. In other words, if the start of the frame is missed, the channel output idles until the next available frame.

3.15.4.3 I2S as Receiver

The I2S component supports only one stereo I2S receive (RX) channel. This channel operates

in master mode or slave mode. Stereo data pairs (such as, left and right audio data) are received serially from a data input line (sdi). These data words are stored in RX FIFO until they are read via the APB bus. The receiving is timed with respect to the serial clock (sclk) and the word select line (ws). The basic usage flow (I2S as Receiver) is shown in Figure 3-89. RX channel generates two interrupts: RX FIFO Data Available and Data Overrun.

- RX FIFO Data Available interrupt – This interrupt is asserted when the trigger level for the RX FIFO is reached. This interrupt is cleared by reading data from the RX FIFO until its level drops below the data available trigger level for the channel.
- Data Overrun interrupt – This interrupt is asserted when an attempt is made to write received data to a full RX FIFO (any data being written is lost while data in the FIFO is preserved). This interrupt is cleared by reading the Receive Channel Overrun (RXCHO) bit [0] of the Receive Overrun Register (ROR).

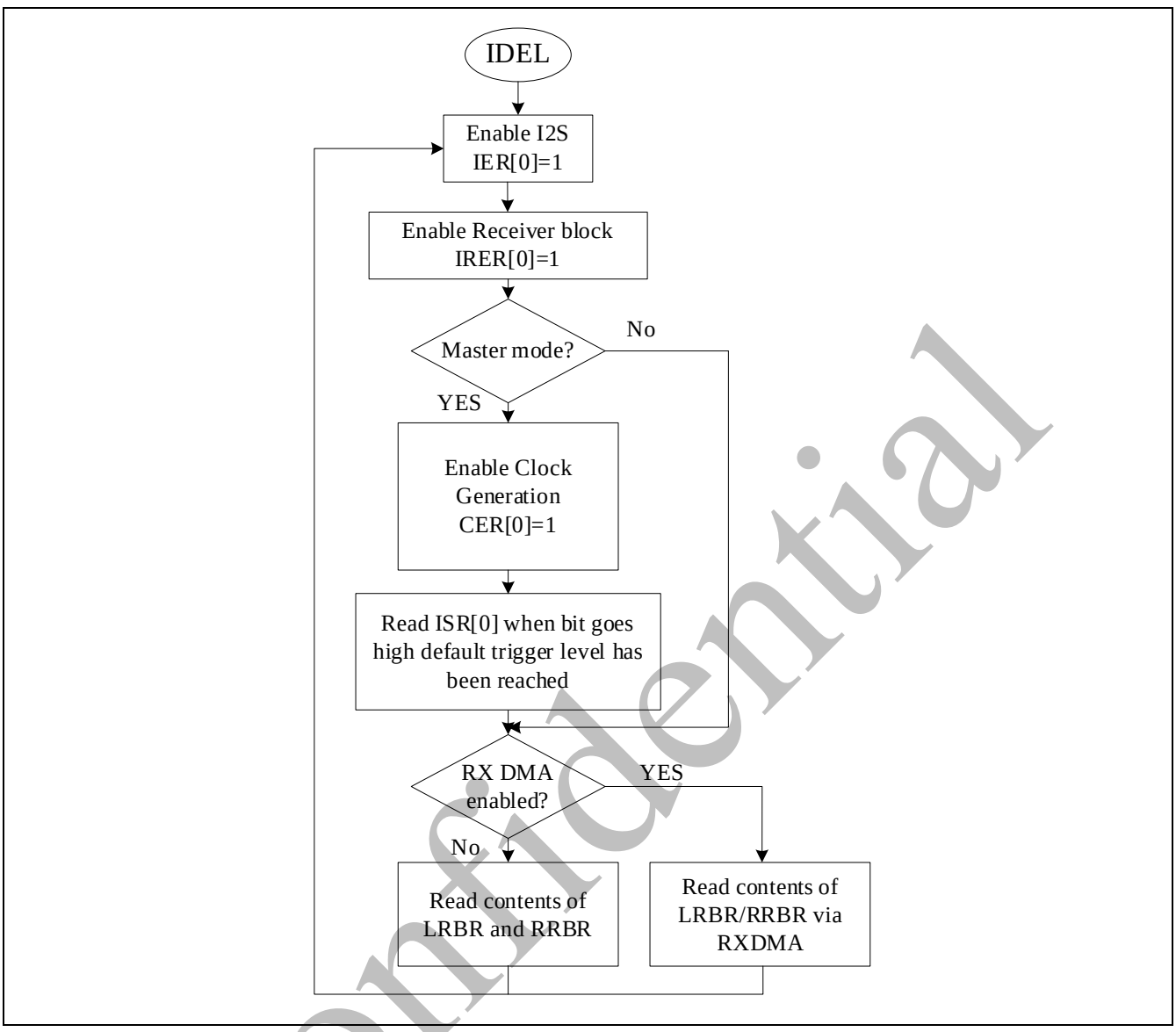


Figure 3-89 Receiver Flow Chart

Receiver Block Enable

The Receiver Block Enable (RXEN) bit of the I2S Receiver Enable Register (IRER) enables/disables all configured RX channels. To enable the receiver block, set IRER[0] to '1'. To disable the block, set this bit to '0'.

When the receiver block is disabled, the following events occur:

- Incoming data is lost;
- Data in the RX FIFO is preserved and the FIFO can be read;
- Any previous programming (such as changes in word size, threshold levels, and so on) of the RX channels is preserved; and
- Any individual RX channel enable is overridden. Enabling the channel resumes receiving on the next left stereo data cycle (for instance, when ws goes low).

When the block is disabled, you can perform any of the following procedures:

- Program (or further program) the RX channel registers;
- Flush the RX FIFO by programming the Receiver FIFOReset (RXFR) bit of the Receiver FIFO Flush Register (RXFFR = 1).
- Flush an individual channel's RX FIFO by programming the Receive Channel FIFO Reset (RXCHFR) bit of the Receive FIFO Flush Register (RFF = 1).

On reset, IRER[0] is set to 0 (disable)

Receive Channel Enable

Each RX channel has its own enable/disable that can be used to allow programming of the channel and to clear the channel's RX FIFO while other RX channels are still receiving data. This enable/disable is controlled by bit 0 of the Receiver Enable Register (RER).

When the RX channel is disabled, the following occurs:

- Incoming data is lost;
- Data in the RX FIFO is preserved;
- FIFO can be read;
- Previous programming of the RX channel is preserved; and
- RX channel can be further programmed.

When the RX channel or block is disabled, you can flush the channel's RX FIFO by writing 1 in bit 0 of the Receive FIFO Flush Register (RFF). When the channel is enabled, it resumes receiving on the next left stereo data cycle (for instance. when ws line goes low).

On reset, the RFF is set to 1 (enable).

Receive Channel Audio Data Resolution

Each RX channel is initially configured with a 32bit audio data resolution . An RX channel can be programmed during operation to any supported audio data resolution that is less than 32. For example, if the RX Channel is initially a 32-bit audio resolution, it can be programmed to support resolutions of 12 16, 20, 24, or 32 bits. However, if RX Channel is initially a 20-bit audio resolution, it can only be programmed to support resolutions of 12, 16, or 20 bits. Any other resolution values are considered invalid.

This programming ensures that the LSB of the received data is placed in the LSB position of the RX FIFO if the resolution of the data being received is reduced. Changes to the resolution are programmed via the Word Length (WLEN) bits of the Receive Configuration registers (RCR). The channel must be disabled prior to any resolution changes.

The RX channel also supports unknown data resolutions. If the received word is greater than the configured channel resolution, the least significant bits are ignored. If the received word is less than the configured/programmed channel resolution, the least significant bits are padded with zeros.

On reset or if an invalid resolution is selected, the RX channel's audio data resolution defaults back to the initial parameter setting of 32.

Receive Channel FIFO

Each Receive Channel has two FIFO banks for left and right stereo data. The FIFO width is 32. The RX FIFO can be cleared and the read/write pointers reset in a number ways, as described as follows:

- On reset
- By disabling i2s (ier = 0)
- By flushing the receiver block (rxffr = 1)
- By flushing an individual rx channel (rff = 1)

Before you flush the receiver block, you must disable the receiver block. The RX FIFO Data Available Level parameter sets the default data available trigger level for the RX FIFO. When this level is reached, a RX channel data available interrupt is generated. The valid values are 0 to FIFO_DEPTH–1, which correspond to trigger levels of 1 to FIFO_DEPTH (for example, Trigger Level = Configured Value + 1). This level can be reprogrammed during operation via the Receive Channel Data Trigger (RXCHDT) bits of the Receive FIFO Configuration Register (RFCR[3:0]). The RX channel needs to be disabled prior to any changes in the trigger level.

Receive Channel Interrupts

RX channel generates two interrupts: RX FIFO Data Available and Data Overrun.

- RX FIFO Data Available interrupt–This interrupt is asserted when the trigger level for the RX FIFO is reached. This interrupt is cleared by reading data from the RX FIFO until its level drops below the data available trigger level for the channel.
- Data Overrun interrupt–This interrupt is asserted when an attempt is made to write received data toa full RX FIFO (any data being written is lost while data in the FIFO is preserved). This interrupt is cleared by reading the Receive Channel Overrun (RXCHO) bit [0] of the Receive Overrun Register (ROR)

The interrupt status of any RX channel can be determined by polling the Interrupt Status

Register (ISR). The RXDA bit [0] indicates the status of the RX FIFO Data Available interrupt; the RXFO bit [1] indicates the status of the RX FIFO Data Overrun interrupt.

Both the Receive Empty Threshold and Data Overrun interrupts can be masked by writing a 1 in the Receive Empty Threshold Mask (RDM) and Receive Overrun Mask (ROM) bits of the Interrupt Mask Register (IMR), respectively. However, the ISR always shows the current status of the interrupts regardless of any masking.

Reading from a Receive Channel

The stereo data pairs received by a RX channel are written to the left and right RX FIFO. These FIFO can be read via the Left Receive Buffer Register (LRBR) and the Right Receive Buffer Register (RRBR). All stereo data pairs must be read using the following two-stage process:

1. Read the left stereo data from LRBR.
2. Read the right stereo data from RRBR.

When RX DMA is enabled, data can be read from RX FIFO via the RXDMA register rather than through LRBR and RRBR.

The RRXDMA register resets the RXDMA read cycle. This register provides the same functionality as the RTXDMA register, but targets RXDMA instead.

3.15.4.4 Clock Generation (Master Mode)

Clock Generation Enable

The Clock Generation Enable (CLKEN) bit of the Clock Enable Register (CER) enables and disables the master mode clock signal: ws. To enable this signal, sets CER[0] to 1; to disable it, sets this bit to 0, in which case ws is held low (ws = 0).

When the CLKEN bit is disabled, any incoming or outgoing data is lost. However, data already in the RX and TX FIFO are preserved. After this bit is enabled, transmission recommences at the start of the next stereo frame.

On enabling CER[0], ws always starts in the left stereo data cycle (ws = 0). One sclk cycle later, it transitions to the right stereo data cycle (ws = 1); hence a 0-to-1 transition. This allows for half a frame of sclk to write data to the TX FIFO and ensures that any connected slave receivers do not miss the start of the data frame (the ws 1-to-0 transition) once the sclk restarts.

To further explain this behavior, the ws transitions—0-to-1 and 1-to-0—are used by the device to clock the start of the right or left stereo data cycle. The transition 1-to-0 indicates the start of a new stereo data pair. Because I2S is simple in terms of control, for every 1-to-0 transition,

the device sends the next entry in its FIFO (similarly, the Receiver assumes new data is being sent and starts receiving). On enabling CER[0], the device starts with ws = 0 for one cycle and then transitions 0-to-1. This allows connected devices to clock off the transition and determine which cycle they are in. However, because it's not a 1-to-0 transition, the devices do not have to start TX or RX with potential garbage (assuming FIFO are empty prior to enable). Additionally, the transmission ensures that after a clk_en, there is ample time to configure TX or RX and to input some data for transmission before the start of the next frame.

Word Select Generation

WS's setting can be reprogrammed during operation of the component by setting the Word Select Size (WSS) bits [4:3] of the Clock Configuration Register (CCR). You must disable the Clock Generation block (CER[0] = 0) before you can change the word select size.

The I2S supports 16, 24, or 32 sclk cycles per left/right ws cycle as illustrated in Figure 3-90 .

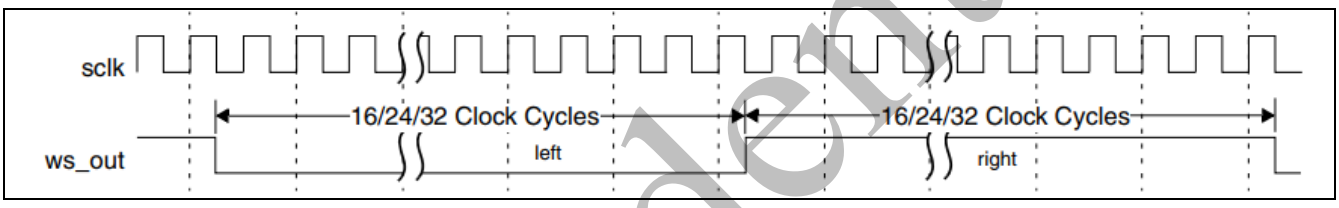


Figure 3-90 I2S Clock Cycles

3.15.4.5 Programming I2S

Slave Mode

- Normal Mode

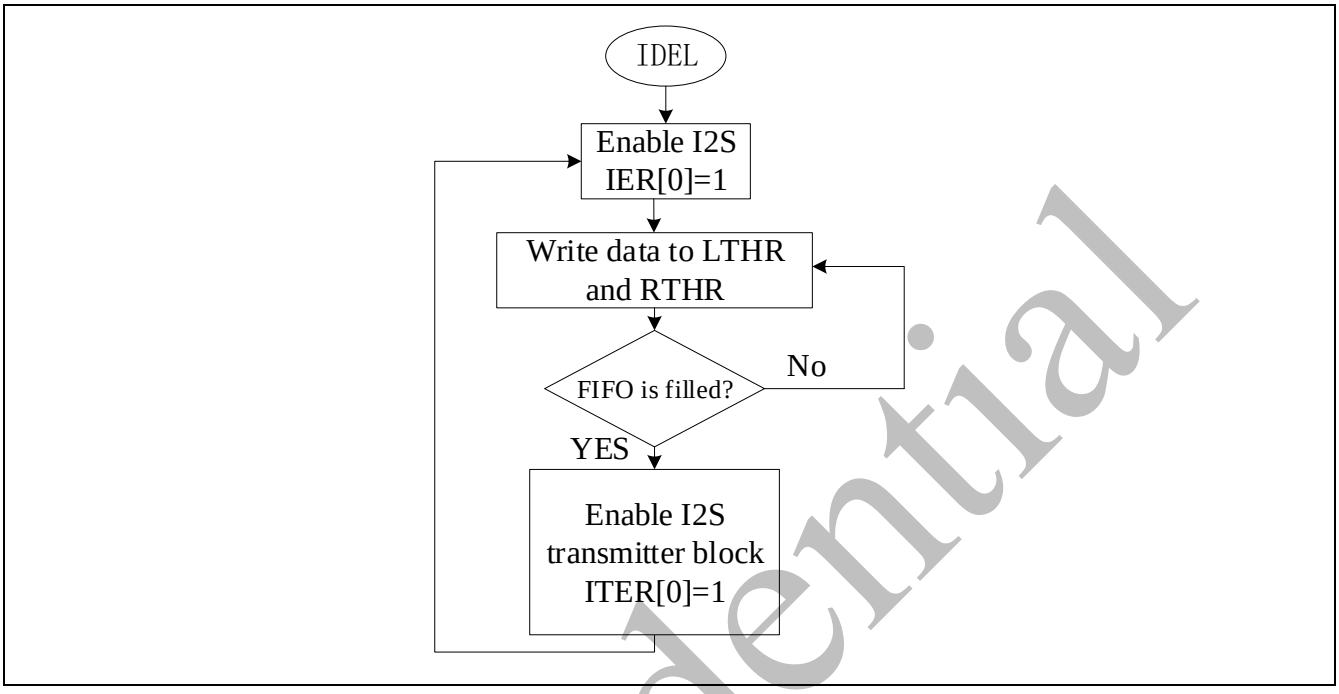


Figure 3-91 Normal Mode Flow Chart

- TX DMA Mode

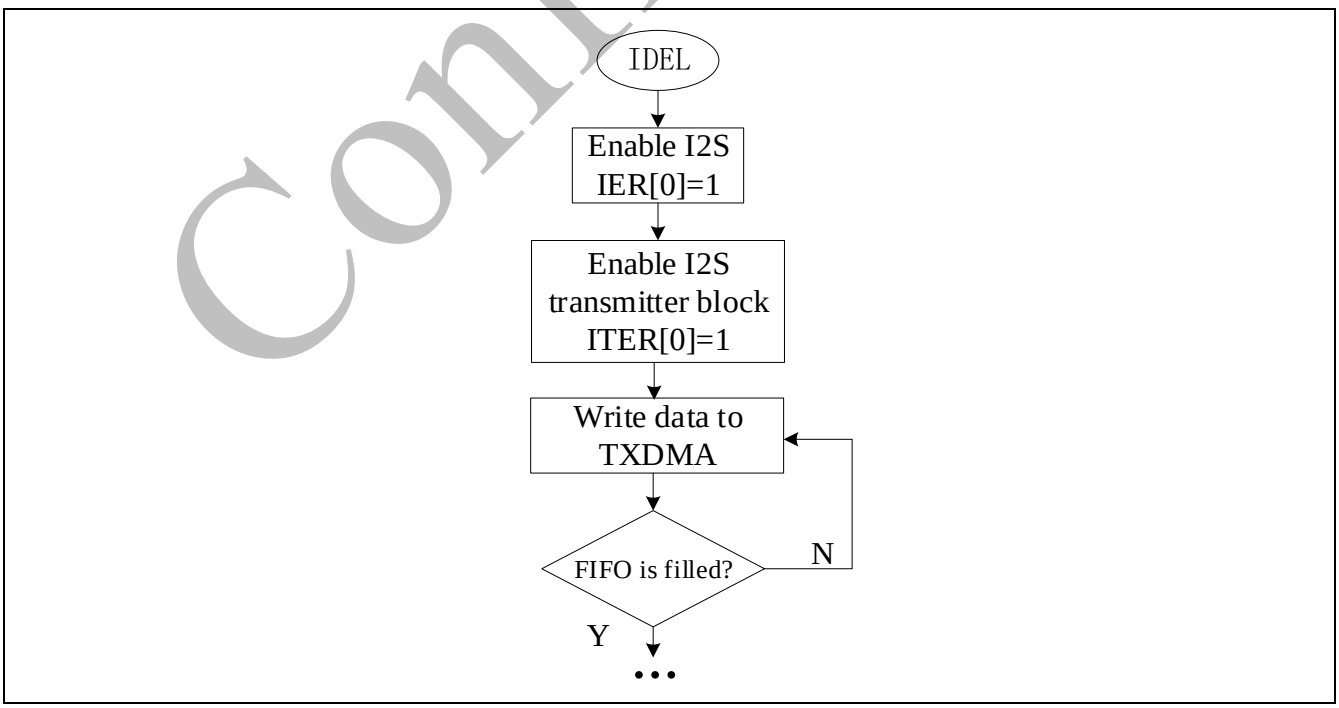


Figure 3-92 TX DMA Mode Flow Chart

Master Mode

To program I2S when it is a transmitter in master mode, complete the following steps:

1. Complete steps of the previous procedure (transmitter, slave mode).
2. Enable the I2S Clock Generation block by writing a 1 in bit 0 (CLKEN) of the Clock Enable Register (CER).

Single Channel transmit Mode

Refer to Figure 3-88. If I2S wants to transmit data by using left channel, CPU just need push data into LTHR or TXDMA, and TXDMA just pushes data into left FIFO.

If I2S wants to transmit data by using right channel, CPU just need push data into RTHR or TXDMA, and TXDMA just pushes data into right FIFO.

Single Channel receive Mode

Refer to Figure 3-89. If I2S wants to receive data by using left channel, CPU just needs read data from LRBR or RXDMA, and RXDMA just reads data from left FIFO.

If I2S wants to transmit data by using right channel, CPU just needs read data from RRBR or RXDMA, and RXDMA just reads data from right FIFO.

3.15.5 Transaction Example

In I2S, serial data is transmitted in two's complement format with the most significant bit (MSB) first. This means that the transmitter and receiver can have different word lengths, and neither the transmitter nor receiver needs to know what size words the other can handle. If the word being transferred is too large for the receiver, the least significant bits (LSB) are truncated. Similarly, if the word size is less than what the receiver can handle, the data is zero padded.

The word select line is used to time the multiplexed data streams. For instance, when ws is low, the word being transferred is left stereo data; when ws is high, the word being transferred is right stereo data. For standard I2S formats, the MSB of a word is sent one sclk cycle after a ws change. Serial data sent by the transmitter can be synchronized with either the negative edge or positive edge of the sclk signal. However, the receiver must latch the serial data on the rising edge of sclk.

illustrates an example I2S transfer in which I2S is a slave. The IDLE state of Word Select line is 0. Whenever the WS line makes a transition to 1, it means that after the next transition (0->1), the data starts being received. Therefore, the I2S slave treats the transfer as a START condition. When the stereo data is completely latched (signaled by Word Select Line going 0 again, also treated as start of new data frame), the data is pushed into the internal FIFO.

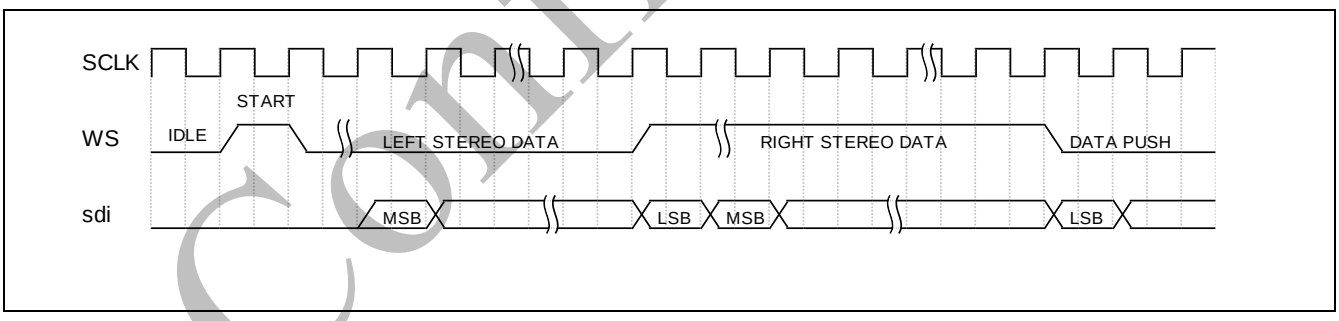


Figure 3-93 Transaction Waveform

3.15.6 I2S Register Map

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

Register	Offset	R/W	Description	Reset Value
I2S Base Address: I2S_BA				
IER	I2S_BA+0x00	R/W	I2S Enable Register	0x0000_0000
IRER	I2S_BA+0x04	R/W	I2S Receiver Block Enable Register	0x0000_0000
ITER	I2S_BA+0x08	R/W	I2S Transmitter Block Enable Register	0x0000_0000
CER	I2S_BA+0x0C	R/W	Clock Enable Register	0x0000_0000
CCR	I2S_BA+0x10	R/W	Clock Configuration Register	0x0000_0000
RXFFR	I2S_BA+0x14	W	Receiver Block FIFO Reset Register	0x0000_0000
TXFFR	I2S_BA+0x18	W	Transmitter Block FIFO Reset Register	0x0000_0000
LRBR	I2S_BA+0x20	R	Left Receive Buffer Register	0x0000_0000
LTHR	I2S_BA+0x20	W	Left Transmit Holding Register	0x0000_0000
RRBR	I2S_BA+0x24	R	Right Receive Buffer Register	0x0000_0000
RTHR	I2S_BA+0x24	W	Right Transmit Holding Register	0x0000_0000
RER	I2S_BA+0x28	R/W	Receive Enable Register	0x0000_0001
TER	I2S_BA+0x28	R/W	Transmit Enable Register	0x0000_0001
RCR	I2S_BA+0x30	R/W	Receive Configuration Register	0x0000_0002
TCR	I2S_BA+0x34	R/W	Transmit Configuration Register	0x0000_0002
ISR	I2S_BA+0x38	R	Interrupt Status Register	0x0000_0010
IMR	I2S_BA+0x3C	R/W	Interrupt Mask Register	0x0000_0033
ROR	I2S_BA+0x40	R	Receive Overrun Register	0x0000_0000
TOR	I2S_BA+0x44	R	Transmit Overrun Register	0x0000_0000
RFCR	I2S_BA+0x48	R/W	Receive FIFO Configuration Register	0x0000_0003
TFCR	I2S_BA+0x4C	R/W	Transmit FIFO Configuration Register	0x0000_0003
RFF	I2S_BA+0x50	W	Receive FIFO Flush Register	0x0000_0000
TFF	I2S_BA+0x54	W	Transmit FIFO Flush Register	0x0000_0000
RXDMA	I2S_BA+0x1C0	R	Receiver Block DMA Register	0x0000_0000
TXDMA	I2S_BA+0x1C8	W	Transmitter Block DMA Register	0x0000_0000

3.15.7 I2S Register Description

3.15.7.1 I2S Enable Register (IER)

Register	Offset	R/W	Description	Reset Value
IER	I2S_BA+0x00	R/W	I2S Enable Register	0x0000_0000

Bits	Description	
[31:20]	Reserved	Reserved.
[19]	left_not_right_stereo_rx	This bit works only when En_single_stereo_rx=1 1: only receive left channel data, Receive datas only push into left FIFO, and right FIFO is empty. 0: only receive right channel data ,Receive datas only push into right FIFO, and left FIFO is empty.
[18]	En_single_stereo_rx	1: enable single channel receive 0: disable single channel receive
[17]	left_not_right_stereo_tx	This bit works only when En_single_stereo_tx=1 1: only transmit datas in left FIFO, right channel is held low 0: only transmit datas in right FIFO, left channel is held low
[16]	En_single_stereo_tx	1: enable single channel transmit 0: disable single channel transmit
[15:1]	Reserved	Reserved.
[0]	IEN	I2S Enable bit. A disable on this bit overrides any other block or channel enables and flushes all FIFO. 0 = I2S Disabled 1 = I2S Enabled.

3.15.7.2 I2S Receiver Block Enable Register (IRER)

Register	Offset	R/W	Description	Reset Value
IRER	I2S_BA+0x04	R/W	I2S Receiver Block Enable Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	RXEN	Receiver block enable. 0 = disable receiver 1 = enable receiver

3.15.7.3 I2S Transmitter Block Enable Register (ITER)

Register	Offset	R/W	Description	Reset Value
ITER	I2S_BA+0x08	R/W	I2S Transmitter Block Enable Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	TXEN	Transmitter block enable. 0 = disable transmitter 1 = enable transmitter

3.15.7.4 Clock Enable Register (CER)

Register	Offset	R/W	Description	Reset Value
CER	I2S_BA+0x0C	R/W	Clock Enable Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	CLKEN	Clock generation enable/disable. This bit enables/disables the clock generation signals when I2S is a master. 0 = disable 1 = enable

3.15.7.5 Clock Configuration Register (CCR)

Register	Offset	R/W	Description	Reset Value
CCR	I2S_BA+0x10	R/W	Clock Configuration Register	0x0000_0000

Bits	Description	
[31:5]	Reserved	Reserved.
[4:3]	WSS	These bits are used to program the number of sclk cycles for which the word select line (ws) stays in the left or right sample mode: 0: 16 clock cycles 1: 24 clock cycles 2: 32 clock cycles The I2S Clock Generation block must be disabled (CER[0] = 0) prior to any changes in this value.
[2:0]	Reserved	Reserved.

3.15.7.6 Receiver Block FIFO Reset Register (RXFFR)

Register	Offset	R/W	Description	Reset Value
RXFFR	I2S_BA+0x14	W	Receiver Block FIFO Reset Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	RXFFR	Receiver FIFO Reset. Writing a 1 to this register flushes all the RX FIFO (this is a self clearing bit). Receiver Block must be disabled prior to writing this bit.

3.15.7.7 Transmitter Block FIFO Reset Register (TXFFR)

Register	Offset	R/W	Description	Reset Value
TXFFR	I2S_BA+0x18	W	Transmitter Block FIFO Reset Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	TXFFR	Transmitter FIFO Reset. Writing a 1 to this register flushes all the TX FIFO (this is a self clearing bit). Transmitter Block must be disabled prior to writing this bit.

3.15.7.8 Left Receive Buffer Register (LRBR)

Register	Offset	R/W	Description	Reset Value
LRBR	I2S_BA+0x20	R	Left Receive Buffer Register	0x0000_0000

Bits	Description	
[31:0]	LRBR	The left stereo data received serially from the receive channel input (sdi) is read through this register. If the RX FIFO is full and the two-stage read operation (for instance, a read from LRBR followed by a read from RRBR) is not performed before the start of the next stereo pair, then the new data is lost and an overrun interrupt occurs. (Data already in the RX FIFO is preserved.) **Note:** Before reading this register again, the right stereo data MUST be read from RRBR, or the status/interrupts will not be valid.

3.15.7.9 Left Transmit Holding Register (LTHR)

Register	Offset	R/W	Description	Reset Value
LTHR	I2S_BA+0x20	W	Left Transmit Holding Register	0x0000_0000

Bits	Description	
[31:0]	LTHR	The left stereo data to be transmitted serially through the transmit channel output (sdo) is written through this register. Writing is a two-stage process: (1) A write to this register passes the left stereo sample to the transmitter. (2) This MUST be followed by writing the right stereo sample to the RTHR register. Data should only be written to the FIFO when it is not full. Any attempt to write to a full FIFO results in that data being lost and an overrun interrupt being generated.

3.15.7.10 Right Receive Buffer Register (RRBR)

Register	Offset	R/W	Description	Reset Value
RRBR	I2S_BA+0x24	R	Right Receive Buffer Register	0x0000_0000

Bits	Description	
[31:0]	RRBR	The right stereo data received serially from the receive channel input (sdi) is read through this register. If the RX FIFO is full and the two-stage read operation (for instance, read from LRBR followed by a read from RRBR) is not performed before the start of the next stereo pair, then the new data is lost and an overrun interrupt occurs. (Data already in the RX FIFO is preserved.) Note: Prior to reading this register, the left stereo data MUST be read from LRBR, or the status/interrupts will not be valid.

3.15.7.11 Right Transmit Holding Register (RTHR)

Register	Offset	R/W	Description	Reset Value
RTHR	I2S_BA+0x24	W	Right Transmit Holding Register	0x0000_0000

Bits	Description	
[31:0]	RTHR	The right stereo data to be transmitted serially through the transmit channel output (sdo) is written through this register. Writing is a two-stage process: (1) A left stereo sample MUST first be written to the LTHR register. (2) A write to this register passes the right stereo sample to the transmitter. Data should only be written to the FIFO when it is not full. Any attempt to write to a full FIFO results in that data being lost and an overrun interrupt being generated.

3.15.7.12 Receive Enable Register (RER)

Register	Offset	R/W	Description	Reset Value
RER	I2S_BA+0x28	R/W	Receive Enable Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	RXCHEN0	Receive channel enable. This bit enables/disables a receive channel, independently of all other channels. On enable, the channel begins receiving on the next left stereo cycle. A global disable of I2S (IER[0] = 0) or the Receiver block (IRER[0] = 0) overrides this value. 0 = Disable 1 = Enable

3.15.7.13 Transmit Enable Register (TER)

Register	Offset	R/W	Description	Reset Value
TER	I2S_BA+0x28	R/W	Transmit Enable Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	TXCHEN0	Transmit channel enable. This bit enables/disables a transmit channel, independently of all other channels. On enable, the channel begins transmitting on the next left stereo cycle. A global disable of I2S (IER[0] = 0) or the Transmitter block (IRER[0] = 0) overrides this value. 0 = Disable 1 = Enable

3.15.7.14 Receive Configuration Register (RCR)

Register	Offset	R/W	Description	Reset Value
RCR	I2S_BA+0x30	R/W	Receive Configuration Register	0x0000_0002

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	WLEN	These bits are used to program the desired data resolution of the receiver and enables the LSB of the incoming left (or right) word to be placed in the LSB of the LRBR (or RRBR) register. 000 = Ignore word length 001 = 12 bit resolution 010 = 16 bit resolution 011 = 20 bit resolution 100 = 24 bit resolution 101 = 32 bit resolution Programmed data resolution must be less than or equal to 32. If the selected resolution is greater than the 32, the receive channel defaults back to 101. The channel must be disabled prior to any changes in this value (RER[0] = 0).

3.15.7.15 Transmit Configuration Register (TCR)

Register	Offset	R/W	Description	Reset Value
TCR	I2S_BA+0x34	R/W	Transmit Configuration Register	0x0000_0002

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	WLEN	These bits are used to program the desired data resolution of the receiver and enables the LSB of the incoming left (or right) word to be placed in the LSB of the LRBR (or RRBR) register. 000 = Ignore word length 001 = 12 bit resolution 010 = 16 bit resolution 011 = 20 bit resolution 100 = 24 bit resolution 101 = 32 bit resolution Programmed data resolution must be less than or equal to 32. If the selected resolution is greater than the 32, the receive channel defaults back to 101. The channel must be disabled prior to any changes in this value (RER[0] = 0).

3.15.7.16 Interrupt Status Register (ISR)

Register	Offset	R/W	Description	Reset Value
ISR	I2S_BA+0x38	R	Interrupt Status Register	0x0000_0010

Bits	Description	
[31:6]	Reserved	Reserved.
[5]	TXFO	Status of Data Overrun interrupt for the TX channel. Attempt to write to full TX FIFO. 0 = TX FIFO write valid 1 = TX FIFO write overrun
[4]	TXFE	Status of Transmit Empty Trigger interrupt. TX FIFO is empty. 1 = trigger level reached 0 = trigger level not reached
[3:2]	Reserved	Reserved.
[1]	RXFO	Status of Data Overrun interrupt for the RX channel. Incoming data lost due to a full RX FIFO. 0 = RX FIFO write valid 1 = RX FIFO write overrun
[0]	RXDA	Status of Receive Data Available interrupt. RX FIFO data available. 1 = trigger level reached 0 = trigger level not reached

3.15.7.17 Interrupt Mask Register (IMR)

Register	Offset	R/W	Description	Reset Value
IMR	I2S_BA+0x3C	R/W	Interrupt Mask Register	0x0000_0033

Bits	Description	
[31:6]	Reserved	Reserved.
[5]	TXFOM	Masks TX FIFO Overrun interrupt. 1 = masks interrupt 0 = unmasks interrupt
[4]	TXFEM	Masks TX FIFO Empty interrupt. 1 = masks interrupt 0 = unmasks interrupt
[3:2]	Reserved	Reserved.
[1]	RXFOM	Masks RX FIFO Overrun interrupt. 1 = masks interrupt 0 = unmasks interrupt
[0]	RXDAM	Masks RX FIFO Data Available interrupt. 1 = masks interrupt 0 = unmasks interrupt

3.15.7.18 Receive Overrun Register (ROR)

Register	Offset	R/W	Description	Reset Value
ROR	I2S_BA+0x40	R	Receive Overrun Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	RXCHO	Read this bit to clear the RX FIFO Data Overrun interrupt. 1 = RX FIFO write valid 0 = RX FIFO write overrun

3.15.7.19 Transmit Overrun Register (TOR)

Register	Offset	R/W	Description	Reset Value
TOR	I2S_BA+0x44	R	Transmit Overrun Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	TXCHO	Read this bit to clear the TX FIFO Data Overrun interrupt. 1 = TX FIFO write valid 0 = TX FIFO write overrun

3.15.7.20 Receive FIFO Configuration Register (RFCR)

Register	Offset	R/W	Description	Reset Value
RFCR	I2S_BA+0x48	R/W	Receive FIFO Configuration Register	0x0000_0004

Bits	Description	
[31:4]	Reserved	Reserved.
[3:0]	RXCHDT	These bits program the trigger level in the RX FIFO at which the Received Data Available interrupt is generated. Trigger Level = Programmed Value + 1 (for example, 1 to 8) Valid RXCHDT values: 0 to 7. If an illegal value is programmed, these bits saturate to 7. The channel must be disabled prior to any changes in this value (that is, RER[0] = 0).

3.15.7.21 Transmit FIFO Configuration Register (RFCR)

Register	Offset	R/W	Description	Reset Value
RFCR	I2S_BA+0x4C	R/W	Transmit FIFO Configuration Register	0x0000_0003

Bits	Description	
[31:4]	Reserved	Reserved.
[3:0]	RXCHDT	Transmit Channel Empty Trigger. These bits program the trigger level in the TX FIFO at which the Empty Threshold Reached Interrupt is generated. Trigger Level = TXCHET TXCHET values: 0 to 7. If an illegal value is programmed, these bits saturate to 7. The channel must be disabled prior to any changes in this value (that is, TERx[0] = 0).

3.15.7.22 Receive FIFO Flush Register (RFF)

Register	Offset	R/W	Description	Reset Value
RFF	I2S_BA+0x50	W	Receive FIFO Flush Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	RXCHFR	Receive Channel FIFO Reset. Writing a 1 to this register flushes an individual RX FIFO. (This is a self clearing bit.) RX channel or block must be disabled prior to writing to this bit.

3.15.7.23 Transmit FIFO Flush Register (TFF)

Register	Offset	R/W	Description	Reset Value
TFF	I2S_BA+0x54	W	Transmit FIFO Flush Register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	TXCHFR	Transmit Channel FIFO Reset. Writing a 1 to this register flushes channel's TX FIFO. (This is a self clearing bit.) TX channel or block must be disabled prior to writing to this bit.

3.15.7.24 Receiver Block DMA Register (RXDMA)

Register	Offset	R/W	Description	Reset Value
RXDMA	I2S_BA+0x1C0	R	Receiver Block DMA Register	0x0000_0000

Bits	Description	
[31:0]	RXDMA	Receiver Block DMA Register. Used to cycle repeatedly through the enabled receive channels, reading stereo data pairs.

3.15.7.25 Transmitter Block DMA Register (TXDMA)

Register	Offset	R/W	Description	Reset Value
TXDMA	I2S_BA+0x1C8	W	Transmitter Block DMA Register	0x0000_0000

Bits	Description	
[31:0]	TXDMA	Transmitter Block DMA Register. This register can be used to cycle repeatedly through the enabled Transmit channels (from lowest numbered to highest) to allow writing of stereo data pairs.

3.16 Analog-to-Digital Converter (ADC)

3.16.1 Overview

ADC contains one 12-bit successive approximation analog-to-digital converters (SAR A/D converter) with nine input channels. The A/D converters can be started by software, external pin (STADC/P5.2/P0.5) or PWM trigger.

3.16.2 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 eight single-end analog input channels, one band-gap input channel, one temperature input channel 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(P5.2/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

3.16.3 Block Diagram

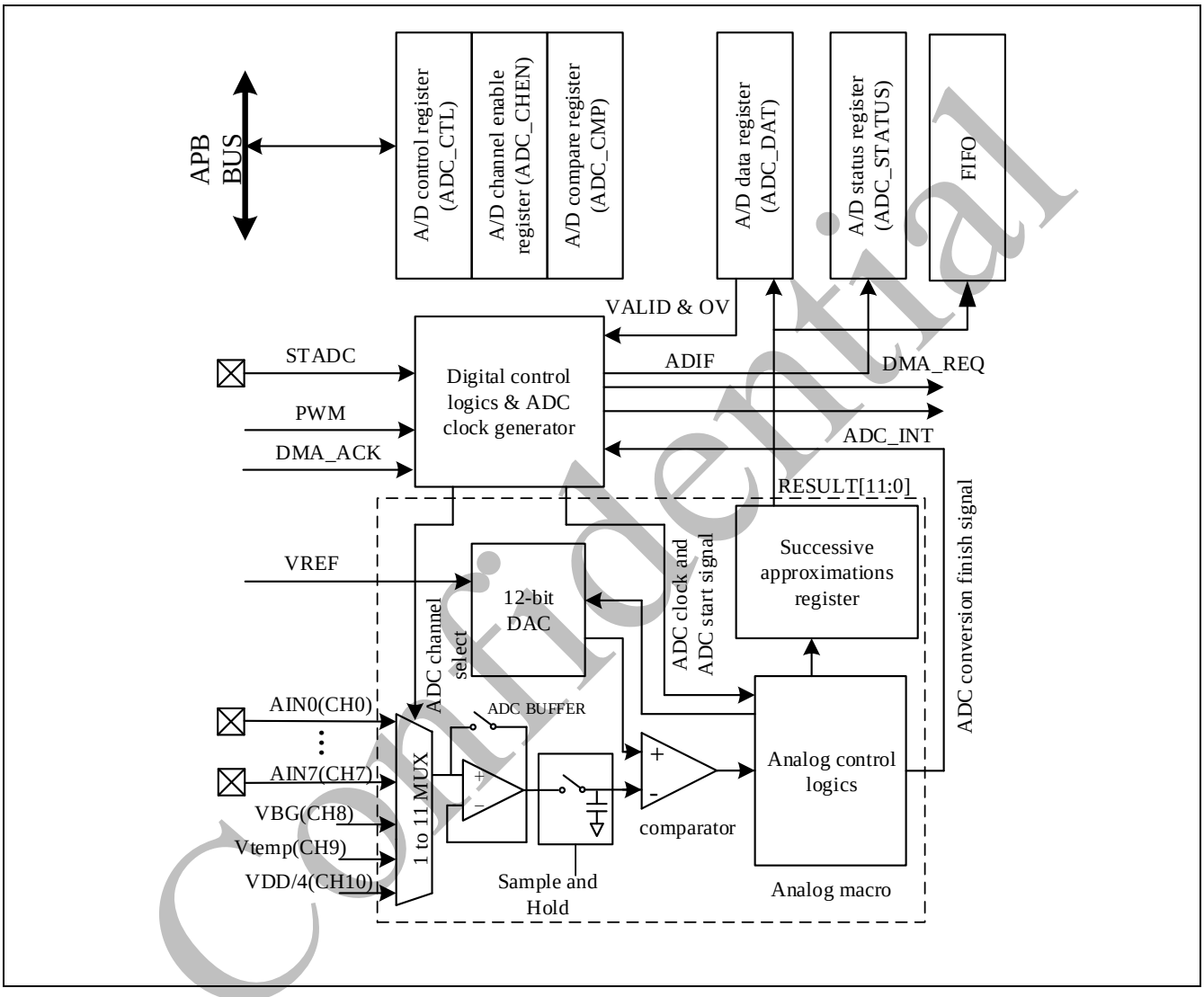


Figure 3-94 ADC Block Diagram

3.16.4 Basic Configuration

The ADC pin functions are configured in SYS_Px_MFP(x=0,1,2,3,4,5) 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=0,1,2,3,4,5) register.

The ADC peripheral clock can be enabled in Adccken(APB1_CLK_CTRL0[9]).

3.16.5 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 11 channels. ADC channels 0-7 are used to test the external input voltage. Channel 8 tests the internal reference voltage VBG-1.2V. Channel 9 is the temperature sensor channel. The linear temperature sensor can measure the temperature on the chip, and the measurement range is -40~85°C.

Channel 10 tests the VDD/4 voltage. VDD is the chip supply voltage.

3.16.5.1 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

3.16.5.2 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 3-95 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.

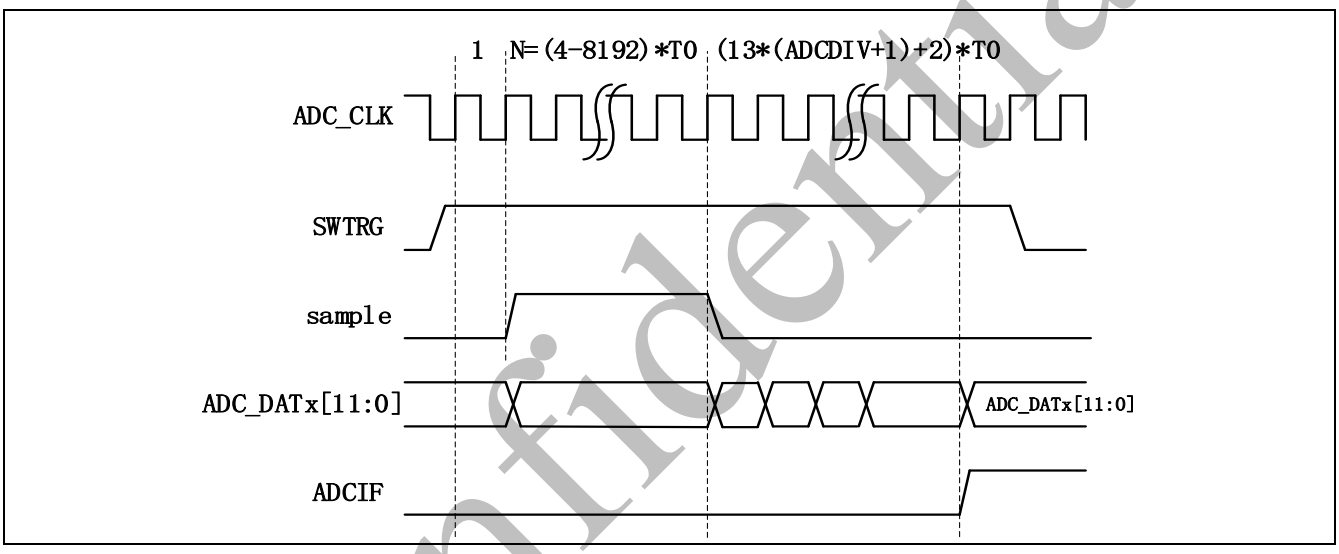


Figure 3-95 Single Mode Conversion Timing Diagram

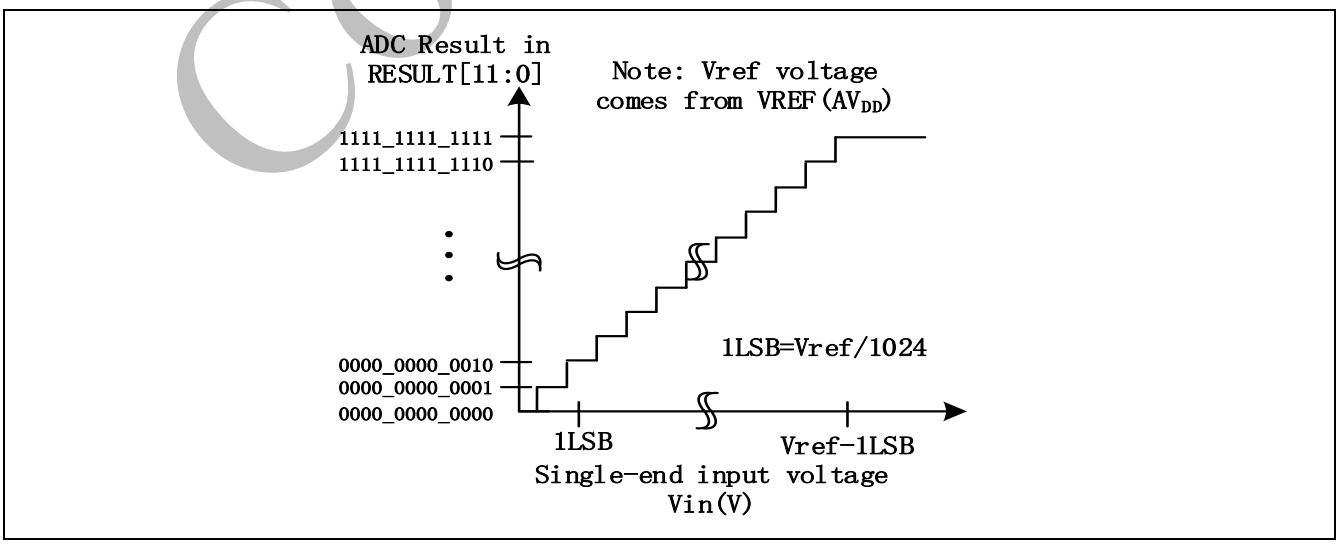


Figure 3-96 Conversion Result Mapping Diagram of ADC Single-end Input

3.16.5.3 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.

3.16.5.4 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.

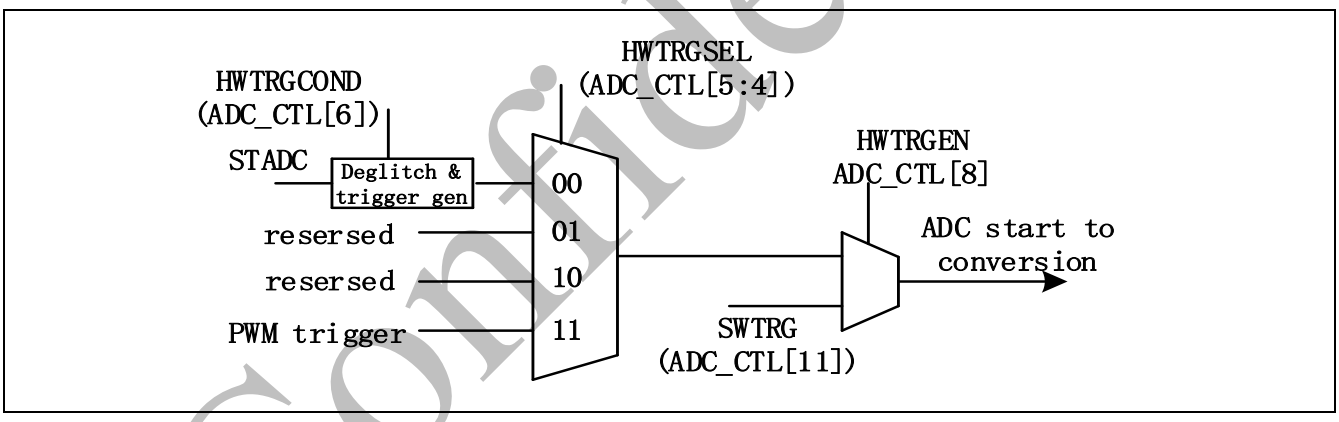


Figure 3-97 ADC Start Conversion Conditions

3.16.5.5 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 3-98 show detailed logic diagram.

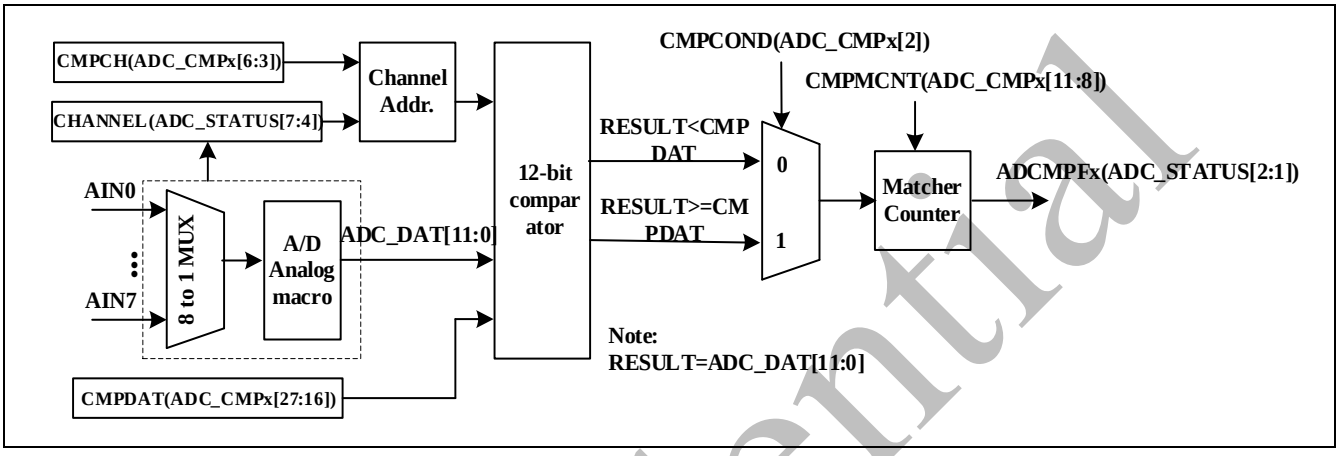


Figure 3-98 A/D Conversion Result Monitor Logics Diagram

3.16.5.6 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.

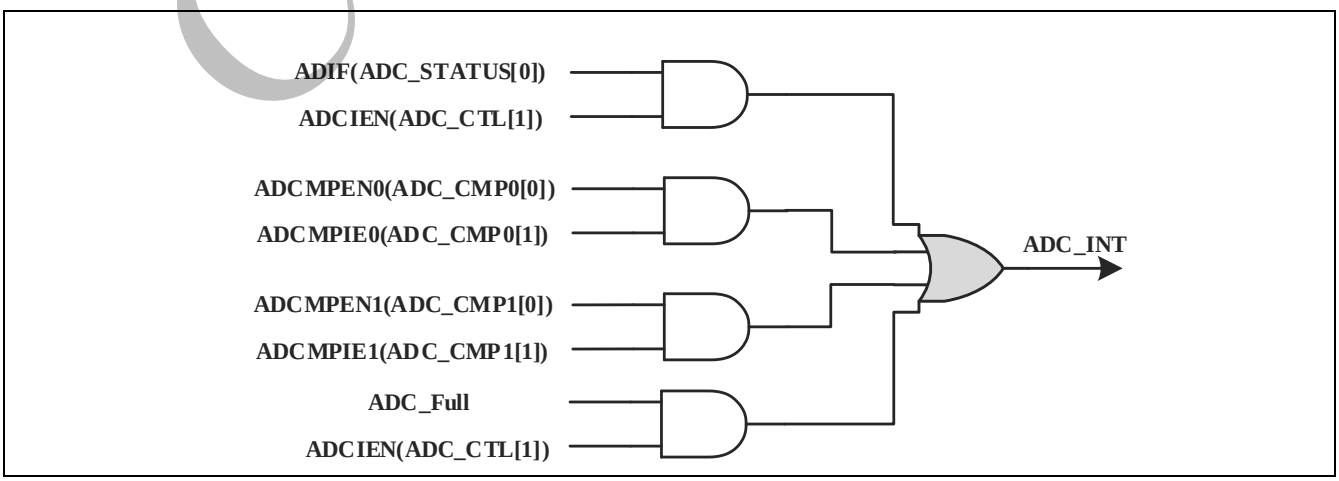


Figure 3-99 A/D Controller Interrupt

3.16.5.7 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.

3.16.5.8 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]).

3.16.5.9 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

3.16.5.10 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.

3.16.5.11 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.

3.16.5.12 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.

3.16.6 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 = 0x4000_5000				
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_LS_CTL	ADC_BA+0x5C	R/W	A/D Left Shift Control Register	0x0000_0000
ADC_SUB_CTL	ADC_BA+0x60	R/W	A/D Subtract offset Register	0x8000_0000
ADC_DATA_SUB	ADC_BA+0x64	R	A/D Subtract offset Data Register	0x0000_0000
ADC_FIFO_CTL	ADC_BA+0x68	R/W	A/D Subtract offset Data Register	0x0000_0000
ADC_BV_CTL	ADC_BA+0x6C	R/W	A/D Bias Voltage Control Register	0x0000_0100
FIFO_POP_DATA	ADC_BA+0x70	R	FIFO POP DATA Register	0x0000_0000

3.16.7 ADC Register Description

3.16.7.1 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 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 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.
[15:12]	Reserved	Reserved.
[11:0]	RESULT	A/D Conversion Result This field contains conversion result of ADC.

3.16.7.2 ADC Control Register (ADC_CTL)

Register	Offset	R/W	Description	Reset Value
ADC_CTL	ADC_BA+0x20	R/W	A/D Control Register	0x0000_0000

Bits	Description	
[31:12]	Reserved	Reserved.
[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.
[10:9]	Reserved	Reserved.
[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 or shunt mode. Others = Reserved. **Note:** Software should disable 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.

3.16.7.3 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	Reserved
[10]	CHEN10	VDD/4check Enable Bit 0 = VDD/4 check Disabled. 1 = VDD/4 check Enabled.
[9]	CHEN9	Temperature check Enable Bit 0 = Temperature check Disabled. 1 = Temperature check Enabled.
[8]	CHEN8	VBG_1.2 check Enable Bit 0 = VBG_1.2 check Disabled. 1 = VBG_1.2 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	Analog Input Channel 4 Enable Bit 0 = Channel 4 Disabled. 1 = Channel 4 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.

3.16.7.4 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.

3.16.7.5 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.

3.16.7.6 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.

3.16.7.7 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

3.16.7.8 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 .
[15:14]	Reserved	Reserved.
[14:12]	SEQ_CH_SEL	PWM sequence in ADC one channel select 000:adc channel 0 001:adc channel 1 010:adc channel 2 011:adc channel 3 100:adc channel 4 101:adc channel 5 110:adc channel 6 111:adc channel 7
[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	ADC delay time inserted before sequential 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 = PWM sequence in ADC one channel.
[1]	Reserved	Reserved
[0]	SEQEN	ADC Sequential 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 mode Disabled. 1 = ADC sequential mode Enabled.

3.16.7.9 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.

3.16.7.10 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]	Reserved	Reserved.
[26:24]	ICTL_VCM	ADC_ICTL_VCM, Default : 5
[23]	Reserved	Reserved.
[22:20]	ICTL_VREF	ADC_ICTL_VREF, Default : 5
[19:18]	Reserved	Reserved.
[17]	SEL_VREF	ADC_SEL_VREF, Default : 1
[16]	EN_BUFTST	ADC_EN_BUFTST, Default : 0
[15:12]	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

3.16.7.11 ADC Left Shift Control Register (ADC_LS_CTL)

Register	Offset	R/W	Description	Reset Value
ADC_LS_CTL	ADC_BA+0x5C	R/W	A/D Left Shift Control Register	0x0000_0000

Bits	Description	
[31:16]	data_left_shift	Read only. 12bit adc data left shift to be 16bit data,this need left_shift_en set
[15:1]	Reserved	Reserved
[0]	left_shift_en	0:MSB add 4bit0 ;{4'd0,fifo_pop_data[11:0]} 1:LSB add 4bit; {fifo_pop_data[11:0],4'd0 }

3.16.7.12 Subtract offset Control Register (ADC_SUB_CTL)

Register	Offset	R/W	Description	Reset Value
ADC_SUB_CTL	ADC_BA+0x60	R/W	A/D Subtract offset Register	0x8000_0000

Bits	Description	
[31:16]	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:0x8000
[15:1]	Reserved	Reserved
[0]	sub_bias_en	0:left shift function disable 1:left shift function enable

3.16.7.13 Subtract Offset Data Register (ADC_DATA_SUB)

Register	Offset	R/W	Description	Reset Value
ADC_DATA_SUB	ADC_BA+0x64	R	A/D Subtract offset Data Register	0x0000_0000

Bits	Description	
[31:17]	Reserved	Reserved
[16:0]	data_left_bias	When ADC valid,then result[11:0] or result1[11:0] data_valid,then left shift data[11:0] to data_left_sub[15:0],then subtract offset 0: it is positive number 1:it is negative number

3.16.7.14 ADC FIFO Control Register (ADC_FIFO_CTL)

Register	Offset	R/W	Description	Reset Value
ADC_FIFO_CTL	ADC_BA+0x68	R/W	A/D Subtract offset Data Register	0x0000_0000

Bits	Description	
[31:2]	Reserved	Reserved
[1]	fifo_full_or_half	Control DMA request in FIFO full state or half full state; 0:half full state 1:full state
[0]	fifo_en	ADC FIFO enable or disable 0:disable ADC FIFO 1:enable ADC FIFO

3.16.7.15 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	
[31：11]	Reserved	Reserved
[10:3]	adc_refgen_ctlv	Bias voltage resistance array control bit ; Default:8'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

3.16.7.16 FIFO POP DATA Register (FIFO_POP_DATA)

Register	Offset	R/W	Description	Reset Value
FIFO_POP_DATA	ADC_BA+0x70	R	FIFO POP DATA Register	0x00000000

Bits	Description	
[31:12]	Reserved	Reserved
[11:0]	fifo_pop_data	pop data from FIFO

3.17 Quadrature Decoder(QDEC)

3.17.1 Overview

PAN108 series has an integrated quadrature decoder that can automatically decode a pair of digital signals. The signals are typically provided by a speed/position feedback system mounted on a motor or trackball.

The signals, typically called A and B, are positioned 90 degrees out of phase, which results in a Gray code output. A Gray code is a sequence where only one bit changes on each count. This is essential to avoid glitches. It also allows detection of direction and relative position. A third optional signal, named index, is used as a reference to establish an absolute position once per rotation.

The QDEC can be used for waking up the chip as soon as there is any kind of movement from the external device connected to it.

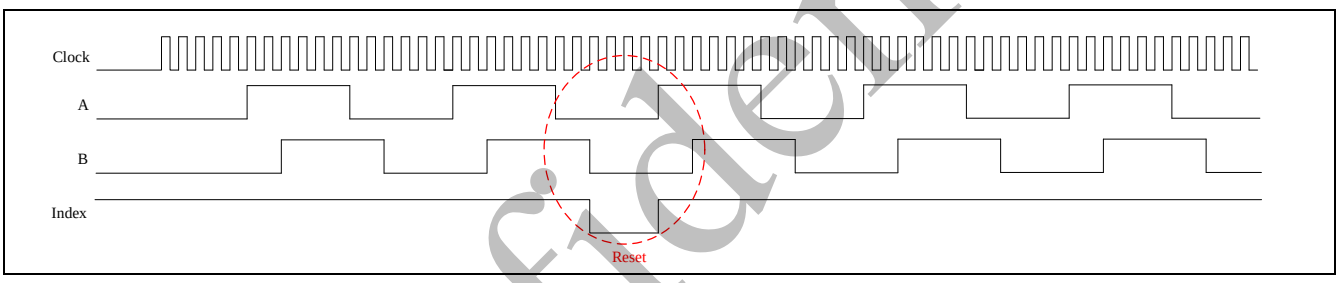


Figure 3-100 Waveform Diagram

3.17.2 Features

- Three-channel input, corresponding to three 16-bit signed counters respectively.
- Support short pulse filtering, and the filtering level length can be configured.
- Counter resolution of 1X, 2X, 4X the frequency of the A and B inputs.
- Interrupt flags include wake-up interrupt, INDEX valid interrupt, FIFO overflow interrupt, FIFO empty interrupt, counter overflow interrupt, counter underflow interrupt, illegal input interrupt.
- Two working modes: in Normal mode, the counter range is [-32768,32767]. In Event mode, the counter range is [-Const, Const].
- Waking up system in low power mode.

3.17.3 Block Diagram

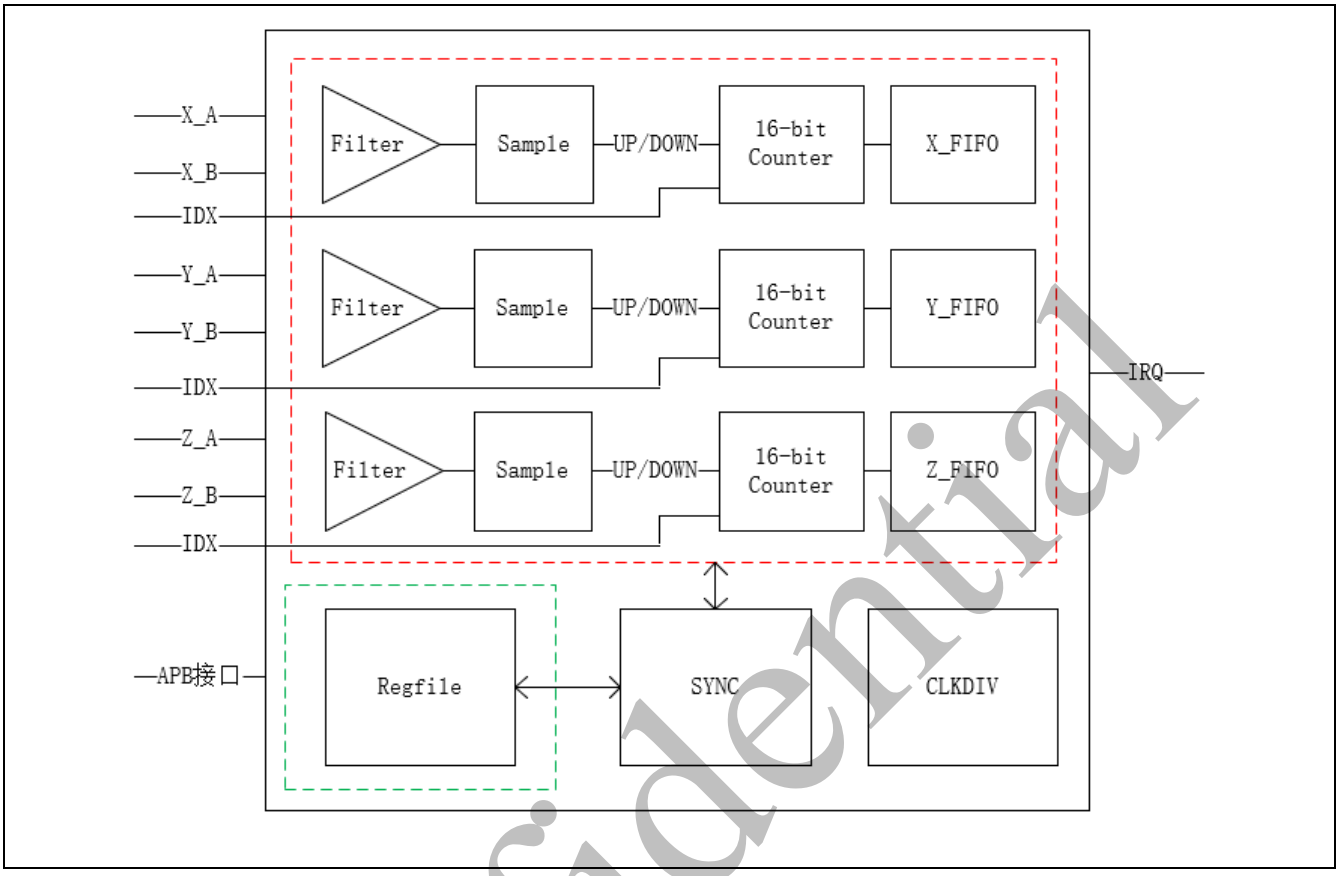


Figure 3-101 QDEC Block Diagram

3.17.4 Functional Description

3.17.4.1 Working Principle

The input of the quadrature decoder is two signals with a phase difference of 90 degrees. CHA can lead or lag the CHB signal by 90 degrees, indicating movement in different directions. The output of CHA and CHB presents the rule of Gray code, each time the Gray code is incremented, only one bit changes. If the A signal is ahead of the B signal, and the counter is incremented, as shown in Figure 3-102; otherwise, it is decremented, as shown in Figure 3-103.

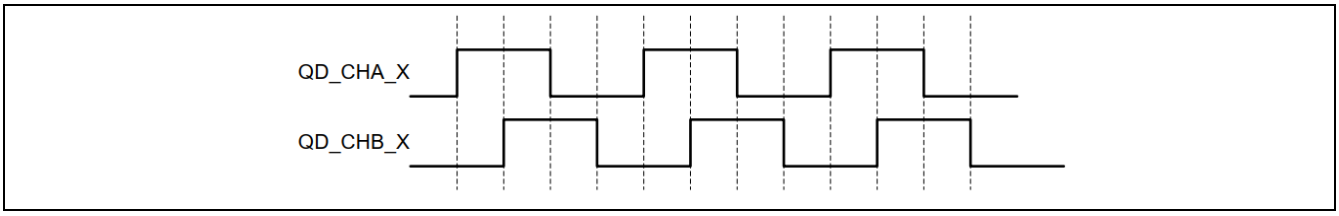


Figure 3-102 CHA Lead CHB Signal

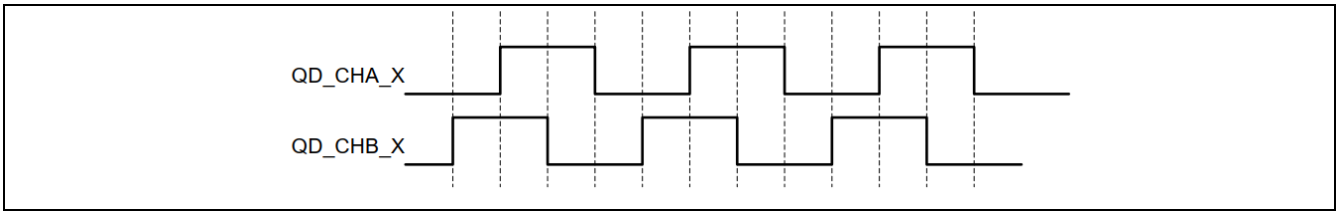
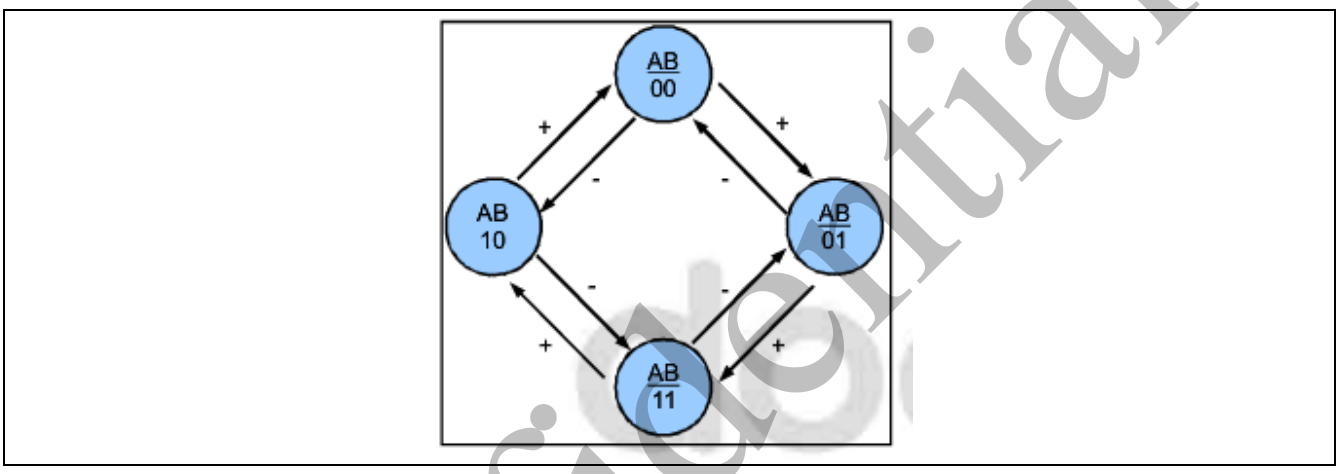


Figure 3-103 CHA Lag CHB Signal

Quadrature phase signals are typically decoded with a state machine and an up/down counter. A conventional decoder has four states, corresponding to all possible values of the A and B inputs. The brief state transition diagram is shown below. State transition marked with a "+" and "-" indicate increment and decrement operations on the quadrature phase counter.



For 4x resolution decoder, the quadrature phase counter changes by four counts. For 2x resolution decoder, the quadrature phase counter changes by two counts. For 1x resolution decoder, the quadrature phase counter changes by one counts.

The quadrature decoder counter starts counting transitions from 0 and could count in positive and negative directions to minimum and maximum limits.

When the quadrature decoder reaches the positive direction limit, the module generates a counter overflow interrupt and reloads the counter to 0.

When the quadrature decoder reaches the negative direction limit, the module generates a counter underflow interrupt and reloads the counter to 0.

The default range for counting is the following: 16-bit counter: -32768 to +32767

In event mode, the limit of counter is decided by register QD_EVT_THRES, the range for the counting is the following: -Constant to + Constant.

3.17.4.2 Interrupt

There are 7 kinds of interrupt states, which are wake-up interrupt, INDEX valid interrupt, FIFO overflow interrupt, FIFO read empty interrupt, counter overflow interrupt, counter

underflow interrupt and illegal input interrupt.

1. **Wake up interrupt**

   WK_EN register should be set before the system enters low power mode. 4 times or more valid transitions of the AB input signal will set the wake-up signal. After the system exits the low-power mode, QDEC triggers a wake-up interrupt. After the interrupt is triggered, the software should clears the wake-up interrupt flag bit and reset the WK_EN register.

2. **INDEX effective interrupt**

   The polarity of the INDEX signal is configurable, low-active by default. The interrupt is triggered when the index input is valid. After the interrupt is triggered, the hardware clears the counter and restarts counting.

3. **FIFO overflow interrupt**

   This interrupt is triggered when the FIFO is full and a write request is initiated. After this interrupt occurs, the hardware automatically suspends QDEC. The user need to check the QDEC configuration, reduce the resolution, reset the counter value, and restart QDEC. If QD_IDX_INT_EN=0, the hardware will not automatically suspend QDEC when FIFO data is full. If WK_EN is enabled, QDEC will not trigger the FIFO overflow interrupt.

4. **FIFO empty interrupt**

   This interrupt is triggered when there is no data in the FIFO. After an interrupt occurs, QDEC continues to run, and the software will read the value of the register after there is data in the FIFO.

5. **Counter overflow interrupt**

   This interrupt is triggered when the value of the 16-bit counter increments to 32767. After the interrupt is triggered, the hardware automatically clears the counter and restarts counting.

   If the event threshold function is enabled, the counter starts counting from 0, and this interrupt is triggered when the value of the counter increases to the set threshold. After the interrupt is triggered, the value of the counter will automatically reset to 0 and restart counting.

6. **Counter underflow interrupt**

   This interrupt is triggered when the value of the 16-bit counter decrements to -32768. After the interrupt is triggered, the hardware automatically clears the counter and restarts counting.

   If the event threshold function is enabled, the counter starts counting from 0, and this

interrupt is triggered when the value of the counter decreases to the lower limit of the threshold. After the interrupt is triggered, the value of the counter will automatically reset to 0 and restart counting.

7. **Illegal input interrupt**

   This interrupt is triggered when the sampler detects an illegal jump of the input signal. After the interrupt is triggered, the hardware automatically suspends QDEC. It is necessary to check whether the sensor input signal is normal, the sampling rate and the sampling frequency are reasonable. Then QDEC will be restarted. If QD_INV_INT_EN = 0, the hardware will not suspend QDEC.

3.17.5 QDEC Register Map

Register	offset	R/W	Description	Reset Value
QuadDec base addr：0x4001_a000-0x4001_aFFF				
QdecEnReg	QDEC_BASE+0x00	R/W	Enable register	0x0000_0002
QdecIntReg	QDEC_BASE+0x04	R/W	Interrupt status register	0x0000_0000
QdecIntRawReg	QDEC_BASE+0x08	R/W	Interrupt raw status register	0x0000_0000
QdecCtlReg	QDEC_BASE+0x0C	R/W	Qdec configuration register	0x000F_0400
QdecXcntReg	QDEC_BASE+0x10	R/W	Channel X counter register	0x0000_0000
QdecYcntReg	QDEC_BASE+0x14	R/W	Channel Y counter register	0x0000_0000
QdecZcntReg	QDEC_BASE+0x18	R/W	Channel Z counter register	0x0000_0000

3.17.6 QDEC Register Description

3.17.6.1 QdecEnReg

Register	offset	R/W	Description	Reset Value
QdecEnReg	QDEC_BASE+0x00	R/W	Enable register	0x0000_0002

Bits	Description	
[31:7]	Reserved	Reserved
[7]	EVT_MODE	1: Event mode 0: Normal mode In normal mode, the counter reaching 32767 or -32768 will trigger an interrupt. In event mode, the counter reaching threshold value will trigger an interrupt.
[6]	WK_EN	1:Wake up function is enabled 0:Wake up function is disabled. Reset value：0
[5]	IDX_EN	1: Index input is enabled 0: Index input is disabled Reset value：0
[4]	CHZ_EN	1:Channel Z is enabled 0:Channel Z is disabled Reset value：0
[3]	CHY_EN	1:Channel Y is enabled 0:Channel Y is disabled Reset value：0
[2]	CHX_EN	1:Channel x is enabled 0:Channel x is disabled Reset value：0
[1]	QD_FILT_EN	1:digital filter is enabled 0:digital filter is disabled Reset value：1
[0]	QD_EN	1:QuadRature decoder is enabled 0:QuadRature decoder is disabled QD_EN should be set at the last step. Decoder will be stop while detecting invalid AB transition. Reset value：0

3.17.6.2 QdecIntReg

Register	offset	R/W	Description	Reset Value
QdecIntReg	QDEC_BASE+0x04	R/W	Interrupt status register	0x0000_0000

Bits	Description	
[31:7]	Reserved	Reserved
[6]	QD_WK_INT	Wake up interrupt.Write 1 to clear this bit.This bit is only valid while WK_EN is set.
[5]	QD_IDX_INT	Valid index input interrupt. Write 1 to clear this bit.
[4]	QD_FIFO_UDF_INT	FIFO underflow interrupt. Write 1 to clear this bit.
[3]	QD_FIFO_OVF_INT	FIFO overflow interrupt. Write 1 to clear this bit.
[2]	QD_INV_INT	Invalid input interrutpt.Always 0 when interrupt is masked. Write 1 to clear this bit.
[1]	QD_UDF_INT	Counter underflow interrutpt.Always 0 when interrupt is masked. Write 1 to clear this bit.
[0]	QD_OVF_INT	Counter overflow interrutpt.Always 0 when interrupt is masked. Write 1 to clear this bit.

3.17.6.3 QdecIntRawReg

Register	offset	R/W	Description	Reset Value
QdecIntRawReg	QDEC_BASE+0x08	R/W	Interrupt raw status register	0x0000_0000

Bits	Description	
[31:7]	Reserved	Reserved
[5]	QD_IDX_INT_RAW	Raw valid index input interrupt. Write 1 to clear this bit.
[4]	QD_FIFO_UDF_INT_RAW	Raw FIFO underflow interrupt. Write 1 to clear this bit.
[3]	QD_FIFO_OVF_INT_RAW	Raw FIFO overflow interrupt. Write 1 to clear this bit.
[2]	QD_INV_INT_RAW	Raw invalid input flag. Caused by invalid A B input transition. Write 1 to clear this bit.
[1]	QD_UDF_INT_RAW	Raw counter underflow flag.Caused by counter underflow. Write 1 to clear this bit.
[0]	QD_OVF_INT_RAW	Raw counter overflow flag, caused by counter overflow. Write 1 to clear this bit.

3.17.6.4 QdecCtlReg

Register	offset	R/W	Description	Reset Value
QdecCtlReg	QDEC_BASE+0x0C	R/W	Qdec configuration register	0x000F_0400

Bits	Description	
[31:16]	QD_EVT_THRES	The number of events on either counter that need to be reached before an interrupt is generated. The vaue should be less than 32767. When an event interrupt is trigged, the counter will be reset. QD_INT_THRES=0, counter range = [-32768,32767] QD_INT_THRES=CONST, counter range = [-CONST,CONST]. QD_EVT_INT_EN = 1 and QD_EVT_THRES=0 must not be happened at the same time. Reset value:0x000F
[15:13]	Reserved	Reserved
[12]	QD_IDX_POL	1：index input signal is high valid 0：index input signal is low valid Reset value: 0x0
[11:9]	QD_FILT_T	If QD_FILT_EN is enabled, filtering is applied to all inputs. The filtered outputs do not change until QD_FILT_T successive samples of the input have the same value. 000-> 1 clk period, which means no digital filter 001-> 2 clk period 010-> 3 clk period … 111-> 8 clk period Reset value: {3b010}(3 clk period)
[8:7]	QD_SAMP_RES	Counter resolution 00->1x resolution 01->2x resolution 10->4x resolution 11->reserved Reset value:0x0
[6]	reserved	Reserved
[5]	QD_IDX_INT_EN	1: Valid index input interrupt enabled 0: Valid index input interrupt disabled Reset value：0x0
[4]	QD_FIFO_UDF_INT_EN	1: FIFO underflow interrupt enabled 0: FIFO underflow interrupt disabled Reset value：0x0
[3]	QD_FIFO_OVF_INT_EN	1: FIFO overflow interrupt enabled 0: FIFO overflow interrupt disabled Reset value：0x0
[2]	QD_INV_INT_EN	1: Invalid input interrupt enabled 0: Invalid input interrupt disabled Reset value：0x0
[1]	QD_UDF_INT_EN	1: Counter underflow interrupt enabled

		0: Counter underflow interrupt disabled Reset value：0x0
[0]	QD_OVF_INT_EN	1: Counter overflow interrupt enabled 0: Counter overflow interrupt disabled Reset value：0x0

3.17.6.5 QdecXcntReg

Register	offset	R/W	Description	Reset Value
QdecXcntReg	QDEC_BASE+0x10	R/W	Channel x counter	0x0000_0000

Bits	Description		
[31:17]	Reserved		Reserved
[16]	QD_XCNT_CLR	RW	Writing 1 to this bit clears channel x counter to zero. Hardware will clear this this bit after reseting counter. Reset value:0
[15:0]	QD_XCNT	R	Contains a signed value of the events. Zero when channel is disabled. Reset value:0

3.17.6.6 QdecYcntReg

Register	offset	R/W	Description	Reset Value
QdecYcntReg	QDEC_BASE+0x14	R/W	Channel y counter	0x0000_0000

Bits	Description		
[31:16]	Reserved		Reserved
[16]	QD_YCNT_CLR	RW	Writing 1 to this bit clears channel y counter to zero. Hardware will clear this this bit after reseting counter
[15:0]	QD_YCNT	R	Contains a signed value of the events, zero when channel is disabled

3.17.6.7 QdecZcntReg

Register	offset	R/W	Description	Reset Value
QdecZcntReg	QDEC_BASE+0x18	R/W	Channel z counter	0x0000_0000

Bits	Description		
[31:16]	Reserved		Reserved
[16]	QD_ZCNT_CLR	RW	Writing 1 to this bit clears channel z counter to zero. Hardware will clear this this bit after reseting counter
[15:0]	QD_ZCNT	R	Contains a signed value of the events, zero when channel is disabled

3.18 Serial Peripheral Interface (SPI)

3.18.1 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 SSPCR0 and SSPCR1.

Four individually maskable interrupt outputs are generated:

- SSPTXINTR requests servicing of the transmit buffer
- SSPRXINTR requests servicing of the receive buffer
- SSPRORINTR indicates an overrun condition in the receive FIFO
- SSPRTINTR indicates that a timeout period expired while data was present in the receive FIFO

A single combined interrupt, SSPINTR 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.

3.18.2 Feature

- Support 2 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.

3.18.3 Feature Description

3.18.3.1 Block Diagram

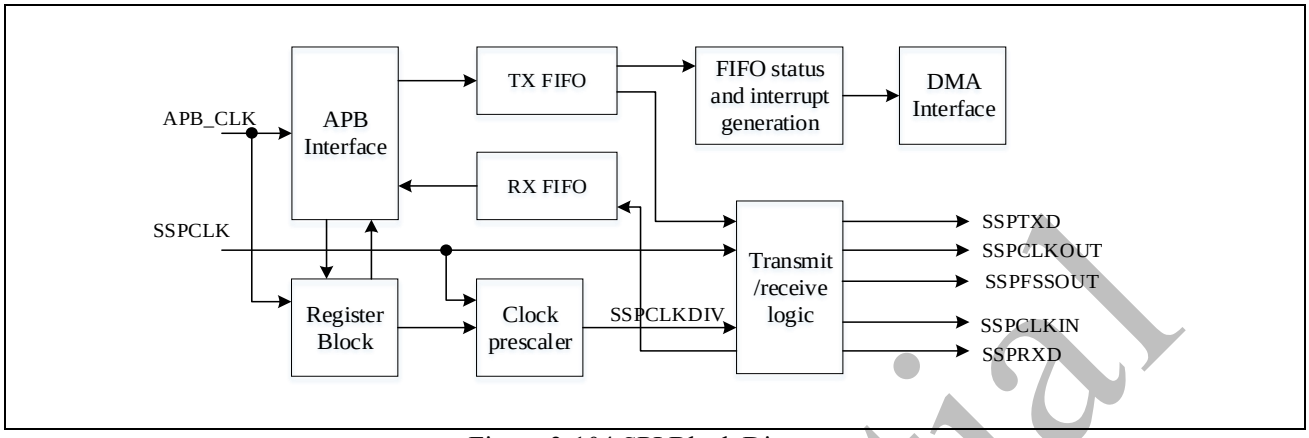


Figure 3-104 SPI Block Diagram

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

3.18.3.2 Function

This chapter describes the functional behavior of SSP Synchronous Serial Port in detail.

The functions of the SSP are described 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

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.

Register block

The register block stores data written or to be read across the AMBA APB interface.

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 SSPCPSR register, to divide SSPCLK by a factor of 2 to 254 in steps of two. By not utilizing the least significant bit of the SSPCPSR 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 SSPCR0 control register, to give the final master output clock SSPCLKOUT.

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 dss(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.

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 dss 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.

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 described 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.

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, SSPTXINTR and SSPRXINTR, are separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels.

DMA interface

The PrimeCell SSP provides an interface to connect to a DMA controller.

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.

3.18.3.3 SSP operation

The operation of the SSP is described in the following sections:

- Interface reset
- Configuring the SSP
- Enable SSP operation
- Clock ratios
- Programming the SSPCR0 Control Register
- Programming the SSPCR1 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

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.

Configuring the SSP

Following reset, the SSP logic is disabled and must be configured when in this state.

Control registers SSPCR0 and SSPCR1 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 SSPCPSR.

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.

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 SSPCPSR register has to be programmed with a value of 2 and the SCR[7:0] field in the SSPCR0 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 SSPCPSR register can be programmed with a value of 2 and the SCR[7:0] field in the SSPCR0 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]

Programming the SSPCR0 Control Register

The SSPCR0 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 (SCR) value, in conjunction with the SSPCPSR clock prescale divisor value (CPSDVSR), 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 DSS bits.

Bit phase and polarity, applicable to Motorola SPI format only, are programmed through the SPH and SPO bits.

Programming the SSPCR1 Control Register

The SSPCR1 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 SSPCR1 register master or slave selection bit (MS) to 0, which is the default value on reset.

Setting the SSPCR1 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 SSPCR1 slave mode SSPTXD output disable bit (SOD). 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.

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 CPSDVSR from 2 to 254, which is programmed in SSPCPSR. The clock is further divided by a value from 1 to 256, which is 1 + SCR, where SCR is the value programmed in SSPCR0.

The frequency of the output signal bit clock SSPCLKOUT is defined below:

FSSPCLKOUT=FSSPCLK/(CPSDVR x (1+SCR))

For example, if SSPCLK is 3.6864MHz, and CPSDVSR = 2, then SSPCLKOUT has a frequency range from 7.2kHz to 1.8432MHz.

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.

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, responds 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.

Texas Instruments synchronous serial frame format

Figure 3-105 shows the Texas Instruments synchronous serial frame format for a single transmitted frame, tx_lsb=0 and rx_lsb=0.

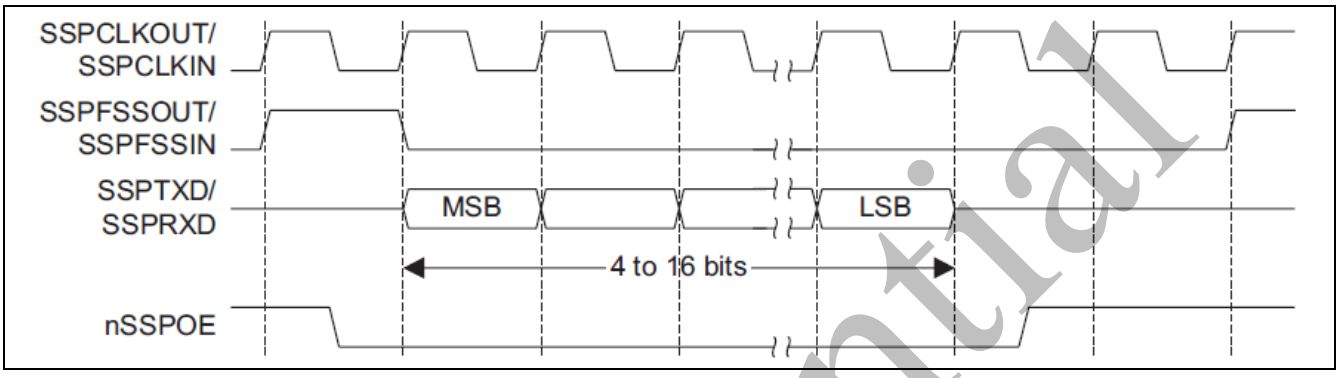


Figure 3-105 Texas Instruments synchronous serial frame format (single transfer), tx_lsb=0, rx_lsb=0

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 3-106 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

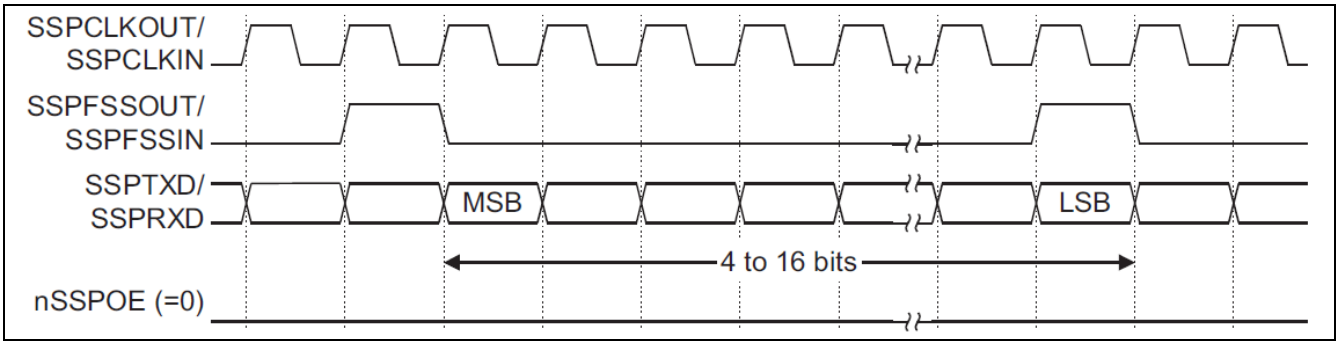


Figure 3-106 TI synchronous serial frame format (continuous transfer), tx_lsb=0, rx_lsb=0

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 SPO and SPH bits within the SSPSCR0 control register.

SPO, clock polarity

When the SPO clock polarity control bit is LOW, it produces a steady state low value on the SSPCLKOUT pin. If the SPO clock polarity control bit is HIGH, a steady state high value is placed on the SSPCLKOUT pin when data is not being transferred.

SPH, clock phase

The SPH 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 SPH phase control bit is LOW, data is captured on the first clock edge transition. If the SPH clock phase control bit is HIGH, data is captured on the second clock edge transition. For continuous back-to-back SPI transactions, if SPH=0, SSPSSFOUT turns high for one cycle of SSPCLKOUT for each SPI transaction; if SPH=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 SPO^SPH=0, master and slave both sample SSPRXD on posedge of SSPCLKOUT/SSPCLKIN, and drive SSPTXD on negedge of SSPCLKOUT/SSPCLKIN.

If SPO^SPH=1, master and slave both sample SSPRXD on negedge of SSPCLKOUT/SSPCLKIN, and drive SSPTXD on posedge of SSPCLKOUT/SSPCLKIN.

Motorola SPI Format with SPO=0, SPH=0

Continuous transmission signal sequences for Motorola SPI format with SPO=0, SPH=0, tx_lsb=0 and rx_lsb=0 are shown in Figure 3-107.

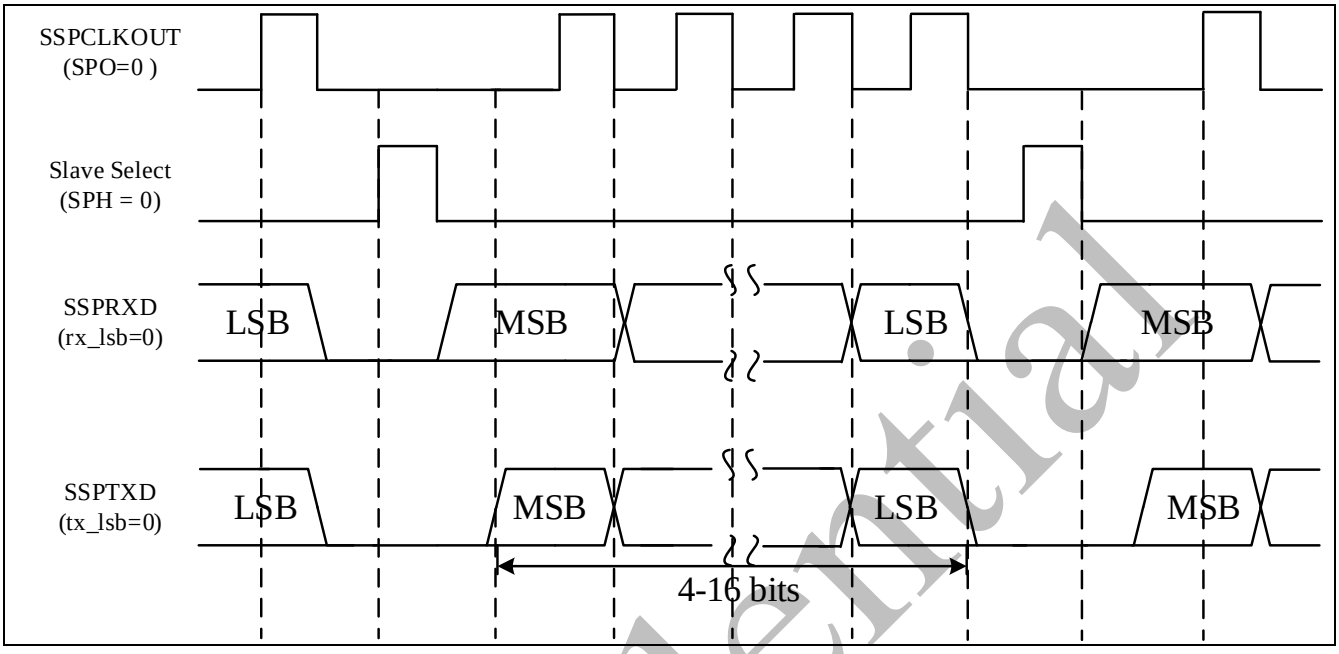


Figure 3-107 Motorola SPI frame format (continuous transfer) with SPO=0 and SPH=0, tx_lsb=0, rx_lsb=0

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 nsspoe 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 nSSPOE 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 SPH 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.

Motorola SPI Format with SPO=0, SPH=1

The transfer signal sequence for Motorola SPI format with SPO=0, SPH=1, tx_lsb=0 and rx_lsb=1 is shown in Figure 3-108.

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 nsspoe 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)

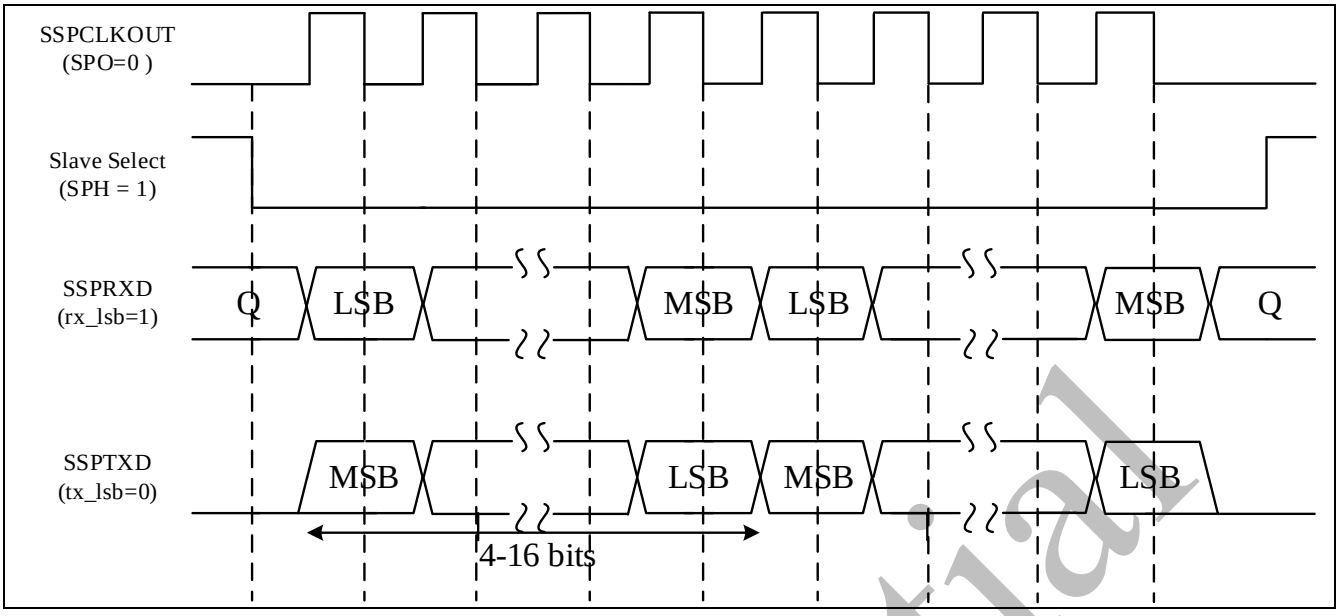


Figure 3-108 Motorola SPI frame format with SPO=0 and SPH=1, tx_lsb=0, rx_lsb=1

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 nSSPOE 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.

Motorola SPI Format with SPO=1, SPH=0

The transmission signal sequences for Motorola SPI format with SPO=1, SPH=0, tx_lsb=1 and rx_lsb=1 are shown in Figure 3-109.

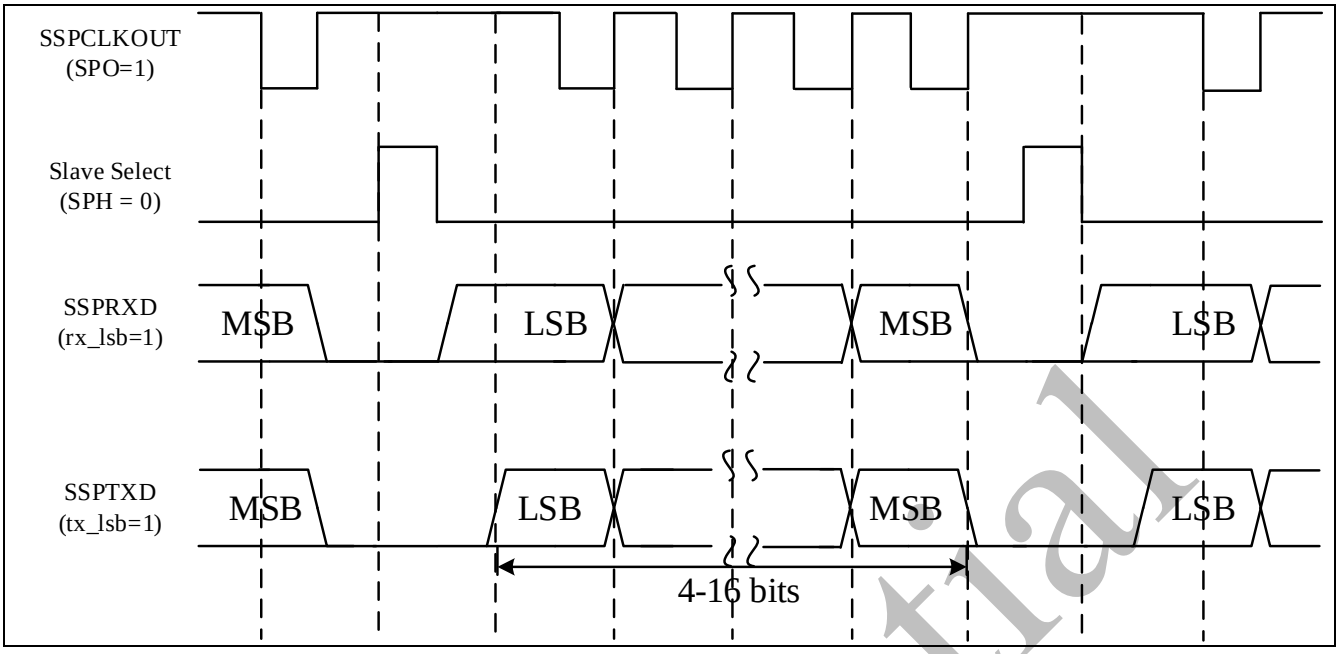


Figure 3-109 Motorola SPI frame format with SPO=1 and SPH=0, tx_lsb=1, rx_lsb=1

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 nsspoe 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 nSSPOE 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 SPH 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.

Motorola SPI Format with SPO=1, SPH=1

The transfer signal sequence for Motorola SPI format with SPO=1, SPH=1, tx_lsb=1 and rx_lsb=0 is shown in Figure 3-110.

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 nsspoe 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 nSSPOE 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.

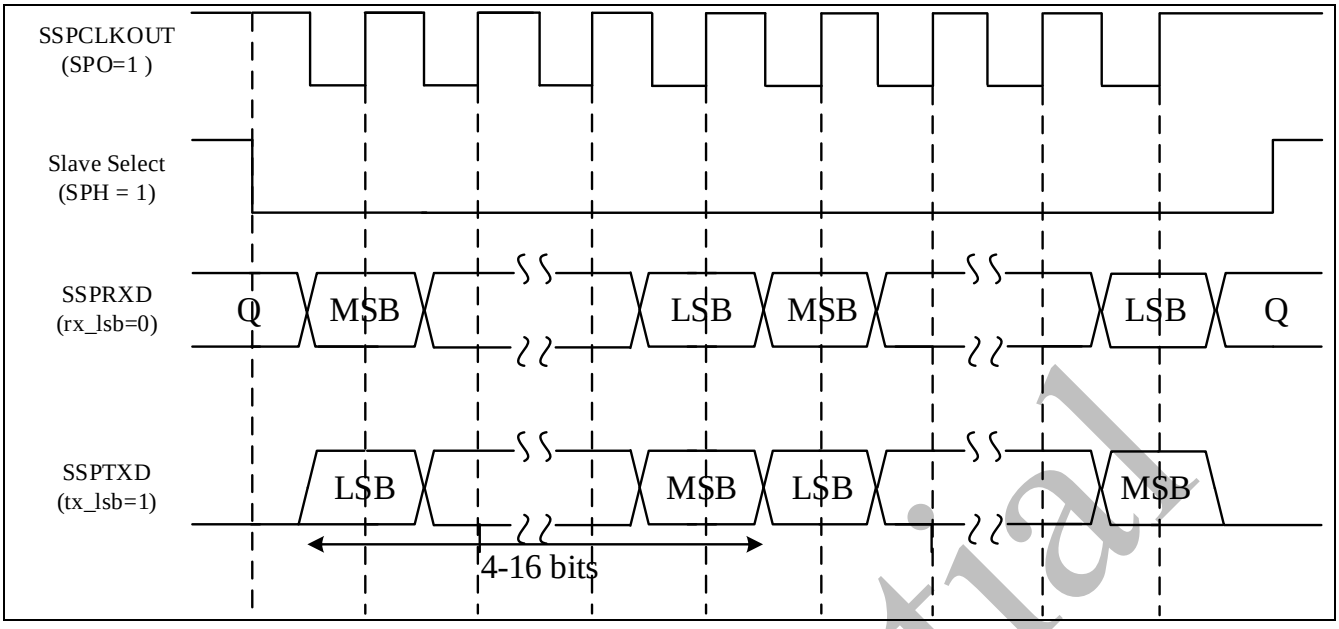


Figure 3-110 Motorola SPI frame format with SPO=1 and SPH=1, tx_lsb=1, rx_lsb=0

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 described 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.

National Semiconductor Microwire frame format

Figure 3-111 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, responds 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.

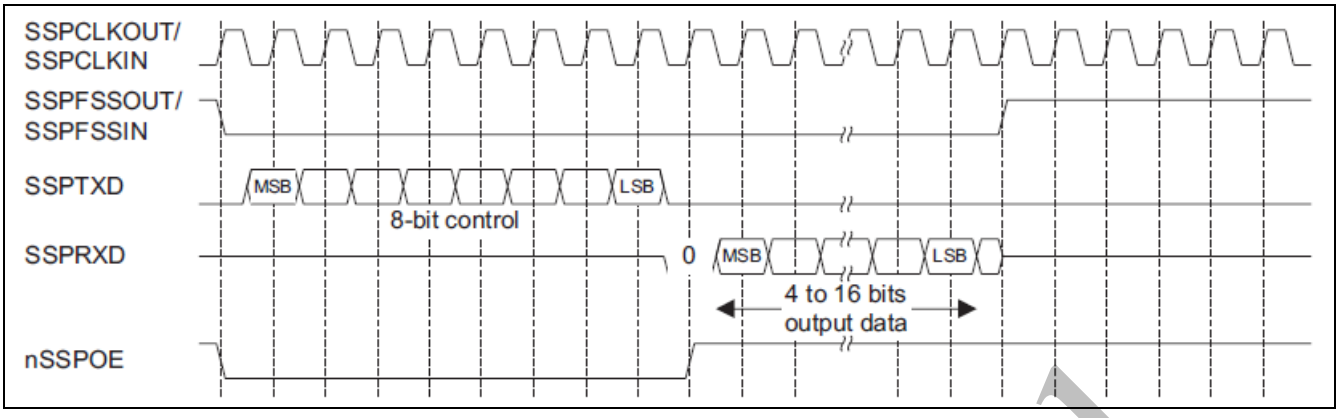


Figure 3-111 Microwire frame format (single transfer), tx_lsb=0 and rx_lsb=0

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 nsspoe 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 responds 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.

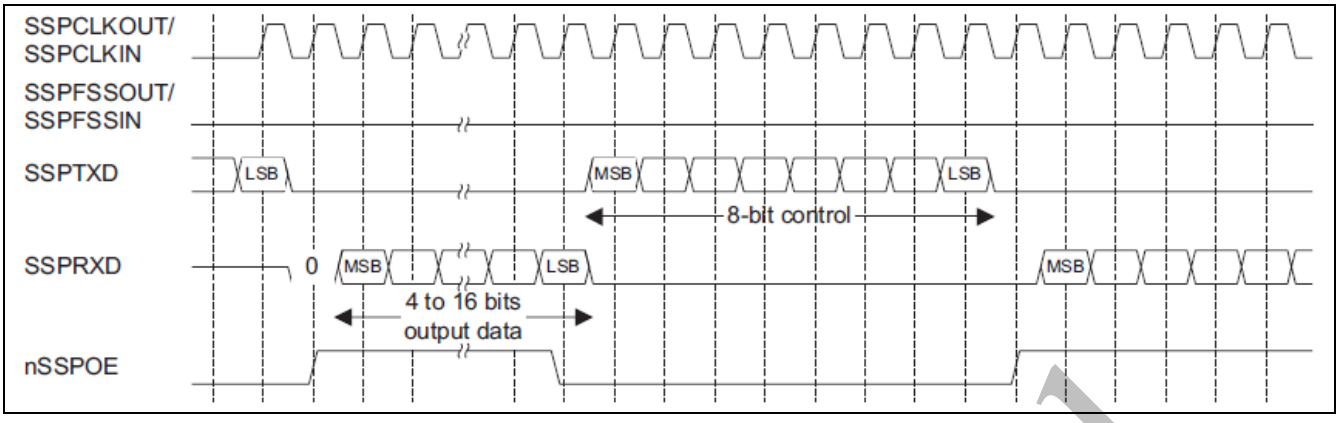


Figure 3-112 icrowire frame format (continuous transfers), tx_lsb=0 and rx_lsb=0

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 3-113 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.

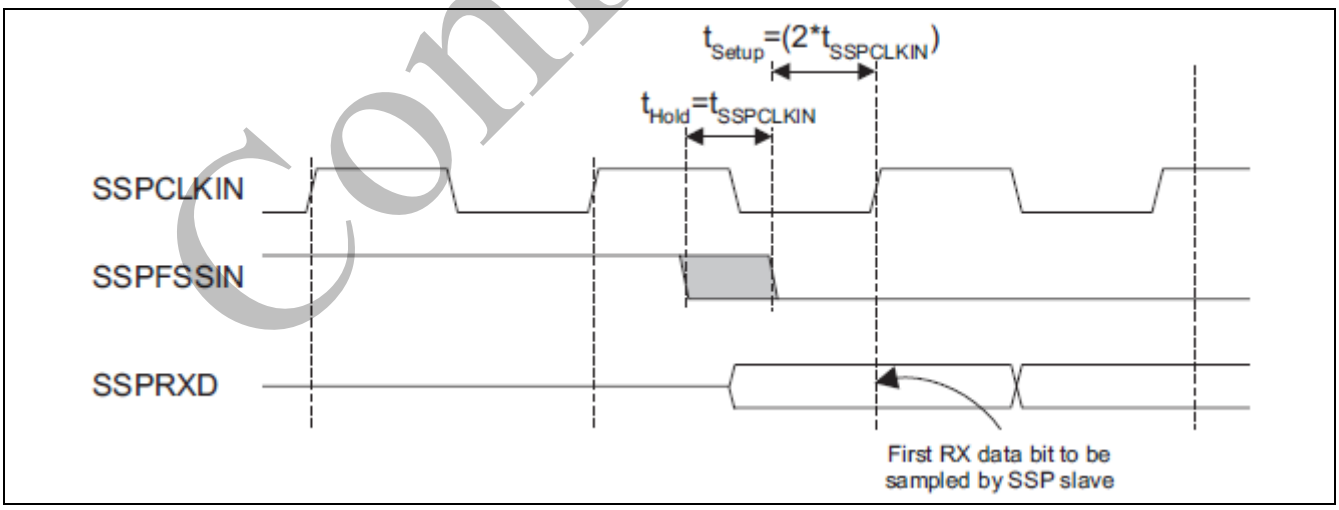


Figure 3-113 Microwire frame format, SSPFSSIN input setup and hold requirements

Examples of master and slave configurations

Figure 3-114, Figure 3-115 and Figure 3-116 show how the SSP peripheral can be connected to other synchronous serial peripherals, when it is configured as a master or slave.

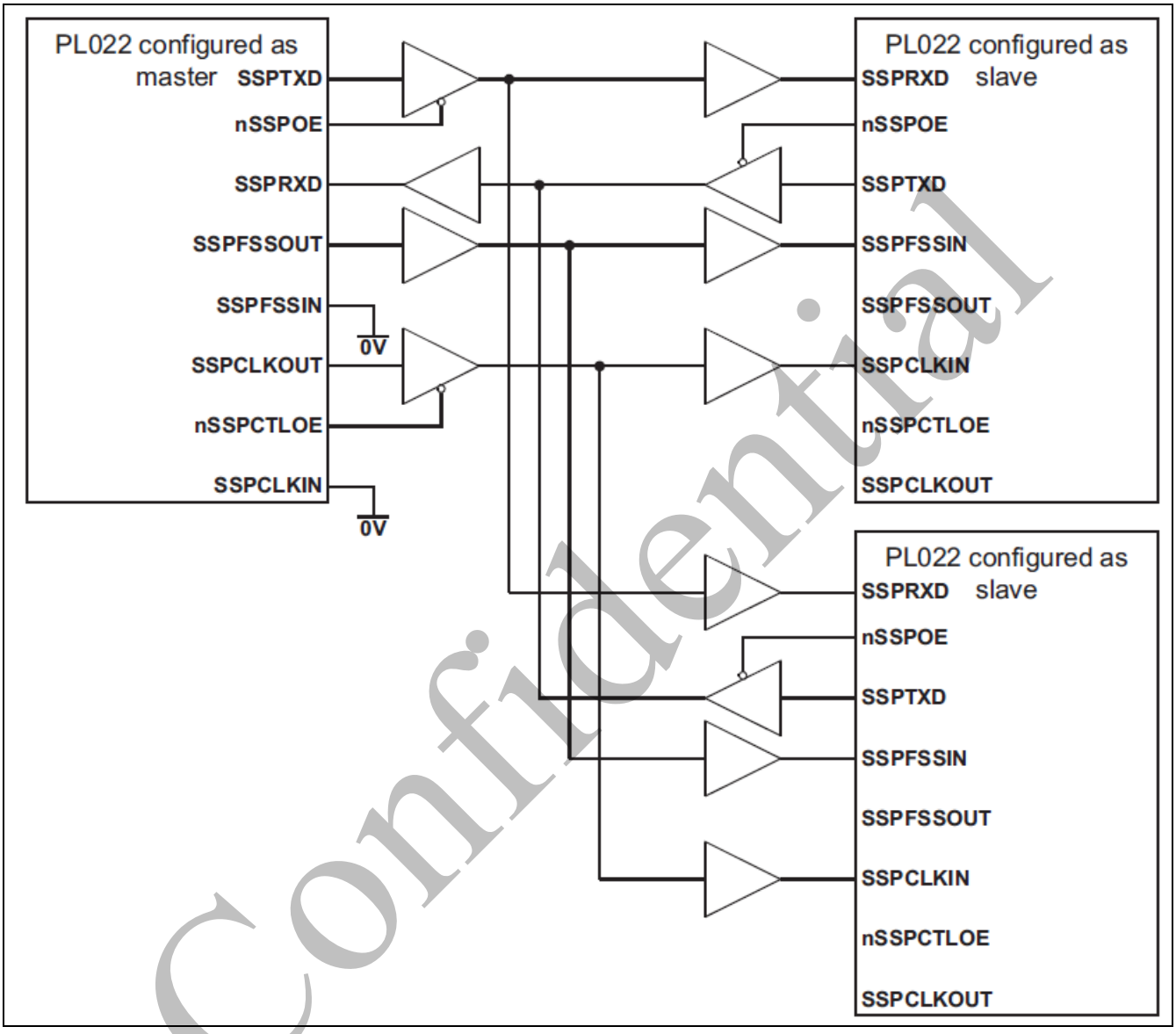


Figure 3-114 SSP master coupled to two slaves

Figure 3-114 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 response, only one slave drives its nSSPOE signal HIGH, thereby enabling its SSPTXD data onto the SSPRXD line of the master.

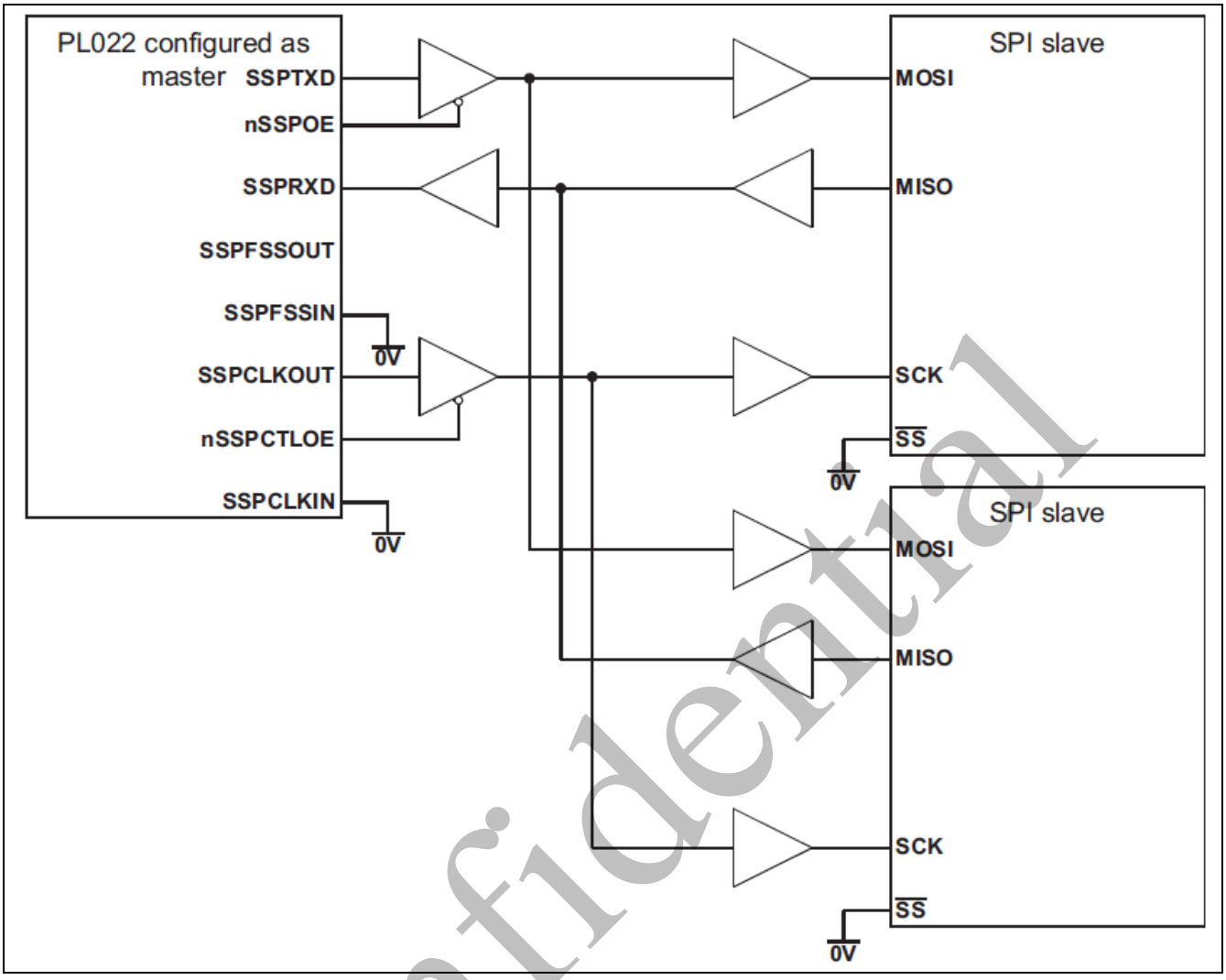


Figure 3-115 SSP master coupled to two slaves

Figure 3-115 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 response, only one slave drives its SPI MISO port onto the SSPRXD line of the master.

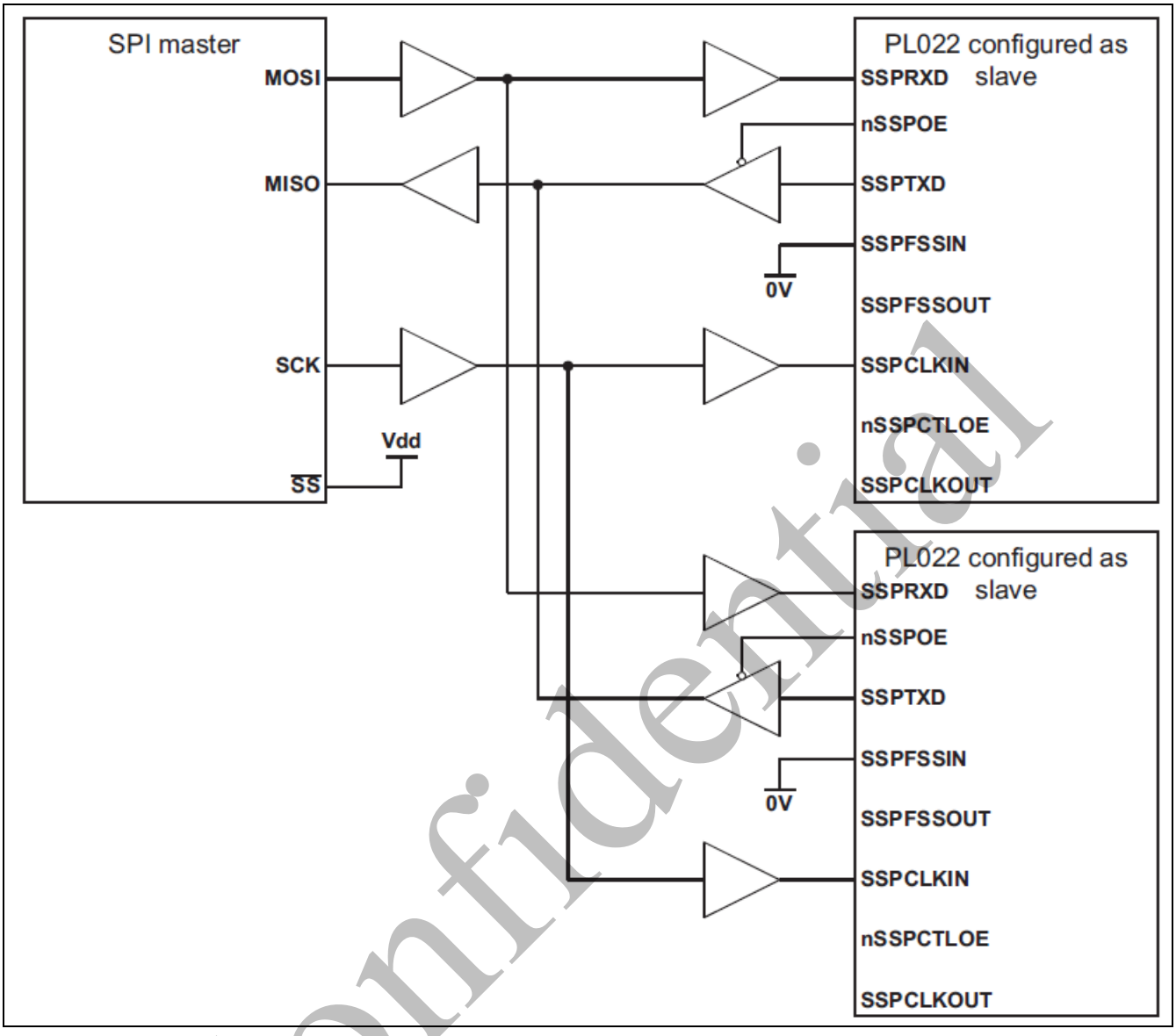


Figure 3-116 SPI master coupled to two SSP slaves

Figure 3-116 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 response, only one slave drives its nSSPOE signal LOW. This enables its SSPTXD data onto the MISO line of the master.

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, SSPDMACR. 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 described above. All request signals are deasserted if the SSP is disabled or the DMA enable signal is cleared.

Table 3-22 shows the trigger points for DMABREQ, for both the transmit and receive FIFOs.

Table 3-22 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 3-117 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.

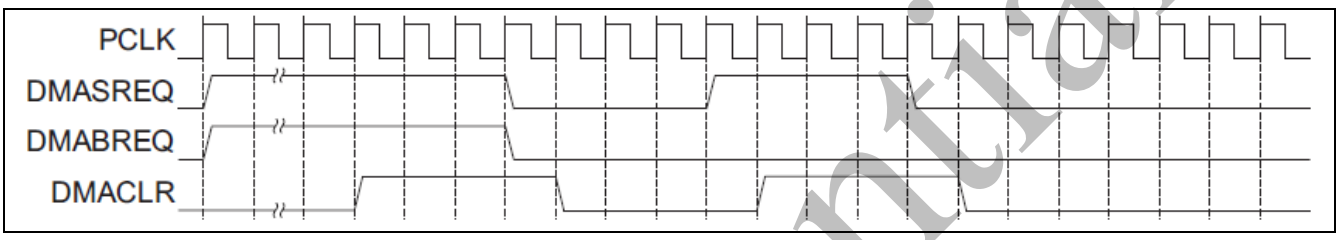


Figure 3-117 DMA transfer waveforms

Interrupts

There are five interrupts generated by SSP. The specific interrupt signals are described in Table 2. Four of these are individual, maskable, active-HIGH interrupts as follows:

- SSPRXINTR
- SSPTXINTR
- SSPRORINTR
- SSPRTINTR
- SSPINTR

You can mask each of the four individual maskable interrupts by setting the appropriate bits in the SSPIMSC 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 SSPTXINTR and SSPRXINTR have been separated from the status interrupts, so that data can be read or written in response to only the FIFO trigger levels.

The status of the individual interrupt sources can be read from SSPRIS and SSPMIS registers.

Table 3-23 Interrupt Signals

Port Name	Description	Asserted condition	Notes
SSPRXINTR	SPI receive FIFO service interrupt request	≥ 4 entries in RX FIFO	-
SSPTXINTR	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.
SSPRORINTR	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.
SSPRTINTR	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 SSPICR register.
SSPINTR	Combined single interrupt	Any of the four individual interrupts above are asserted and enabled	An OR function of the individual masked sources

3.18.4 SSP Register Map

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

Register	Offset	R/W	Description	Reset Value
SSP Base Address: SSP0_BA =0x4000_1000 SSP1_BA =0x4001_1000				
SSPCR0	SSPx_BA+0x00	R/W	Control Register 0	0x0000_0000
SSPCR1	SSPx_BA+0x04	R/W	Control Register 1	0x0000_0000
SSPDR	SSPx_BA+0x08	R/W	Receive FIFO(read) and Transmit FIFO Data Register	Configurable
SSPSR	SSPx_BA+0x0C	R	Status Register	0x0000_0003
SSPCPSR	SSPx_BA+0x10	R/W	Clock Prescale Register	0x0000_0000
SSPIMSC	SSPx_BA+0x14	R/W	Interrupt Mask Set and Clear Register	0x0000_0000
SSPRIS	SSPx_BA+0x18	R	Raw Interrupt Status Register	0x0000_0008
SSPMIS	SSPx_BA+0x1C	R	Masked Interrupt Status Register	0x0000_0000
SSPICR	SSPx_BA+0x20	W	Interrupt Clear Register	0x0000_0000
SSPDMACR	SSPx_BA+0x24	R/W	DMA Control Register	0x0000_0000

3.18.5 SSP Register Description

3.18.5.1 Control Register 0 (SSPCR0)

Register	Offset	R/W	Description	Reset Value
SSPCR0 x=0, 1	SSPx_BA+0x00	R/W	Control Register 0	0x0000_0000

Bits	Description	
[15:8]	SCR	Serial Clock Rate. The value SCR is used to generate the transmit and receive bit rate of the SSP. The bit rate is: $\frac{f_{\text{ssp\_apb\_clk}}}{CPSDVSR \times (1 + SCR)}$ where CPSDVSR is an even value from 2-254, programmed through the SSPCPSR register and SCR is a value from 0-255
[7]	SPH	Serial Clock Phase. For Motorola SPI frame format only 0 = Data are captured on the first edge of the serial clock. Slave Select pulse high between each spi bus data item. 1 = Data are captured on the second edge of the serial clock. Slave Select keep low for continuous transmission.
[6]	SPO	Serial Clock Polarity. For Motorola SPI frame format only 0 = Inactive state of serial clock is low 1 = Inactive state of serial clock is high
[5:4]	FRF	Frame format: 00 = Motorola SPI frame format 01 = TI synchronous serial frame format 10 = National Microwire frame format 11 = Reserved, undefined operation.
[3:0]	DSS	Data Size Select: 0000 ~ 0010 = Reserved. 0011 = 4-bit data. 0100 = 5-bit data. 0101 = 6-bit data. 0110 = 7-bit data. 0111 = 8-bit data. 1000 = 9-bit data. 1001 = 10-bit data. 1010 = 11-bit data. 1011 = 12-bit data. 1100 = 13-bit data. 1101 = 14-bit data. 1110 = 15-bit data. 1111 = 16-bit data.

3.18.5.2 Control Register 1 (SSPCR1)

Register	Offset	R/W	Description	Reset Value
SSPCR1 x=0, 1	SSPx_BA+0x04	R/W	Control Register 1	0x0000_0000

Bits	Description	
[15:6]	Reserved	Reserved
[5]	RX_LSB	control receive order of receive fifo data item, compatible with DSS of SSPCR0 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 DSS of SSPCR0.
[4]	TX_LSB	control transmit order of transmit fifo data item, compatible with DSS of SSPCR0 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 DSS of SSPCR0.
[3]	SOD	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 SOD 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]	MS	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.
[1]	SSE	Synchronous Serial Port Enable: 0 = SPI operation disabled. 1 = SPI operation enabled
[0]	LBM	Loop Back Mode: 0 = Normal serial port operation enabled. 1 = Output of transmit serial shifter is connected to input of receive serial shifter internally.

3.18.5.3 Data Register (SSPDR)

Register	Offset	R/W	Description	Reset Value
SSPDR x=0, 1	SSPx_BA+0x08	R/W	Receive FIFO(read) and Transmit FIFO Data Register	Configurable

Bits	Description	
[15:0]	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.

3.18.5.4 Status Register (SSPSR)

Register	Offset	R/W	Description	Reset Value
SSPSR x=0, 1	SSPx_BA+0x0C	R	Status Register	0x0000_0003

Bits	Description	
[15:5]	Reserved	Reserved
[4]	BSY	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]	RFF	Receive FIFO Full: 0 = Receive FIFO is not full. 1 = Receive FIFO is full.
[2]	RNE	Receive FIFO Not Empty: 0 = Receive FIFO is empty. 1 = Receive FIFO is not empty
[1]	TNF	Transmit FIFO Not Full 0 = Transmit FIFO is full. 1 = Transmit FIFO is not full.
[0]	TNE	Transmit FIFO Empty 0 = Transmit FIFO is not empty. 1 = Transmit FIFO is empty

3.18.5.5 Clock Prescale Register (SSPCPSR)

Register	Offset	R/W	Description	Reset Value
SSPCPSR x=0, 1	SSPx_BA+0x10	R/W	Clock Prescale Register	0x0000_0000

Bits	Description	
[15:8]	Reserved	Reserved
[7:0]	CPSDVSR	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.

3.18.5.6 Interrupt Mask Set and Clear Register (SSPIMSC)

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 corresponding mask.

Register	Offset	R/W	Description	Reset Value
SSPIMSC x=0, 1	SSPx_BA+0x14	R/W	Interrupt Mask Set and Clear Register	0x0000_0000

Bits	Description	
[15: 4]	Reserved	Reserved
[3]	TXIM	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]	RXIM	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]	RTIM	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]	RORIM	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.

3.18.5.7 Raw Interrupt Status Register (SSPRIS)

Register	Offset	R/W	Description	Reset Value
SSPRIS x=0, 1	SSPx_BA+0x18	R/W	Raw Interrupt Status Register	0x0000_0008

Bits	Description	
[15:4]	Reserved	Reserved
[3]	TXRIS	Gives the raw interrupt state, prior to masking, of the *SSPTXINTR* interrupt
[2]	RXRIS	Gives the raw interrupt state, prior to masking, of the *SSPRXINTR* interrupt
[1]	RTRIS	Gives the raw interrupt state, prior to masking, of the *SSPRTINTR* interrupt
[0]	RORRIS	Gives the raw interrupt state, prior to masking, of the *SSPRORINTR* interrupt

3.18.5.8 Masked Interrupt Status Register (SSPMIS)

Register	Offset	R/W	Description	Reset Value
SSPMIS x=0, 1	SSPx_BA+0x1C	R	Masked Interrupt Status Register	0x0000_0000

Bits	Description	
[15:4]	Reserved	Reserved
[3]	TXMIS	Gives the transmit FIFO masked interrupt state, after masking, of the *SSPTXINTR* interrupt
[2]	RXMIS	Gives the receive FIFO masked interrupt state, after masking, of the *SSPRXINTR* interrupt
[1]	RTMIS	Gives the receive timeout masked interrupt state, after masking, of the *SSPRTINTR* interrupt
[0]	RORMIS	Gives the receive over run masked interrupt status, after masking, of the *SSPRORINTR* interrupt

3.18.5.9 Interrupt Clear Register (SSPICR)

Register	Offset	R/W	Description	Reset Value
SSPICR x=0, 1	SSPx_BA+0x20	R	Interrupt Clear Register	0x0000_0000

Bits	Description	
[15:2]	Reserved	Reserved
[1]	RTIC	Clears the *SSPRTINTR* interrupt
[0]	RORIC	Clears the *SSPRORINTR* interrupt

3.18.5.10 DMA Control Register (SSPDMACR)

Register	Offset	R/W	Description	Reset Value
SSPDMACR x=0, 1	SSPx_BA+0x24	R	DMA Control Register	0x0000_0000

Bits	Description	
[15:2]	Reserved	Reserved
[1]	TXDMAE	Transmit DMA Enable. 0 = DMA for the transmit FIFO is disabled. 1 = DMA for the transmit FIFO is enabled.
[0]	RXDMAE	Receive DMA Enable. 0 = DMA for the receive FIFO is disabled. 1 = DMA for the receive FIFO is enabled.

3.19 UART Controller (UART)

3.19.1 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

3.19.2 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
- Loopback 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

3.19.3 Block Diagram

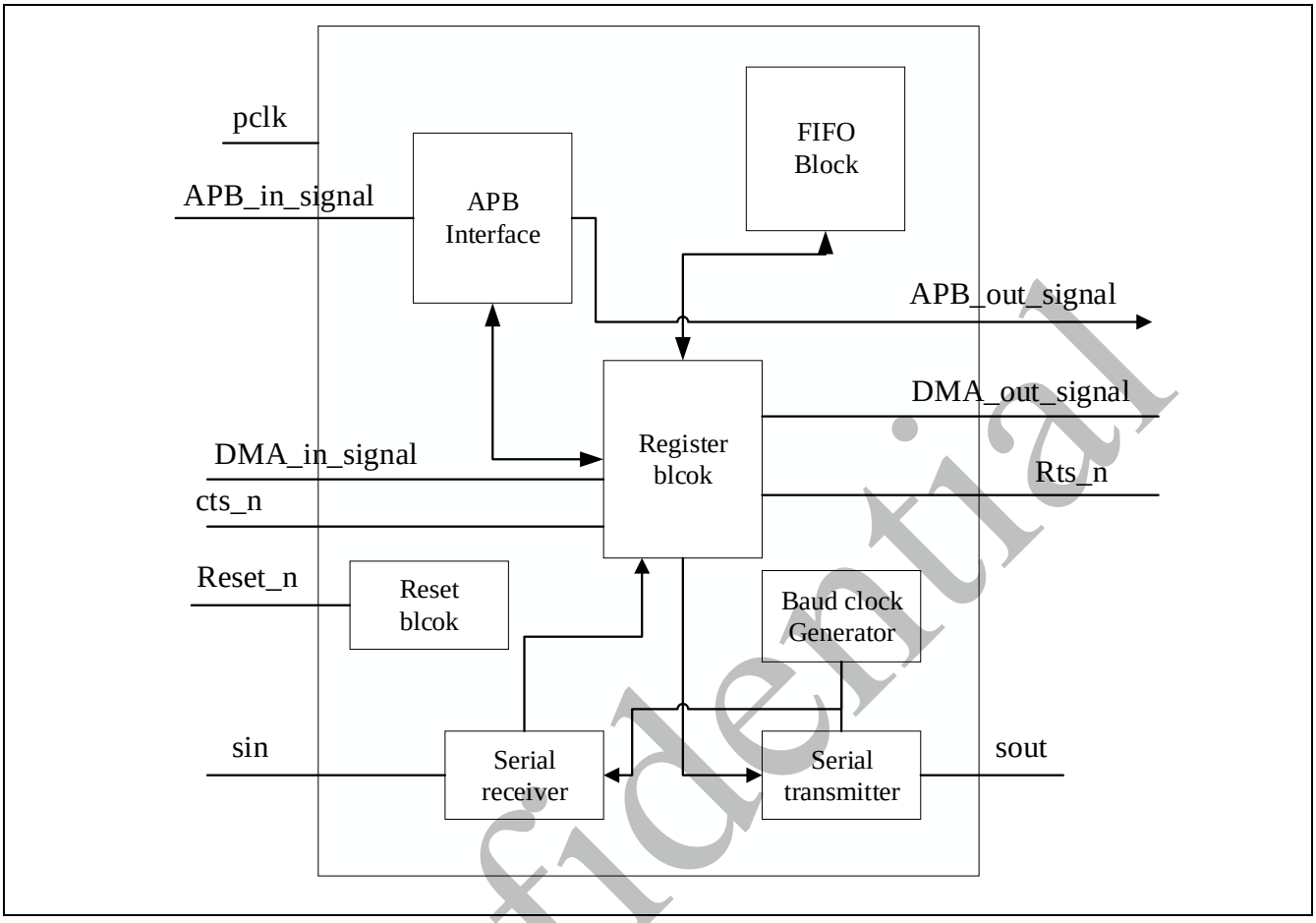


Figure 3-118 UART Controller Block Diagram

3.19.4 Functional Description

3.19.4.1 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 3-119.

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.

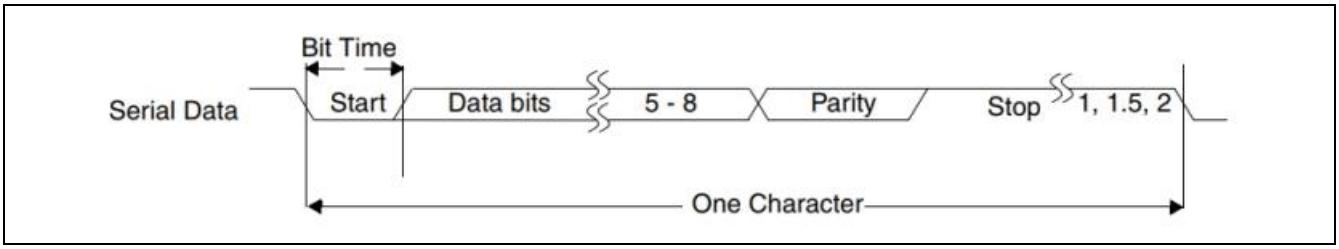


Figure 3-119 Serial Data Format

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 3-120 shows the sampling points of the first two bits in a serial character.

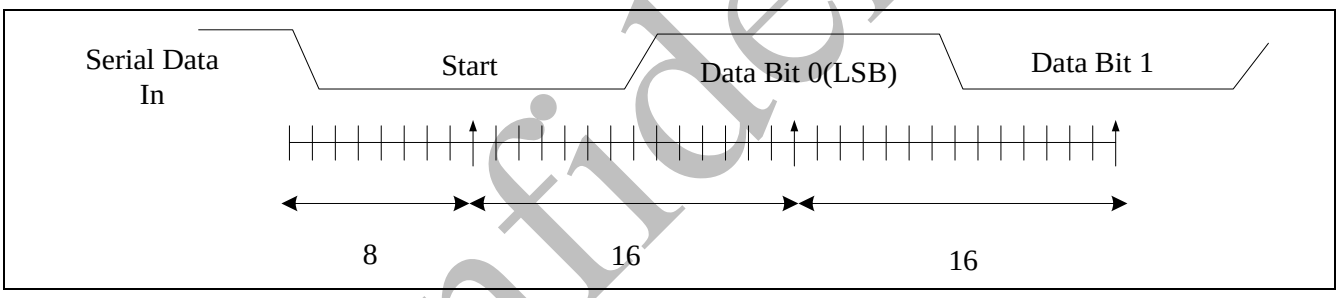


Figure 3-120 Receiver Serial Data Sample Points

3.19.4.2 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.

3.19.4.3 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:

$$Error\% = 0.000799 \times 100 = 0.0799$$

3.19.4.4 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.

3.19.4.5 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 THRE interrupt mode)
- Busy Detect Indication

These interrupt types are covered in more detail in Table 3-24.

Table 3-24 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

3.19.4.6 Auto Flow Control

Figure 3-121 shows a block diagram of the Auto Flow Control functionality. When Auto RTS and Auto CTS 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 RTS 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 CTS 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 THRE 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.

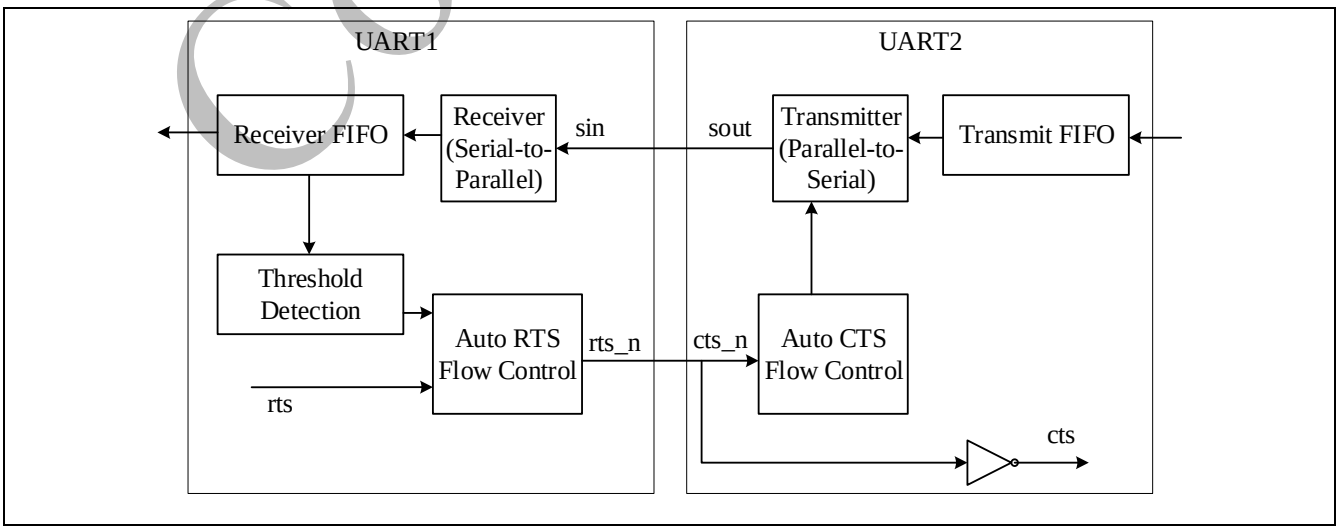


Figure 3-121 Auto Flow Control Block Diagram

Auto RTS and Auto CTS are described as follows:

- Auto RTS – Becomes active when the following occurs:
  - RTS (MCR[1] bit and MCR[5]bit are both set)
  - FIFOs are enabled (FCR[0]) bit is set)
- Auto CTS – becomes active when the following occurs:
  - AFCE (MCR[5] bit = 1)
  - FIFOs are enabled through FIFO Control Register FCR[0] bit

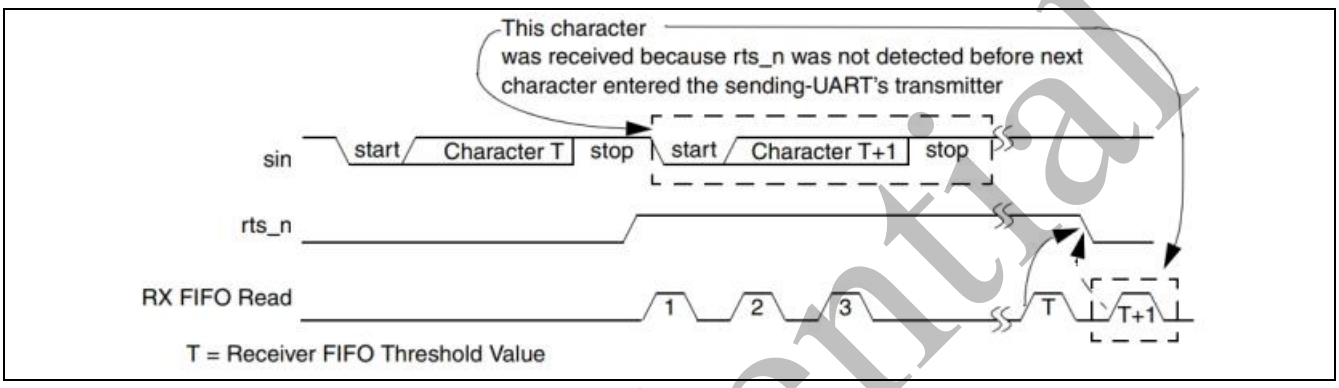


Figure 3-122 Auto RTS Timing

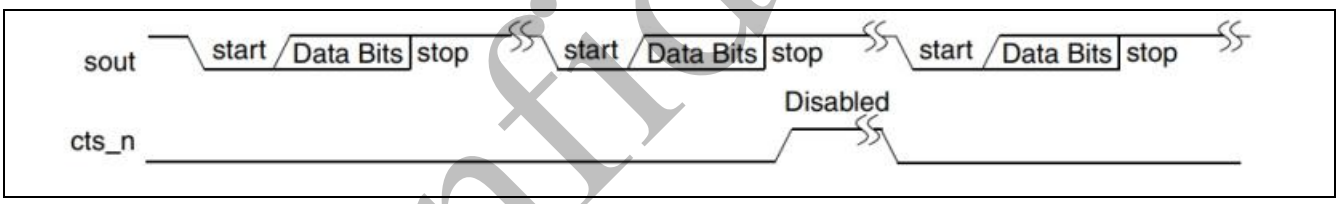


Figure 3-123 Auto CTS Timing

3.19.4.7 Programmable THRE Interrupt

When Programmable THRE Interrupt mode is selected, it can be enabled using the Interrupt Enable Register (IER[7]).When FIFOs and THRE mode are enabled, the THRE 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 3-124.

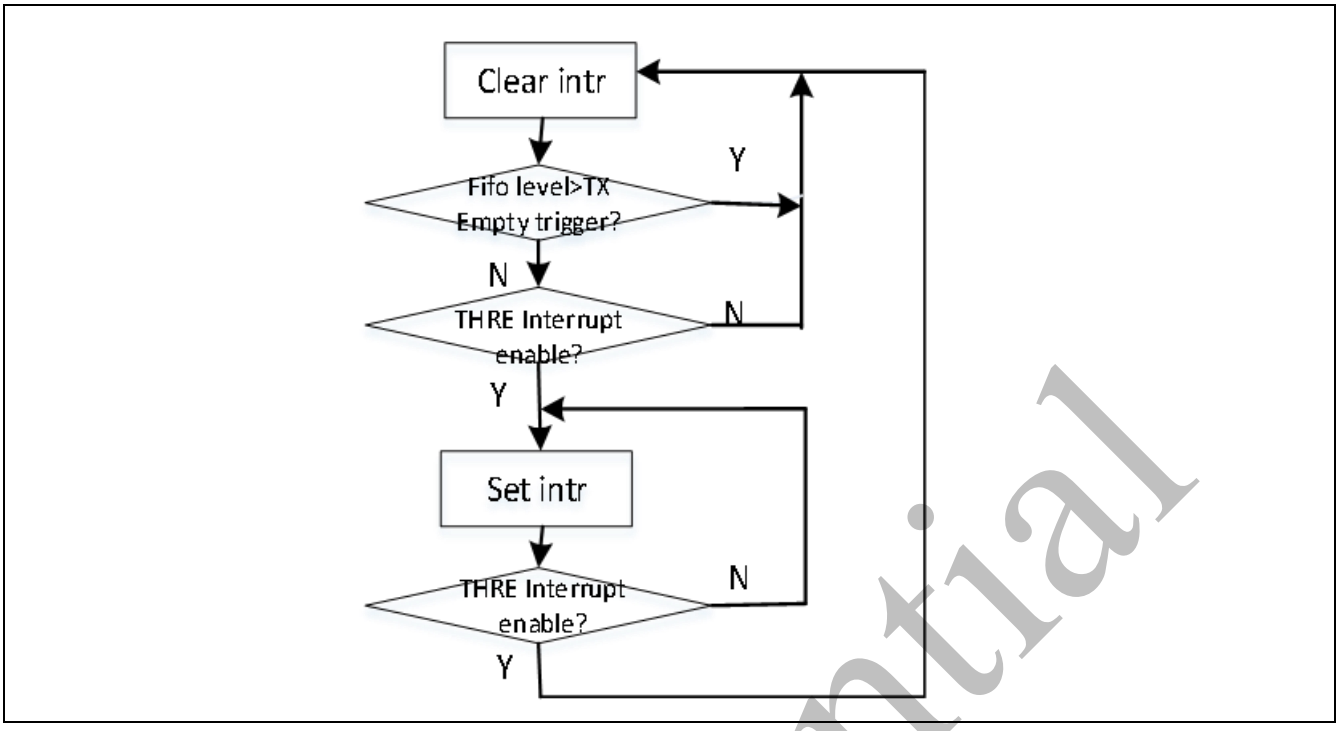


Figure 3-124 Flowchart of Interrupt Generation for Programmable THRE Interrupt Mode

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.

3.19.4.8 DMA Support

DMA Interface Signal

The UART supports DMA signaling with use of the signal described in Table 3-25.

Table 3-25 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 3-125 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.

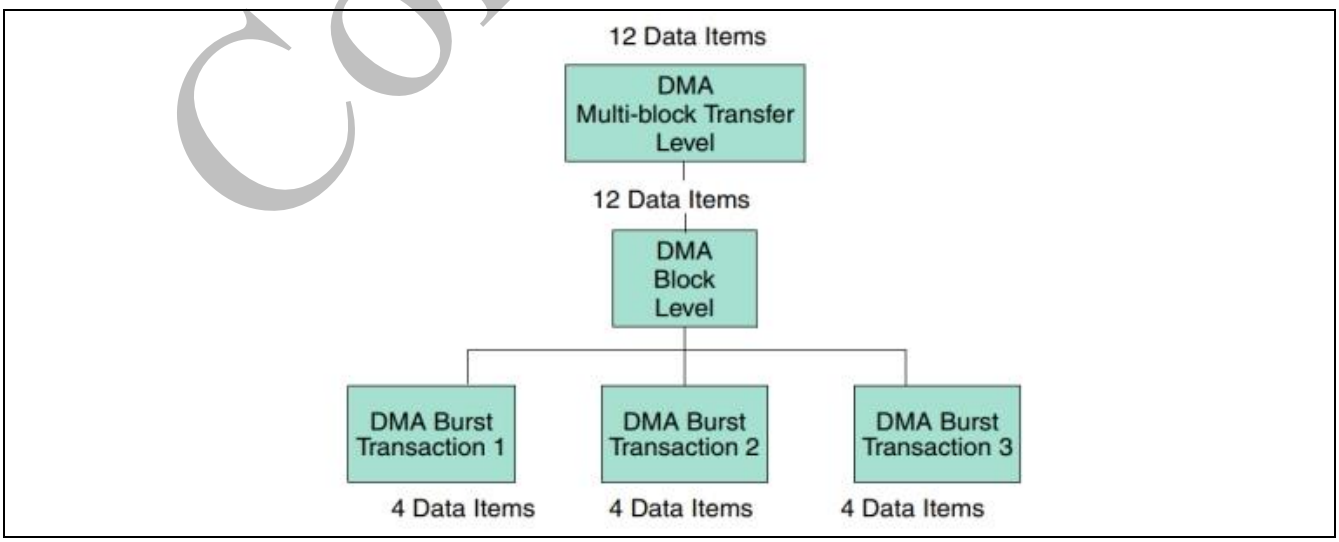


Figure 3-125 Breakdown of DMA Transfer into Burst Transactions

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 3-126, a series of burst transactions followed by single transactions are needed to complete the block transfer.

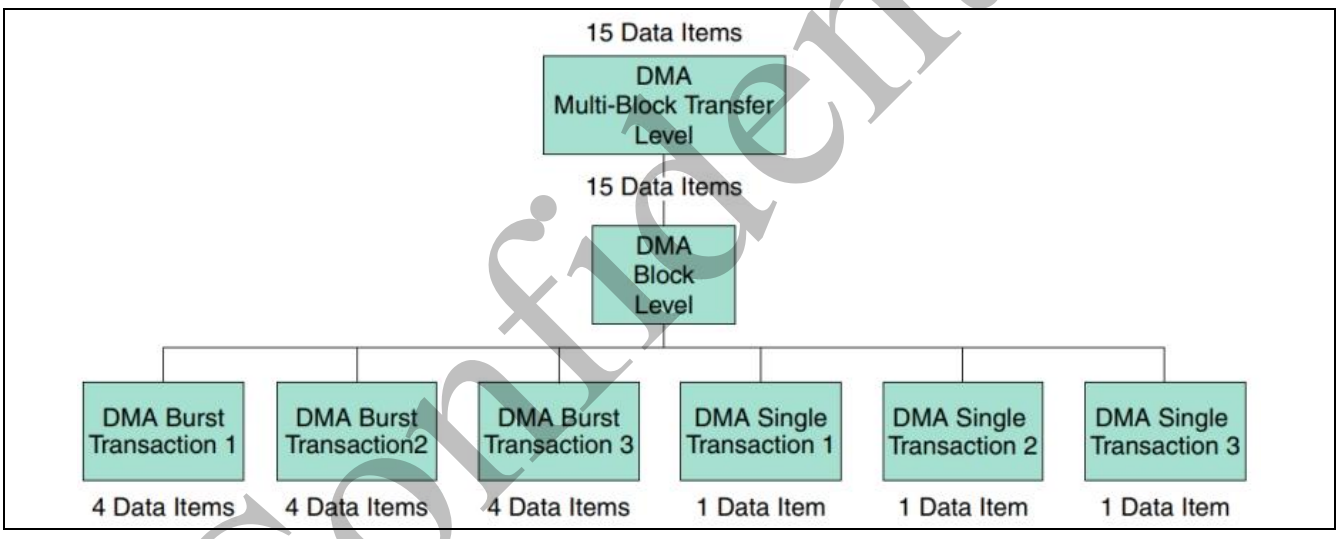


Figure 3-126 Breakdown of DMA Transfer into Single and Burst Transactions

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 (TET) 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

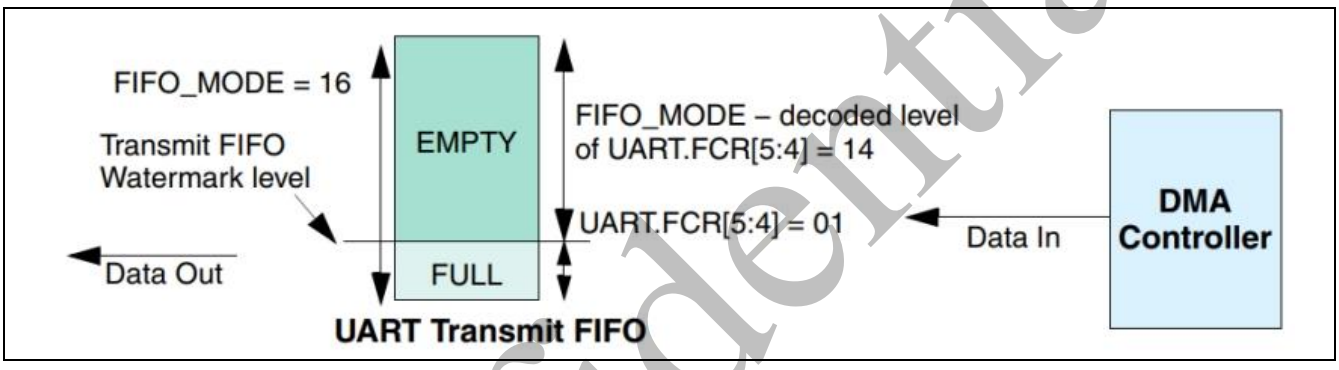


Figure 3-127 Case 1 Watermark Levels

- 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)

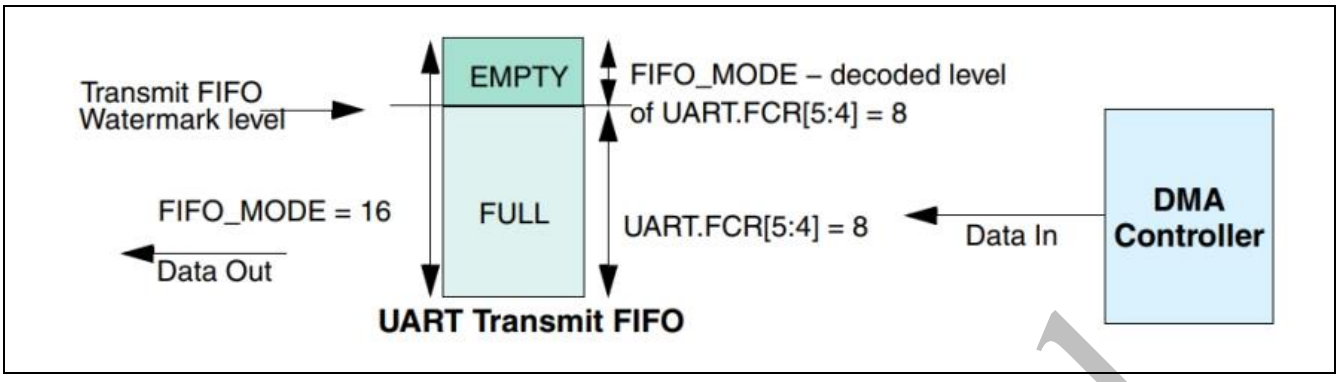


Figure 3-128 Case 2 Watermark Levels

- 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 1-11, 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 (RT) 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 3-129, 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.

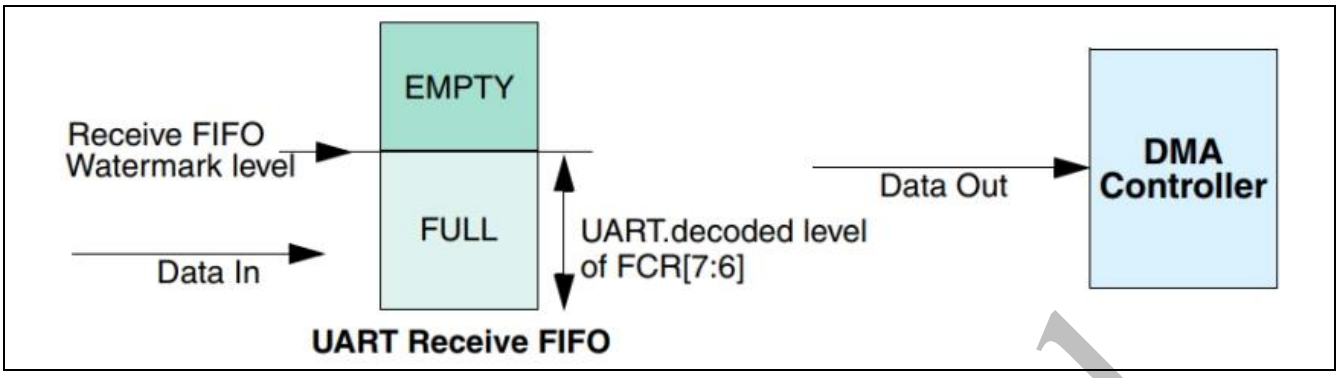


Figure 3-129 UART Receive FIFO

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 underflowed, 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.

3.19.5 Programing Example

3.19.5.1 Flowchart for UART Transmit Programming Example

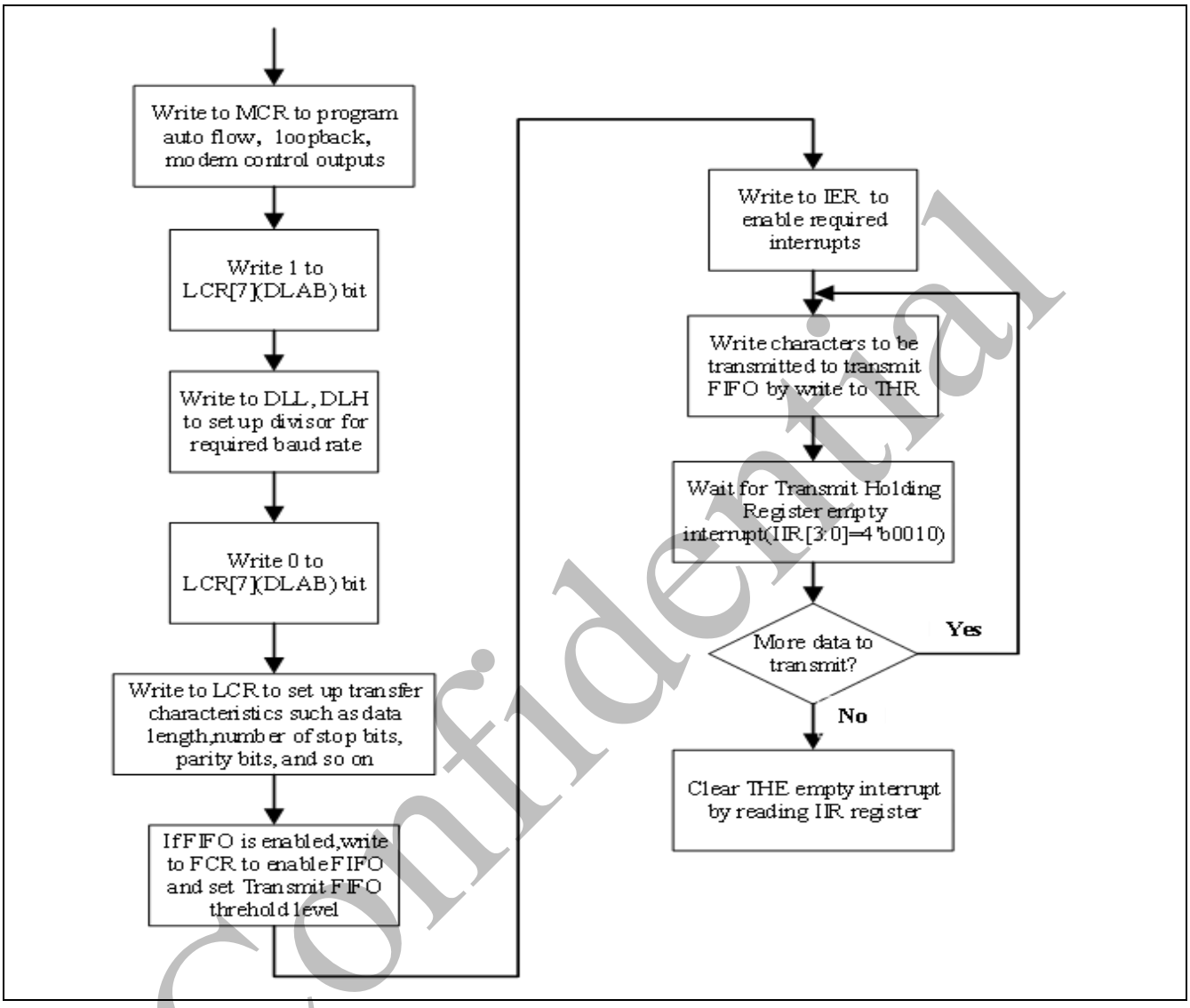


Figure 3-130 Flowchart for UART Transmit Programming Example

3.19.5.2 Flowchart for UART Receive Programming Example

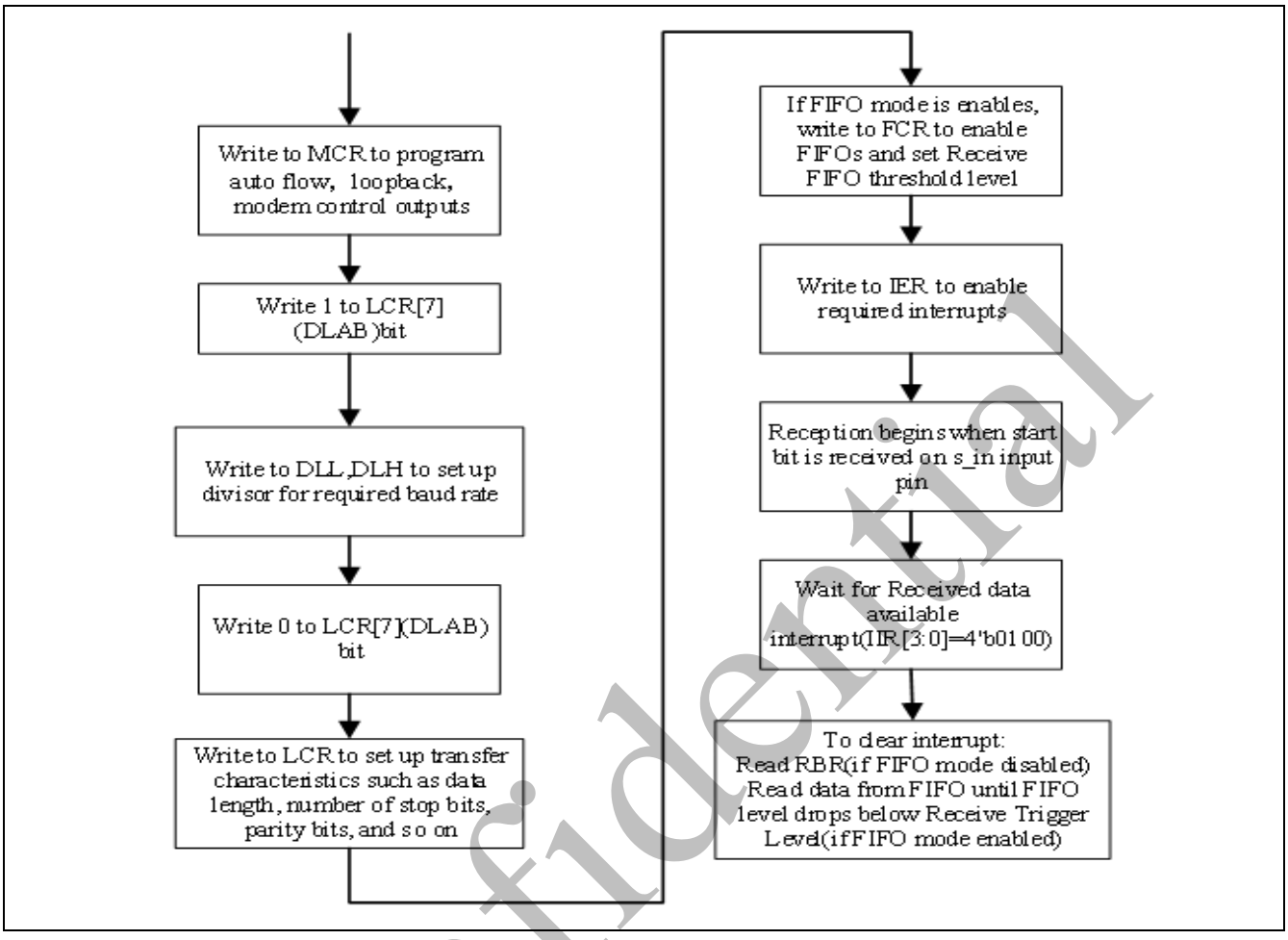


Figure 3-131 Flowchart for UART Receive Programming Example

3.19.6 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

3.19.7 UART Register Description

3.19.7.1 Receive Buffer Register (RBR)

This register can be accessed only when the DLAB 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 DR 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.

3.19.7.2 Transmit Holding Register (THR)

This register can be accessed only when the DLAB 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 9th 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 THRE bit (LSR[5]) is set. If FIFOs are enabled (FCR[0] = 1) and THRE 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.

3.19.7.3 Divisor Latch Low (DLL)

This register can be accessed only when the DLAB 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.

3.19.7.4 Divisor Latch High (DLH)

This register can be accessed only when the DLAB 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.

3.19.7.5 Interrupt Enable Register (IER)

This register can be accessed only when the DLAB 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]	PTIME	Programmable THRE Interrupt Mode Enable. This is used to enable/disable the generation of THRE Interrupt. 0 – disabled 1 – enabled
[6:4]	Reserved	Reserved
[3]	EDSSI	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]	ELSI	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]	ETBEI	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]	ERBFI	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

3.19.7.6 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]	FIFOSE	FIFOs Enabled. This is used to indicate whether the FIFOs are enabled or disabled. 00 – disabled 11 – enabled
[5:4]	Reserved and read as 0	Reserved
[3:0]	IID	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 3-26.

Table 3-26 Interrupt Control Functions

IID	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 (nonFIFO 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. THRE Mode disabled) or XMIT FIFO at or	Reading the IIR register (if source of interrupt); or, writing into THR (FIFOs or THRE Mode not selected

			below threshold (Prog. THRE Mode enabled)	or disabled) or XMIT FIFO above threshold (FIFOs and THRE 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 CTS (that is, DCTS 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

3.19.7.7 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]	RT	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]	TET	TX Empty Trigger. This is used to select the empty threshold level at which the THRE 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]	XFIFOR	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]	RFIFOR	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]	FIFOE	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.

3.19.7.8 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]	DLAB	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]	BC	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 Loopback Mode, as determined by MCR[4], the sout line is forced low until the Break bit is cleared. When in Loopback Mode, the break condition is internally looped back to the receiver and the sir_out_n line is forced low.
[5]	SP	Stick Parity. This bit is used to force parity value. When PEN, EPS, and Stick Parity are set to 1,the parity bit is transmitted and checked as logic 0. If PEN and Stick Parity are set to 1 and EPS 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]	EPS	Even Parity Select. This is used to select between even and odd parity to be transmitted or checked, when parity is enabled (PEN set to 1). 0 – odd parity 1 – even parity
[3]	PEN	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]	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 DLS (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]	DLS	Data Length Select. When *DLS_E* 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

3.19.7.9 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]	SIRE	SIR Mode Enable. (Write Protected) 0 – IrDA SIR Mode disabled 1 – IrDA SIR Mode enabled
[5]	AFCE	Auto Flow Control Enable. 0 – Auto Flow Control Mode disabled 1 – Auto Flow Control Mode enabled
[4]	LB	LoopBack 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 loopback 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]	OUT2	OUT2. 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 Loopback 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]	OUT1	OUT1. 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 Loopback 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]	RTS	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 RTS Flow Control is not enabled (MCR[5] set to 0), the rts_n signal is set low by programming MCR[1] (RTS) 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 Loopback 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]	DTR	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 Loopback 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.

3.19.7.10 Line Status Register (LSR)

Register	Offset	R/W	Description	Reset Value
MCR x = 0, 1	UARTx_BA+0x14	R	Line Status Register	0x0000_0060

Bits	Description	
[31:9]	Reserved	Reserved
[8]	ADDR_RCVD	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]	RFE	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]	TEMT	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]	THRE	Transmit Holding Register Empty bit. If THRE 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 THRE Interrupt to occur, if the THRE Interrupt is enabled. If THRE_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 THRE interrupts, which are then controlled by the FCR[5:4] threshold setting.
[4]	BI	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 BI 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]	FE	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 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 (FE) bit (LSR[3]) is set if a break interrupt has occurred, as indicated by Break Interrupt (BI) 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 FE bit.
[2]	PE	Parity Error bit. This is used to indicate the occurrence of a parity error in the receiver if the Parity Enable (PEN) 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 (PE) bit (LSR[2]) can be set if a break interrupt has occurred, as indicated by Break Interrupt (BI) 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 PE bit.
[1]	OE	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 OE 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 OE bit.
[0]	DR	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.

3.19.7.11 Modem Status Register (MSR)

Register	Offset	R/W	Description	Reset Value
MSR x = 0, 1	UARTx_BA+0x18	R	ModemStatus Register	0x0000_0000

Bits	Description	
[31:8]	Reserved	Reserved
[7]	DCD	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 Loopback Mode (MCR[4] set to 1), DCD is the same as MCR[3] (Out2).
[6]	RI	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 Loopback Mode (MCR[4] set to 1), RI is the same as MCR[2] (Out1).
[5]	DSR	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), DSR is the same as MCR[0] (DTR).
[4]	CTS	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 Loopback Mode (MCR[4] = 1), CTS is the same as MCR[1] (RTS).
[3]	DDCD	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 DDCD bit. In Loopback Mode (MCR[4] = 1), DDCD reflects changes on MCR[3] (Out2). **Note**: If the DDCD bit is not set and the dcd_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDCD 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 Loopback Mode (MCR[4] = 1), TERI reflects when MCR[2] (Out1) has changed state from a high to a low.
[1]	DDSR	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 DDSR bit. In Loopback Mode (MCR[4] = 1), DDSR reflects changes on MCR[0] (DTR). **Note**: If the DDSR bit is not set and the dsr_n signal is asserted (low) and a reset occurs (software or otherwise), then the DDSR bit is set when the reset is removed if the dsr_n signal remains asserted.
[0]	DCTS	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 DCTS bit. In Loopback Mode (MCR[4] = 1), DCTS reflects changes on MCR[1] (RTS). **Note**: If the DCTS bit is not set and the cts_n signal is asserted (low) and a reset occurs (software or otherwise), then the DCTS bit is set when the reset is removed if the cts_n signal remains asserted.

3.19.7.12 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]	Scratchpad Register	This register is for programmers to use as a temporary storage space. It has no defined purpose in the UART.

3.19.7.13 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]	RFF	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]	RFNE	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]	TFE	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]	TFNF	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

3.19.7.14 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.

3.19.7.15 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.

3.19.7.16 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.

3.19.7.17 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.

3.19.7.18 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

3.19.7.19 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 'ADDR_MATCH' (LCR_EXT[1]) and 'DLS_E' (LCR_EXT[0]) bits are set to 1.

3.19.7.20 Transmit Address Register (TAR)

Register	Offset	R/W	Description	Reset Value
TAT x=0,1	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 *DLS_E* (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' (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' bit will be auto-cleared by the hardware.

3.19.7.21 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]	TRANSMIT_MODE	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 bit is used as a control knob to indicate the UART on when to send the address.
[2]	SEND_ADDR	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 TxFIFO 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 DLS_E bit is set to 1 and TRANSMIT_MODE is set to 0.
[1]	ADDR_MATCH	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 RxFIFO. User is responsible to read the data and differentiate b/n address and data. **Note**: This field is applicable only when DLS_E is set to 1.
[0]	DLS_E	Extension for DLS. 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 DLS

3.20 Timer Controller (TMR)

3.20.1 Overview

The PN1080 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.

3.20.2 Features

- Support 3 timer, each timer features are described as below
- 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

3.20.3 Block Diagram

The timer controller block diagram and clock control are shown in Figure 3-132 and Figure 3-133.

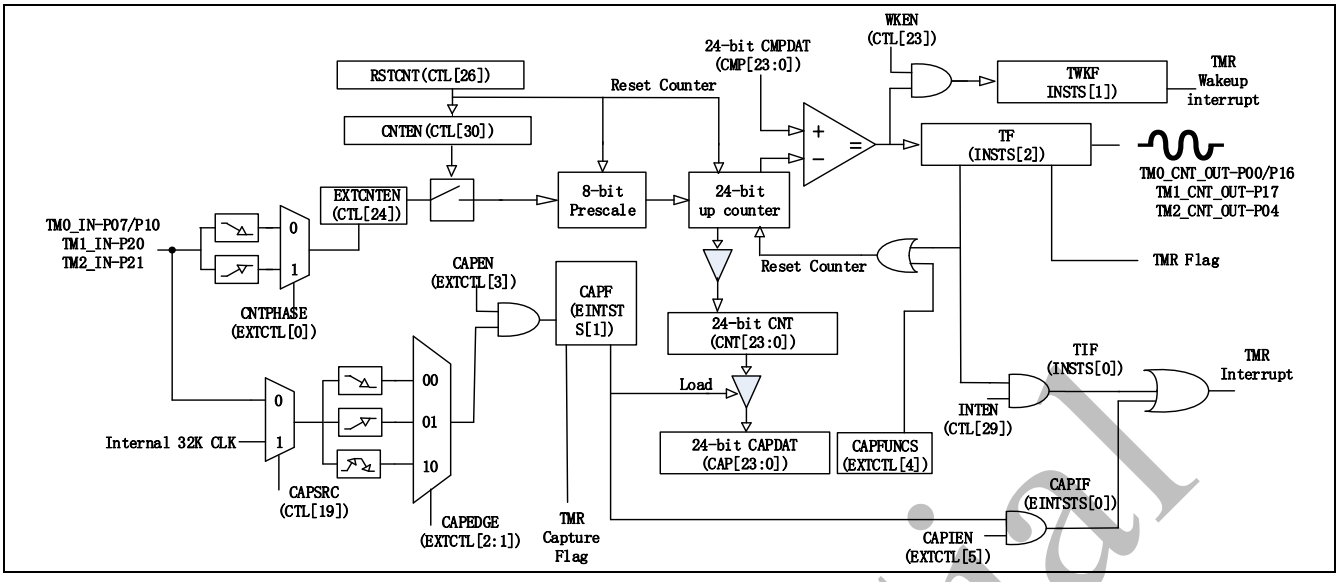


Figure 3-132 Timer Controller Block Diagram

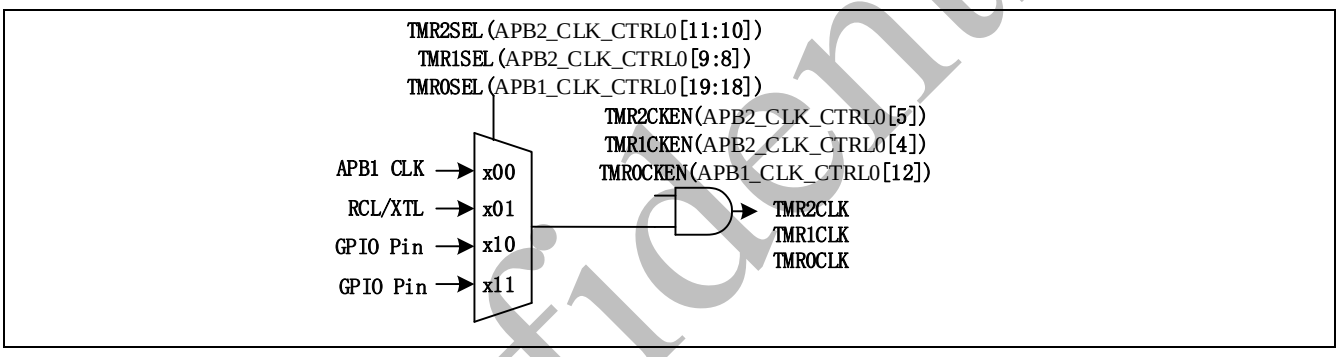


Figure 3-133 Clock Source of Timer Controller

3.20.4 Basic Configuration

3.20.4.1 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 3-133.

When choose apb subsystem clock as TMRnum 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 0~2) 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.

3.20.4.2 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.

3.20.4.3 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.

3.20.5 Functional Description

3.20.5.1 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.

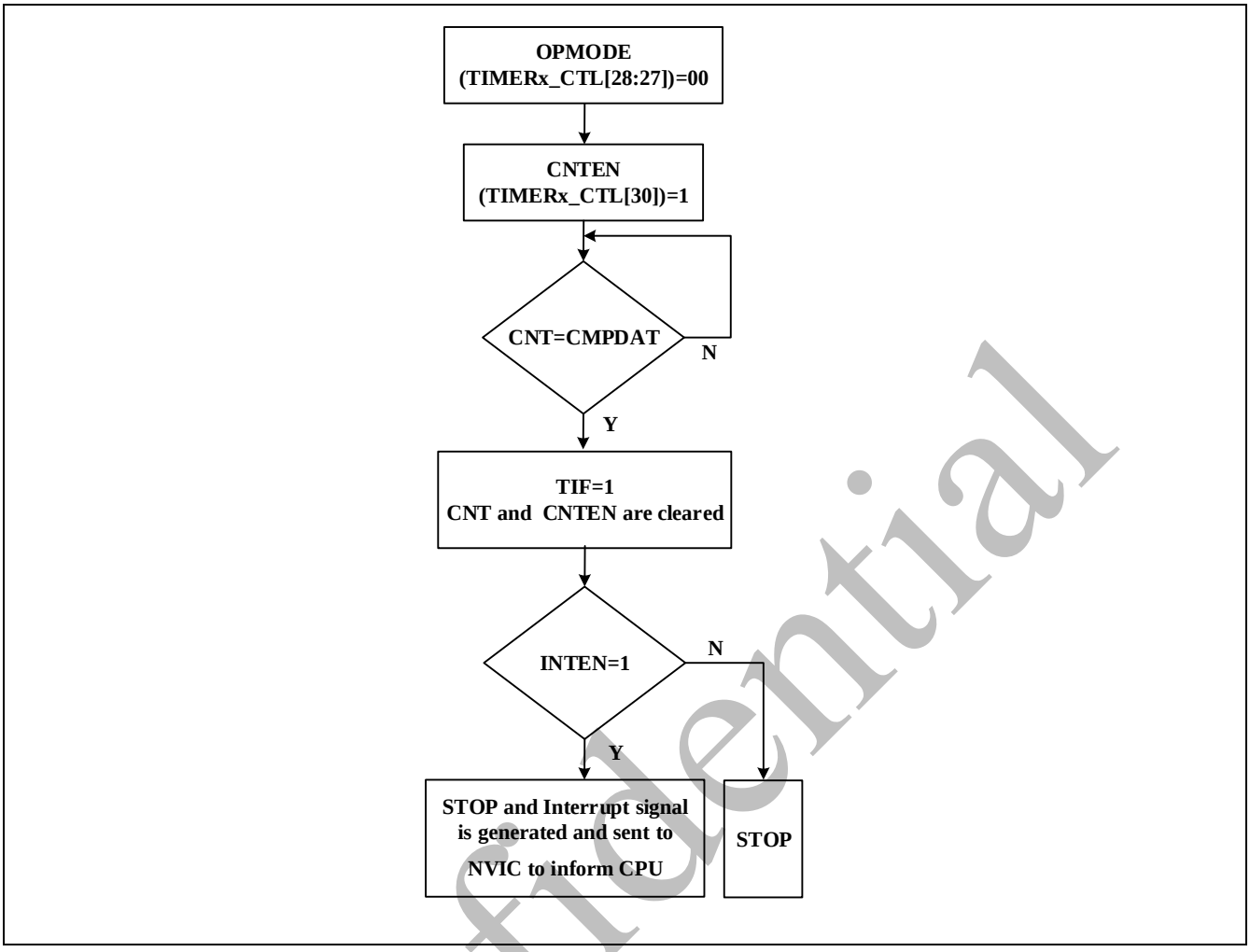


Figure 3-134 One-shot Mode

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.

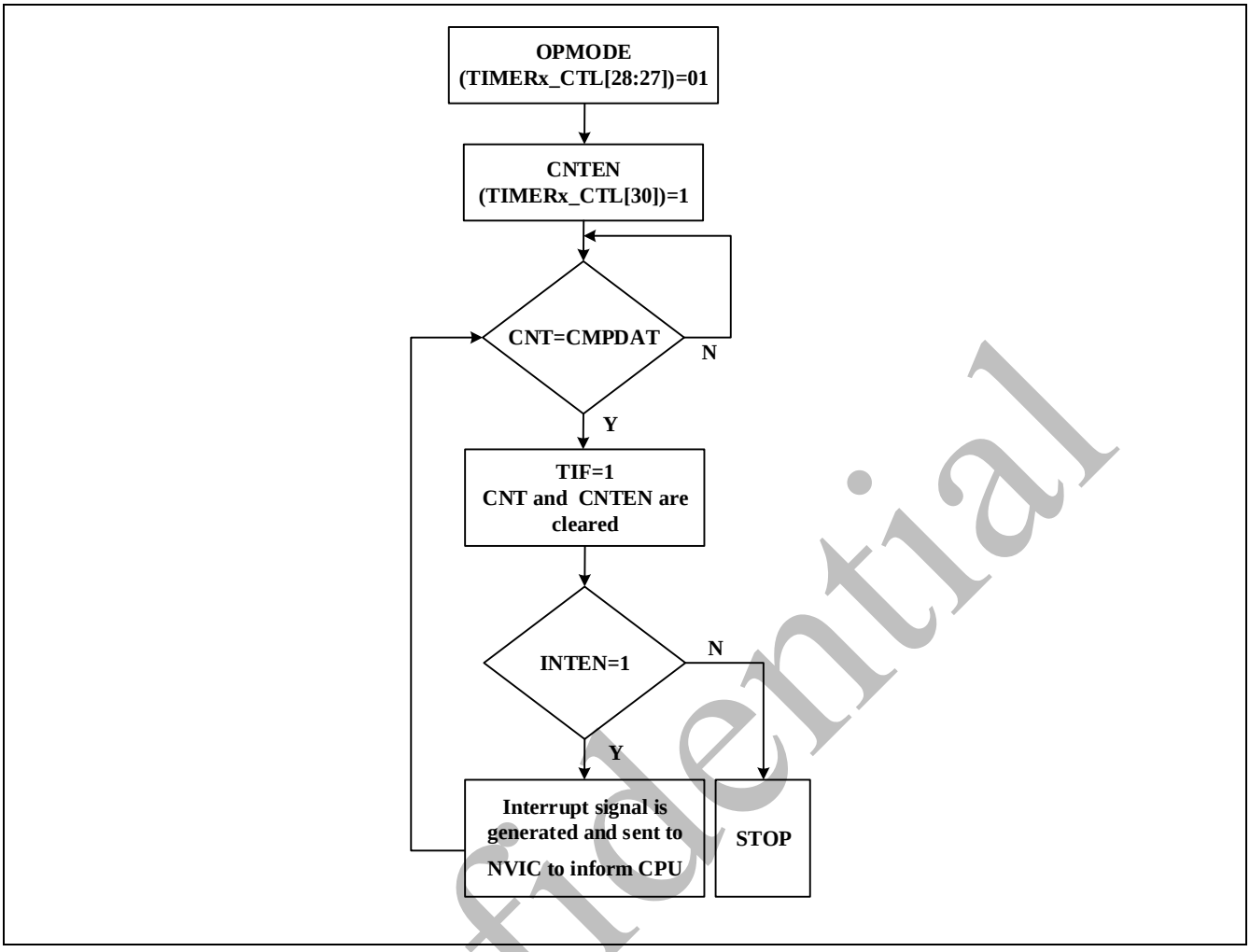


Figure 3-135 Periodic Mode

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 224-1. 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 224-1, 0, 1, 2, 3, …to 224-1 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.

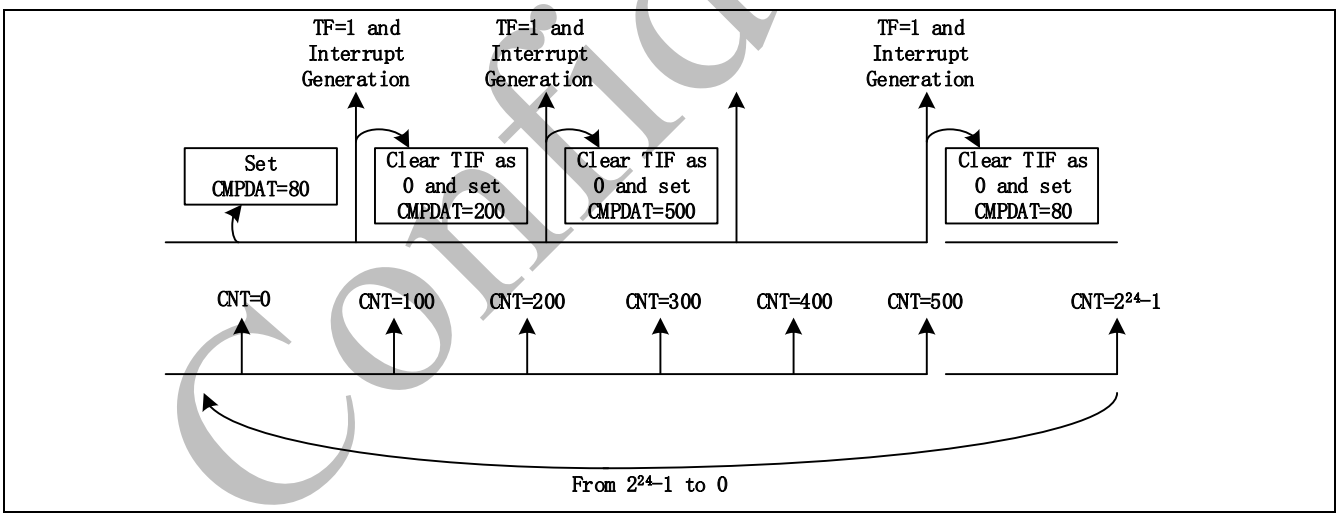


Figure 3-136 Continuous Counting Mode

3.20.5.2 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.

3.20.5.3 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= 0~2) 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 or 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 3-137.

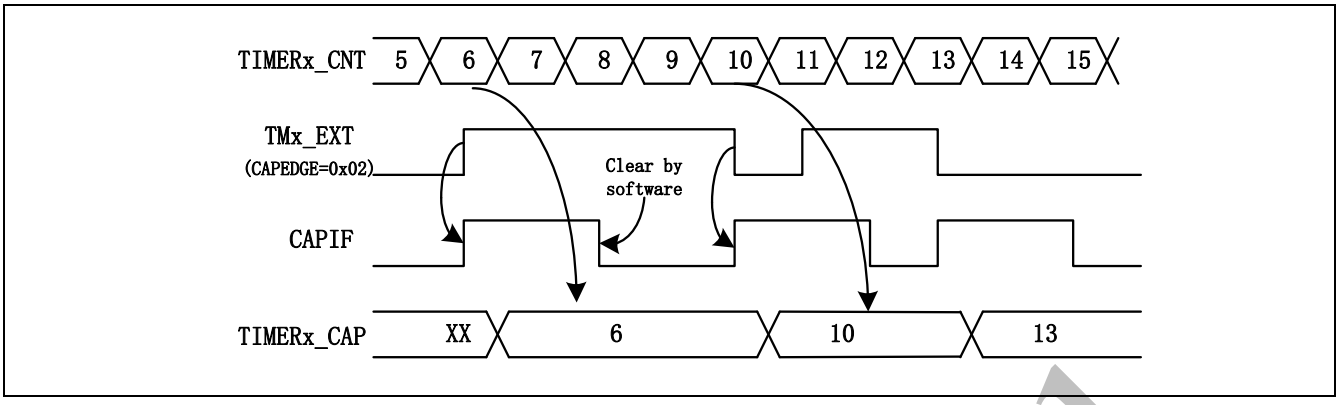


Figure 3-137 Free-Counting Capture Mode

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= 0~2). 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 3-138. The difference between Free-Counting Capture Mode and External Reset Counter Mode is whether the TMR_CNT value will be cleared or not.

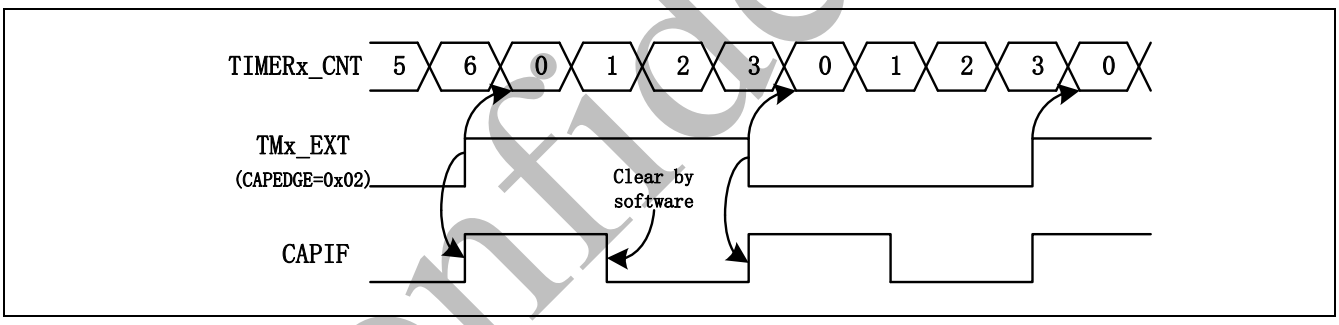


Figure 3-138 External Reset Counter Mode

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= 0~2) 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 3-27.

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 3-27 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= 0~2) pin is detected. CNT is captured to CAPDAT.
	0	0	01	A 0 to 1 transition on TMRx_EXT (x= 0~2) pin is detected. CNT is captured to CAPDAT.
	0	0	10	Either 1 to 0 or 0 to 1 transition on TMRx_EXT (x= 0~2) 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= 0~2) pin is detected. CNT is reset to 0.
	0	1	01	A 0 to 1 transition on TMRx_EXT (x= 0~2) pin is detected. CNT is reset to 0.
	0	1	10	Either 1 to 0 or 0 to 1 transition on TMRx_EXT (x= 0~2) 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= 0~2) 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= 0~2) 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= 0~2) 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= 0~2) pin is detected to reset CNT as 0 and then starts counting, while 1 to 0 transition stops counting.

3.20.6 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: TMR0_BA = 0x4000_8000 TMR1_BA = 0x4001_4000 TMR2_BA = 0x4001_5000				
TIMER0_CTL	TMR0_BA+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TIMER0_CMP	TMR0_BA+0x04	R/W	Timer0 Compare Register	0x0000_0000
TIMER0_INTSTS	TMR0_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER0_CNT	TMR0_BA+0x0C	R	Timer0 Data Register	0x0000_0000
TIMER0_CAP	TMR0_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER0_EXTCTL	TMR0_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER0_EINTSTS	TMR0_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
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

3.20.7 TMR Register Description

3.20.7.1 Timer Control Register (CTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_CTL	TMR0_BA+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
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= 0~2) 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.

3.20.7.2 Timer Compare Register (CMP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CMP	TMR0_BA+0x04	R/W	Timer0 Compare Register	0x0000_0000
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.

3.20.7.3 Timer Interrupt Status Register (INTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_INTSTS	TMR0_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
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.

3.20.7.4 Timer Data Register (TIMERx_CNT)

Register	Offset	R/W	Description	Reset Value
TIMER0_CNT	TMR0_BA+0x0C	R	Timer0 Data Register	0x0000_0000
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.

3.20.7.5 Timer Capture Data Register (TIMERx_CAP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CAP	TMR0_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
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* (TIMERx_EINTSTS [0]) will set to 1 and the current timer counter value *CNT* (TIMERx_CNT [23:0]) will be auto-loaded into this CAPDAT field.

3.20.7.6 Timer External Control Register (EXTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_EXTCTL	TMR0_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
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= 0~2) pin is used to save the 24-bit timer counter value to CAPDAT register. **Note2**: When CAPFUNCS is 1, transition on TMx_EXT (x= 0~2) 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= 0~2) 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

3.20.7.7 Timer External Interrupt Status Register (TIMERx_EINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_EINTSTS	TMR0_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
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. **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.

3.20.8 Operation Steps

3.20.8.1 One-shot Mode

1. RCC module enable APB1_CLK_EN and APB2_CLK_EN.
2. RCC module enable TMR0CKEN, TMR1CKEN and TMR2CKEN.
3. OPMODE(CTL[28:27]) set to 2'b00.
4. CMPDAT(CMP[23:0]) set expected value.
5. Enable CNTEN(CTL[30]).
6. Read TF flag in while loop or read TIF flag in ISR function if INTEN is effective. Get CNT value.

3.21 CLKTRIM

3.21.1 Overview

The PAN108 series chip has an analog RC oscillator clock generation circuit, including a low-speed RC32K clock. Because the BLE protocol has strict requirements on the sleep time, the sleep time is mainly controlled by the RC32K clock, and 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 entering and exiting the low-power mode. Based on simple design considerations, a circuit is needed that uses an external accurate crystal (or DPLL with crystal as a reference source) to accurately measure the true frequency of the RC32K analog clock at a certain moment.

3.21.2 Features

3.21.2.1 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.

3.21.2.2 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.
- Four interrupt flag: coarse tuning completed flag, fine tuning completed flag, precision tuning completed flag, and counter overflow interruption.
- Polling mode and interrupt mode.

3.21.3 Block Diagram

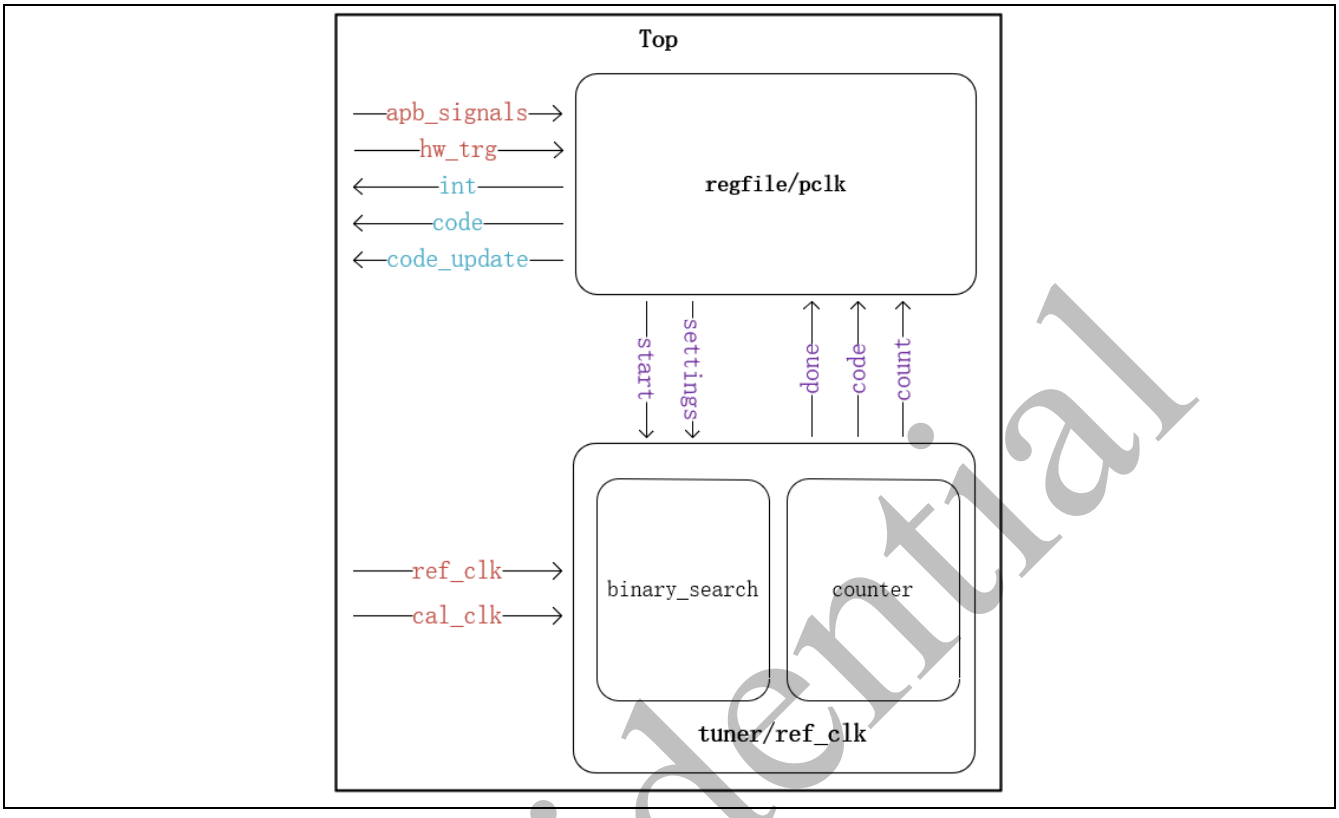


Figure 3-139 CLKTRIM Block Diagram

3.21.4 Functional Description

3.21.4.1 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.

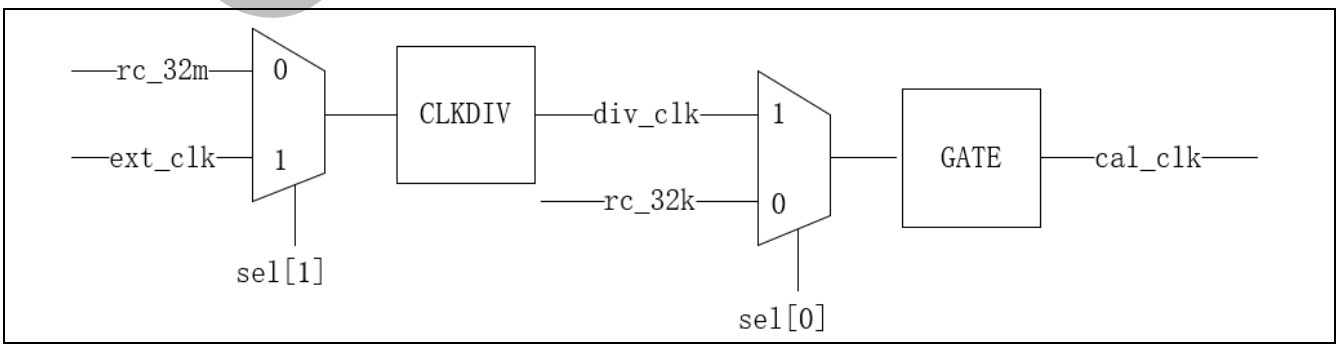


Figure 3-140 Clock Source

3.21.4.2 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:

$$M / Fcal = N / Fref$$

$$Fcal = Fref * M / N$$

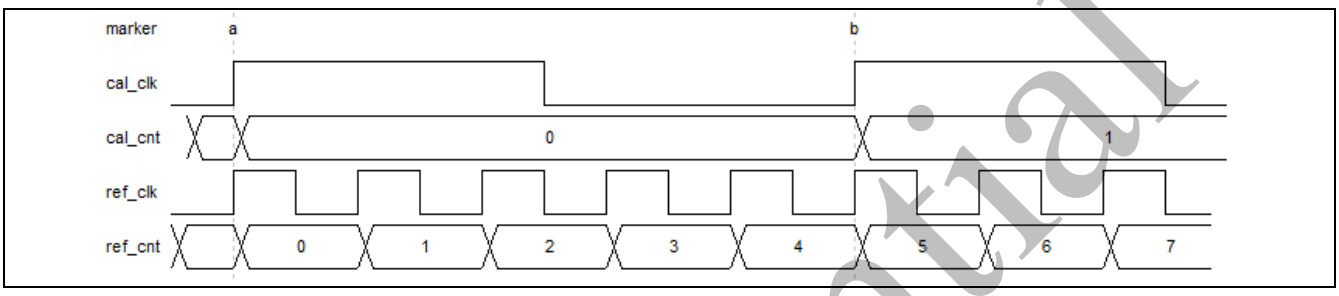


Figure 3-141 Measure Principle

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.

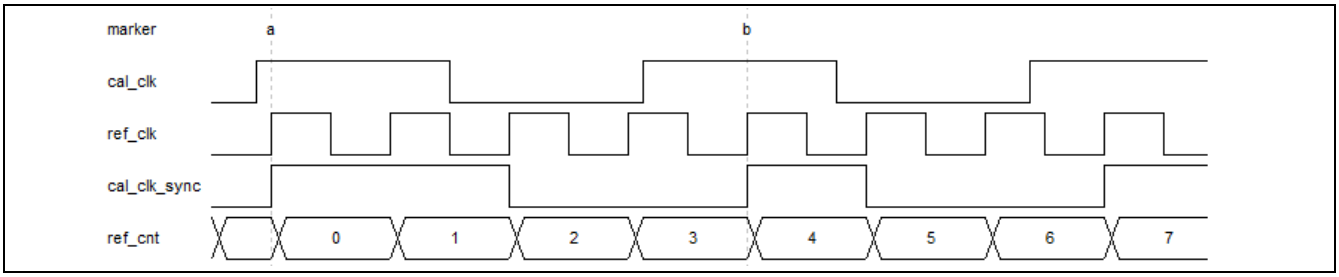


Figure 3-142 Measure Principle

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.

3.21.4.3 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.

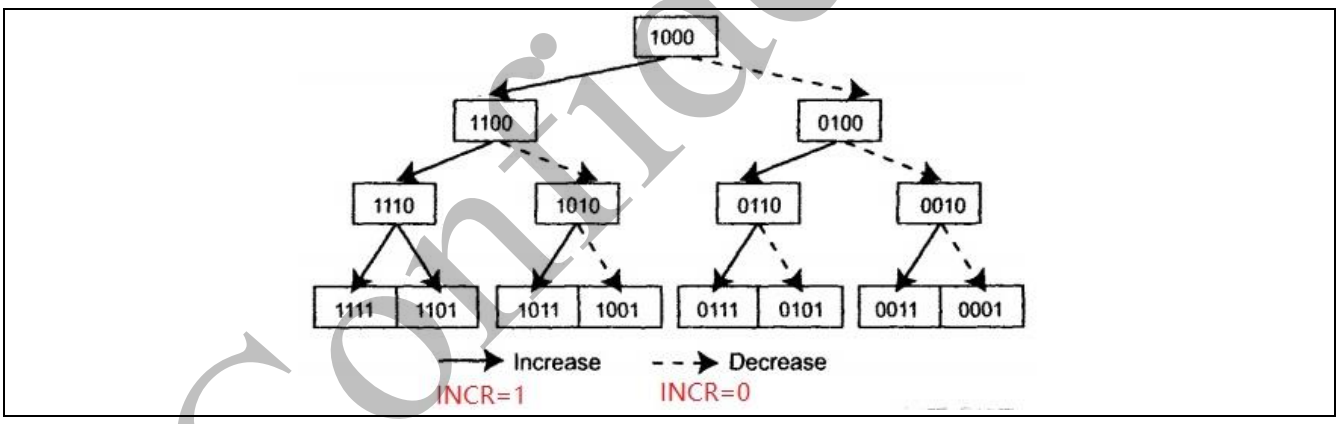


Figure 3-143 Calibration Principle

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.

3.21.5 CLKTRIM Register Map

Register	Offset	R/W	Description	Reset Value
CLKTRIM_BASE addr：0x4001_6000-0x4001_6FFF				
ClktrimEnReg	0x00	R/W	Enable register	0x0000_0000
ClktrimCodeReg	0x04	R/W	Tuning code register	0x0520_2004
ClktrimCtlReg	0x08	R/W	Control register	0x0032_0000
ClktrimIntReg	0x0C	R/W	Interrupt flag register	0x0000_0000
ClktrimCalCntReg	0x10	R/W	Measurement period register	0x0400_0064
ClktrimIdeaCntReg	0x14	R/W	Ideal count register	0x0001_869F
ClktrimRefCntReg	0x18	R	Reference count register	0x0000_0000

3.21.6 CLKTRIM Register Description

3.21.6.1 ClktrimEnReg

Register	Offset	R/W	Description	Reset Value
ClktrimEnReg	CLKTRIM_BASE +0x00	R/W	Enable register	0x0000_0000

Bits	Description	
[31:4]	Reserved	Reserved
[3]	PRECISION_EN	1:Start precision tuning 0:This bit will be cleared automatically after precision tuning finished or REFCNT overflow happens Reset_value:{1b0}
[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}
[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}
[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}

3.21.6.2 ClktrimCodeReg

Register	offset	R/W	Description	Reset Value
ClktrimCodeReg	CLKTRIM_BASE+0x04	R/W	Tuning code register	0x0580_1002

Bits	Description	
[31:27]	Reserved	Reserved
[26:24]	BIT_WIDTH	Decide n-bit binary search, n=BIT_WIDTH+1 Coarse tuning:3-bit Fine tuning:6-bit Precision tuning:8-bit If BIT_WIDTH = m, RC32K_P/RC32_F/RC32K_C[m:0] should be set to{1'b1, m{1'b0}.The other bits are decided by users. Reset_value:{0x5}
[23:16]	RC32K_P	Precision tuning code Reset_value:{8b1000_0000}
[15:14]	Reserved	Reserved
[13:8]	RC32K_F	Fine tuning code Reset_value:{6b01_0000}
[7:3]	Reserved	Reserved
[2:0]	RC32K_C	Coarse tuning code Reset_value:{3b010}

3.21.6.3 ClktrimCtlReg

Register	offset	R/W	Description	Reset Value
ClktrimCtlReg	CLKTRIM_BASE+0x08	R/W	Control register	0x0032_0000

Bits	Description	
[31:16]	ERR_RANGE	Decide ERROR range. ERROR = abs{REFCNT - IDEA_CNT} If ERROR < ERR_RANGE,binary search will be early terminated. Reset_value:{0x32}
[15:9]	Reserved	Reserved
[8]	EARLY_TERM_EN	1: Early termination binary search is enable 0: disable Reset_value:{1b0}
[7:1]	Reserved	Reserved
[0]	DECR	The relation ship between tuning code and frequency. 0:Increase relation 1:Decrease relation Reset_value:{0x0}

3.21.6.4 ClktrimIntReg

Register	offset	R/W	Description	Reset Value
ClktrimIntReg	CLKTRIM_BASE+0x0C	R/W	Interrupt flag register	0x0000_0000

Bits	Description	
[31:13]	Reserved	Reserved
[12]	HW_TRG_FLAG	Hardware trigger measurement stop flag 1: Hardware trigger measurement stop 0:No chagne This bit is cleared by writing 1 to it.
[11]	HW_TRG_INT	Hardware trigger measurement interrutpt 1:hardware measurement stop interrupt is trigger 0:no change Always 0 when interrupt is masked. This bit is cleared by writing 1 to it.
[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]	PTUNE_STOP_FLAG	Precision tuning stop flag 1: Clock measurement is stop 0: No change This bit is cleared by writing 1 to it.
[7]	PTUNE_STOP_INT	Precision 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.
[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.
[0]	INT_EN	1:Interrupt is enabled 0:Interrupt is masked

3.21.6.5 ClktrimCalCntReg

Register	offset	R/W	Description	Reset Value
ClktrimCalCntReg	CLKTRIM_BASE+0x10	R/W	Measurement period register	0x0400_0064

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}

3.21.6.6 ClktrimIdeaCntReg

Register	offset	R/W	Description	Reset Value
ClktrimIdeaCntReg	CLKTRIM_BASE+0x14	R/W	Ideal count register	0x0001_869F

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} //

3.21.6.7 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.

3.21.7 Software flow

3.21.7.1 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 section 3.2. 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.

3.21.7.2 Calibration

1. Select the clock to be calibrated and the reference clock;
2. Configure RC32K_P/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.

3.22 KEYSCAN

3.22.1 Overview

Keyscan supports key matrix with up to 24 rows by 8 columns. Each individual rows or columns can be enabled or disabled through register settings. GPIO pins can be configured to be used for keyscan. A few key scan parameters can be set through registers, including polarity (low or high indicating key pressed); scan interval; row interval and de-bounce time.

The keyscan has a polling mode and an interrupt mode. Keyscan will automatically scan the output/input pins, and store the row/column info corresponding to the key pressed into read only registers. For polling mode, software need to read register bit periodictity. When keyscan interrupt is enabled, a valid key press or release both can trigger an interrupt. After a keyscan interrupt is serviced, software can read related register bit to acquire the key state.

3.22.2 Features

- The size of the button can be configured, m*n (m=1-24, n=1-8), and the maximum support is 24*8.
- Support key debounce.
- The scan period can be configured;
- The button polarity is configurable, high-level active or low-level active;
- Two flag bits, the button state change flag bit, the wake-up flag bit under low power consumption;
- The read-only register keeps the key state.

3.22.3 Block Diagram

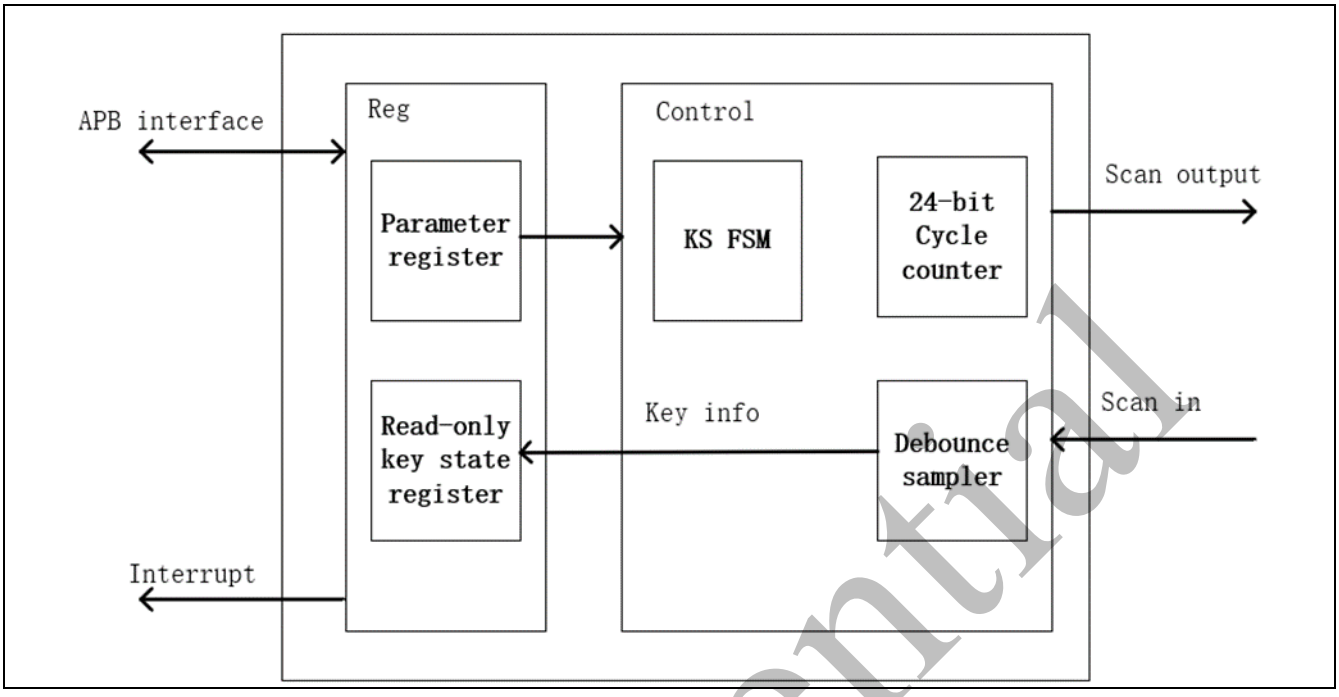


Figure 3-144 Keyscan Block Diagram

Reg module works in APB clock domain and has the following functions:

- Scan the parameter configuration and pass the parameters to the Control module.
- Save the button information sampled by the Control module to the read-only register.
- Generate an interrupt signal.

The Control module works in the sclk (sclk, scan clk) scan clock domain and has the following functions:

- Output the row scan signal.
- Sample the column input signal and pass the sampled key information to the Reg module.

3.22.3.1 Pins Description

System interface

Table 3-28 System Interface Description

Pins	I/O	Description
pclk	I	APB clock
presetn	I	APB clock domain reset signal
ksclk	I	CTL module working clock
ks_resetn	I	Ksclk clock domain reset signal
ks_int	O	Keyscan module interrupt signal

APB bus interface

Table 3-29 APB Bus Interface Description

Pins	I/O	Description
paddr[31:0]	I	APB address
pwrite	I	APB read and write enable signal 1: Write 0: Read
psel	I	APB select signal
penable	I	APB enable signal
pwdata[31:0]	I	APB write data line
prdata[31:0]	O	APB read data line

GPIO interface

Table 3-30 GPIO Interface Description

Pins	I/O	Description
ksi[7:0]	I	GPIO column input signal
kso[17:0]	O	GPIO row output signal
ksoe[17:0]	O	GPIO row output enable signal 1: GPIO output enable 0: GPIO output is invalid

3.22.4 Functional Description

3.22.4.1 Scanning Principle

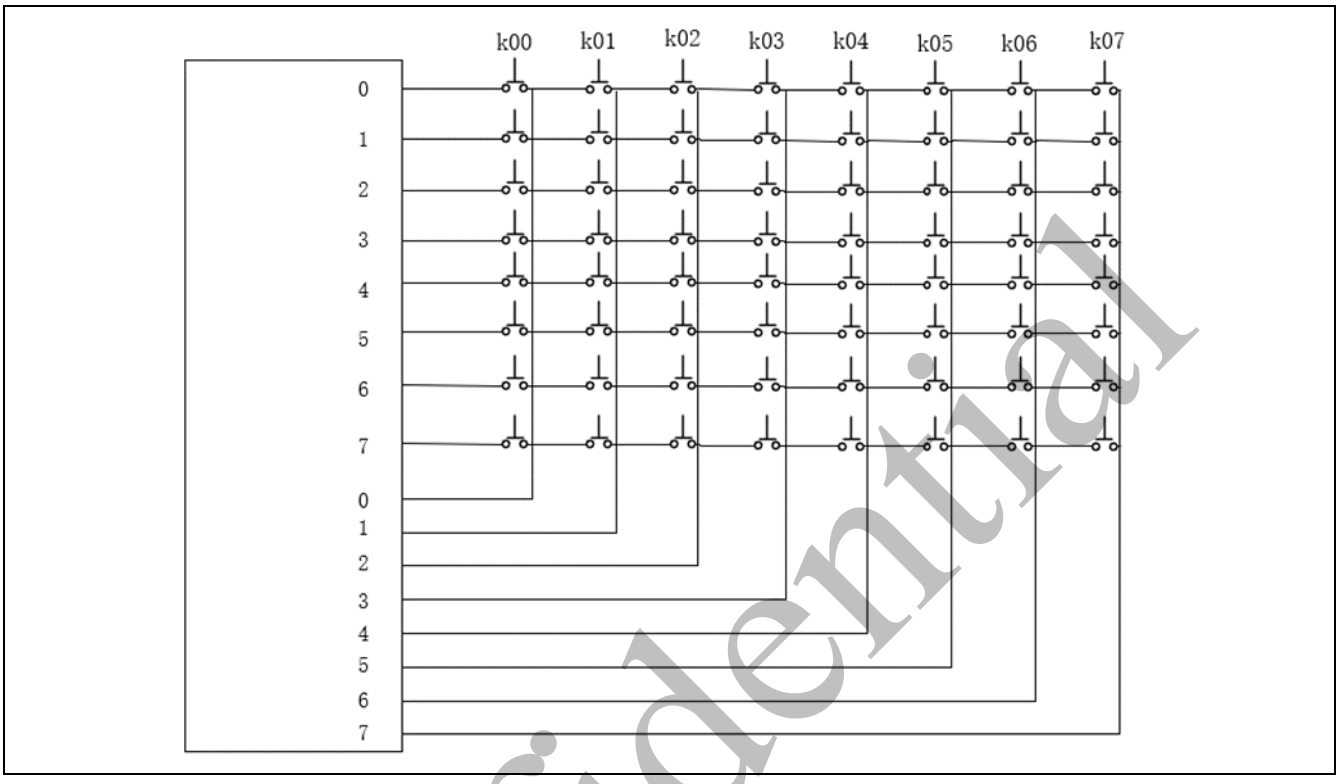


Figure 3-145 Scanning Principle

Take the 8*8 key matrix as an example, the scan signal is active at low level, and the column input defaults to high level.

When scanning row 0, row 0 outputs low level, and rows 1-7 output high level. When a key in row 0 is pressed, the corresponding column input signal changes from high to low, and the other column input signals remain high. For example, when k00 is on, the input signal in column 0 is low level, and the input signal in columns 1-7 is high level. Keyscan reads the column input signal, and saves the key state at this time to the corresponding register.

When scanning the first row, the first row outputs low level, and the remaining rows output high level. Keyscan reads the column input signal, and saves the key state at this time to the corresponding register.

By analogy, until all rows are scanned, the 8*8 key status is saved in the corresponding register.

The Keyscan module supports up to 24*8 key matrix, each key corresponds to a read-only register. You can scan only part of the row and part of the column through the configuration register.

The column input signal is read by keyscan after debounce, and the debounce time can be

configured by register.

After scanning one row, the time from scanning to the next row can be configured by register, this time is called row interval.

Scan all rows once, which is called a frame. The time from the end of scanning one frame to scanning the next frame can be configured by register, and this time is called frame interval.

The user can flexibly configure the row interval and frame interval according to the actual situation to achieve different scanning frequencies.

State Machine

There are five states in Keyscan: IDLE, INTERVAL, STEP, DEB, WAIT.

- **IDLE**. The keyscan_en signal is not turned on, and the keyscan is in the IDLE state. The software configuration register needs to be maintained in this state. After the keyscan_en signal is turned on, the state machine transitions from IDLE to INTERVA state.
- **INTERVAL**, frame interval status (scan interval). After scanning all rows once, it will enter the INTERVAL state. The next scan will be started after waiting for the scan interval.
- **STEP**, valid state of the row. In this state, the keyscan sets the output signal of a certain row to the valid state, and the output signals of the other rows are in the invalid state. And it detects the column input signal.
- **DEB**, debounce state. It happens after detecting a change in the button state. After the button state is maintained for the debounce time, the state information is written into the register and PAN108 series enters the WAIT state.
- **WAIT**, row interval status (row interval). Scan the next row after waiting for the row interval time.

Scan Mode

There are two ways to use the keyscan module: polling mode and interrupt mode.

In polling mode, after keyscan is enabled, the key status will be reflected in the register in real time. The software reads the value of the register regularly to check the button status.

In interrupt mode, the interrupt enable bit needs to be turned on. After scanning all the rows, if a change in the button state (button pressed or released) is detected, an interrupt is triggered and the enable signal is reset. The software needs to reset the enable signal and then start scanning again.

Keyscan supports up to 24*8 keys. No matter which scanning mode is used, and no matter how many keys are implemented, the hardware will scan 24 lines once, which is called a

frame. The scanning order is from row KSO[0] to row KSO[23].

Parameter Configuration

The keyscan module has two clock domains, the read and write registers are in the APB clock domain, and the scan control is in the sclk clock domain. The sclk clock is obtained by dividing the frequency of the RCC module by the abp clock. The frequency division coefficient range is [0, 1022].

f_sclk = f_ apbclk / rcc_clkdiv

The hardware scans all 24 rows every time, even if the realized key matrix is 1*1. Scanning all 24 lines is called a frame. The time for scanning a frame is called a frame period, and the reciprocal is called the frame frequency. The frame frequency should be at least twice the frequency of key presses. For example, the key press frequency is 25HZ, that is, the key is pressed 25 times in 1 second. Then the frame frequency is at least 50HZ, which means scanning 50 times per second.

The frame frequency is related to the sclk clock frequency, scan line interval, frame interval, debounce period, and how many rows of button states are changed within a frame. In the worst case, all 24 rows of keys change within a frame. At this time, the frame period can be roughly calculated with the following formula:

*Frame period = ((line interval + debounce period) * 24 + frame interval) * sclk clock period.*

The frame frequency should meet:

*Frame frequency=1/frame period> 2 * f_key_press*

It can be seen from the above formula that if the sclk clock frequency is relatively low, configure a small line interval, debounce period, and frame interval. If the sclk clock frequency is relatively high, the line interval, debounce period, and frame interval should be configured larger.

GPIO Allocation

GPIO has built-in pull-up input, pull-down input function.

KSO[17:0] corresponds to the row, which needs to configure the GPIO port mode as output mode

KSI[7:0] corresponds to the column, which need to configure the GPIO port as pull-up input or pull-down input

For example, to realize an 8*8 button matrix, you can use KSO[7:0] as the row and KSI[7:0] as the column by configuring the IoCfgReg register. You can also use KSO[3:0] KSO[10:7]

as the row. That is, any 8 rows can be selected from KSO[17:0].

GPIO		GPIO		GPIO		GPIO	
P00	Ks_o[6]	P10	Ks_o[4]	P20	Ks_i[0]	P30	Ks_i[4]
P01	Ks_o[7]	P11	Ks_o[5]	P21	Ks_i[1]	P31	Ks_i[5]
P02	Ks_i[0]	P12		P22	Ks_i[2]	P32	Ks_o[16]
P03	Ks_i[1]	P13		P23	Ks_i[3]	P33	Ks_o[17]
P04	Ks_o[10]	P14	Ks_i[4]	P24	Ks_o[8]	P34	Ks_o[18]
P05	Ks_o[11]	P15	Ks_i[5]	P25	Ks_o[9]	P35	Ks_o[19]
P06	Ks_o[12]	P16	Ks_i[6]	P26	Ks_o[14]	P36	Ks_o[20]
P07	Ks_o[13]	P17	Ks_i[7]	P27	Ks_o[15]	P37	Ks_o[21]/Ks_i[2]
P40	Ks_o[2]	P44		P50	Ks_o[0]	P54	Ks_o[23]
P41		P45	Ks_o[22]	P51	Ks_o[1]	P55	Ks_o[4]
P42		P46	Ks_o[0]	P52	Ks_o[2]	P56	Ks_o[11]
P43	Ks_o[15]	P47	Ks_o[1]	P53	Ks_o[3]	P57	Ks_o[5]

3.22.4.2 Configuration Process

IOMUX Configuration

According to the size of the key matrix to be implemented, the user can configure GPIO as the input and output of the keyscan.

For the GPIO that implements the keyscan output function, PUEN is disabled.

For the GPIO that implements the keyscan input function, enable PUEN.

Register configuration

IoCfgReg register (4001_9004)

Configure KSOE and KSIE, select row output and column output port

IntCfgReg register (4001_9008)

Disable or enable INT_EN, select polling mode or interrupt mode

KsCfgReg register (4001_900C)

Since the power consumption is related to the clock frequency, if the formula in section 2.6.1 is satisfied, the sclk clock frequency should be as low as possible. The following are the default values of some registers:

DEB_VALUE =0100. Represents 16 sclk clock cycles

RINTVAL_VALUE = 0000, which means 1 sclk clock cycle.

SINTVAL_VALUE=1000, which means 256 sclk clock cycles

Assuming the button state change frequency is 25HZ, and the above default configuration is used, at this time, the sclk clock cycle should meet:

$$T < 20ms / ((16+1)*24+256) = 35us$$

In other words, the sclk clock frequency should be greater than 28KHz. In actual use, the above registers should be flexibly configured according to the button jitter time and usage scenarios.

KsEnReg register (4001_9000)

Set KS_EN to 1, turn on keyboard scan.

Example: 2*2 Button Matrix

Turn on the APB clock and keyscan clock.

Configure IOMUX as follow:

1. IoCfgReg register (4001_9004), configured as 0x0300_0003
2. IntCfgReg register (4001_9008) is configured as 0x0000_0001,
3. KsEnReg register (4001_9000), configured as 0x0000_0001

Example: Wake Up in Low Power Consumption Mode

1. Configure row interval, frame interval, debounce time and other parameters.
2. Enable ks_en and wk_en.
3. Write 1 to clear the wkf register and wait for the system to enter the low-power mode.
4. Any button state change will wake up the system and trigger an interrupt after the system wakes up.
5. Execute the interrupt program and clear the wkf register.

3.22.5 Keyscan Regiser Map

Register	offset	R/W	Description	Reset Value
KeyScan base addr:0x4001_9000-0x4001_9FFF				
KSENREG	KS_BA+0x00	R/W	Keyscan enable register	0x0000_0000
IOCFGREG	KS_BA+0x04	R/W	Configurate keyscan IO register	0x0000_0000
INTCFGREG	KS_BA+0x08	R/W	Configurate interrupt register	0x0000_0000
KSCFGREG	KS_BA+0x0C	R/W	Configurate keyscan register	0x0000_0804
KSINFOREG0	KS_BA+0x10	R	Key information register0	0x0000_0000
KSINFOREG1	KS_BA+0x14	R	Key information register1	0x0000_0000
KSINFOREG2	KS_BA+0x18	R	Key information register2	0x0000_0000
KSINFOREG3	KS_BA+0x1C	R	Key information register3	0x0000_0000
KSINFOREG4	KS_BA+0x20	R	Key information register4	0x0000_0000
KSINFOREG5	KS_BA+0x24	R	Key information register5	0x0000_0000

3.22.6 Keyscan Register Description

3.22.6.1 KsEnReg

Register	offset	R/W	Description	Reset Value
KSENREG	KS_BA+0x00	R/W	Keyscan enable register	0x0000_0000

Bits	Description	
[31:1]	Reserved	Reserved
[1]	WK_EN	1: Wake up function is enabled 0: Wake up function is disabled Reset value: 0x0
[0]	KS_EN	1: Keyscan is enabled 0: Keyscan is disabled If int_en is set, ks_en will be cleared while detecting interrupt; If wk_en is set, ks_en can not be cleared by hardware Reset value: 0x0

3.22.6.2 IoCfgReg

Register	offset	R/W	Description	Reset Value
IOCFGREG	KS_BA+0x04	R/W	Configurate keyscan IO register	0x0000_0000

Bits	Description	
[31:8]	KSOE	Keyscan output configuration 1: KSO enabled 0: KSO disabled，keyscan does not scan this row Reset_value: 0x0
[7:0]	KSIE	Keyscan input configuration 1: KSI enabled 0: KSI disabled,keyscan does not check this column Reset_value: 0x0

3.22.6.3 IntCfgReg

Register	offset	R/W	Description	Reset Value
INTCFGREG	KS_BA+0x08	R/W	Configurate interrupt register	0x0000_0000

Bits	Description	
[31:3]	Reserved	Reserved
[3]	WKF	Keyscan wakeup flag This bit is cleared by writing 1 to it. This bit is valid only when WK_EN is set. Reset_value: 0x0
[2]	KSIF	Keyscan key_change interrupt Flag. 1: key_change occurred 0: no change This bit is cleared by writing 1 to it. This bit is valid only when INT_EN is set. Reset_value: 0x0
[1]	KSF	Keyscan key_change flag 1: key_change occurred 0: key_change did not occur This bit is cleared by writing 1 to it. Reset_value: 0x0
[0]	INT_EN	Keyscan key_change interrupt Enable bit 1: key_change Interrupt enabled 0: key_change Interrupt disabled INT_EN =1, KSIF=KSF; INT_EN = 0, KSIF=0 Reset_value: 0x0

3.22.6.4 KsCfgReg

Register	offset	R/W	Description	Reset Value
KSCFGREG	KS_BA+0x0C	R/W	Configurate keyscan register	0x0000_0804

Bits	Description	
[31:14]	Reserved	Reserved
[13]	POL	Configure the valid level 1: high valid 0: low valid Reset value = 0x0
[12]	CLR_KEY	Clear all key status register 1: clear enabled 0: no change This bit will be cleared automatically after writing 1 to it Reset value: 0x0
[11:8]	SINTVAL_VALUE	Determine the interval time between the last row and the first row of next scan. Period = 2 ^ SINTVAL_VALUE + 1(sclk) Eg. SINTVAL_VALUE = 1000, period = 2 ^ 8 + 1 =257(sclk) Reset value: {4b1000}
[7:4]	RINTVAL_VALUE	Determine period between the current row and the next row. Period = RINTVAL_VALUE + 1(sclk) Eg. RINTVAL_VALUE = 0000, period = 1 (sclk) Reset value: {4b0000}
[3:0]	DEB_VALUE	Debounce time. Only the keys state hold larger or equal than debounce time can they be written to registers. Debounce time = 2 ^ DEB_VALUE - 1(sclk) (DEB_VALUE >=1) If DEB_VALUE is set to 0,it means debounce time is 2 sclk. Eg. DEB_VALUE = 0100, debounce time = 2 ^ 4 - 1= 15 (sclk) Reset value: {4b0100}

3.22.6.5 KsInfoReg0

Register	offset	R/W	Description	Reset Value
KSINFOREG0	KS_BA+0x10	R	Key information register0	0x0000_0000

Bits	Description	
[31:24]	ROW3	ROW3[x]—ROW3_COLx x=0-7 1: key is pressed 0: key is released
[23:16]	ROW2	ROW2[x]—ROW2_COLx x=0-7 1: key is pressed 0: key is released
[15:8]	ROW1	ROW1[x]—ROW1_COLx x=0-7 1: key is pressed 0: key is released
[7:0]	ROW0	ROW0[x]—ROW0_COLx x=0-7 1: key is pressed 0: key is released

3.22.6.6 KsInfoReg1

Register	offset	R/W	Description	Reset Value
KSINFOREG1	KS_BA+0x14	R	Key information register1	0x0000_0000

Bits	Description	
[31:24]	ROW7	ROW7[x]—ROW7_COLx x=0-7 1: key is pressed 0: key is released
[23:16]	ROW6	ROW6[x]—ROW6_COLx x=0-7 1: key is pressed 0: key is released
[15:8]	ROW5	ROW5[x]—ROW5_COLx x=0-7 1: key is pressed 0: key is released
[7:0]	ROW4	ROW4[x]—ROW4_COLx x=0-7 1: key is pressed 0: key is released

3.22.6.7 KsInfoReg2

Register	offset	R/W	Description	Reset Value
KSINFOREG2	KS_BA+0x18	R	Key information register2	0x0000_0000

Bits	Description	
[31:24]	ROW11	ROW11[x]—ROW11_COLx x=0-7 1: key is pressed 0: key is released
[23:16]	ROW10	ROW10[x]—ROW 10_COLx x=0-7 1: key is pressed 0: key is released
[15:8]	ROW9	ROW9[x]—ROW9_COLx x=0-7 1: key is pressed 0: key is released
[7:0]	ROW8	ROW8[x]—ROW8_COLx x=0-7 1: key is pressed 0: key is released

3.22.6.8 KsInfoReg3

Register	offset	R/W	Description	Reset Value
KSINFOREG3	KS_BA+0x1C	R	Key information register3	0x0000_0000

Bits	Description	
[31:24]	ROW15	ROW15[x]—ROW15_COLx x=0-7 1: key is pressed 0: key is released
[23:16]	ROW14	ROW14[x]—ROW14_COLx x=0-7 1: key is pressed 0: key is released
[15:8]	ROW13	ROW13[x]—ROW13_COLx x=0-7 1: key is pressed 0: key is released
[7:0]	ROW12	ROW12[x]—ROW12_COLx x=0-7 1: key is pressed 0: key is released

3.22.6.9 KsInfoReg4

Register	offset	R/W	Description	Reset Value
KSINFOREG4	KS_BA+0x20	R	Key information register4	0x0000_0000

Bits	Description	
[31:24]	ROW19	ROW19[x]—ROW19_COLx x=0-7 1: key is pressed 0: key is released
[23:16]	ROW18	ROW18[x]—ROW18_COLx x=0-7 1: key is pressed 0: key is released
[15:8]	ROW17	ROW17[x]—ROW17_COLx x=0-7 1: key is pressed 0: key is released
[7:0]	ROW16	ROW16[x]—ROW16_COLx x=0-7 1: key is pressed 0: key is released

3.22.6.10 KsInfoReg5

Register	offset	R/W	Description	Reset Value
KSINFOREG5	KS_BA+0x24	R	Key information register5	0x0000_0000

Bits	Description	
[31:24]	ROW23	ROW23[x]—ROW23_COLx x=0-7 1: key is pressed 0: key is released
[23:16]	ROW22	ROW22[x]—ROW22_COLx x=0-7 1: key is pressed 0: key is released
[15:8]	ROW21	ROW21[x]—ROW21_COLx x=0-7 1: key is pressed 0: key is released
[7:0]	ROW20	ROW20[x]—ROW20_COLx x=0-7 1: key is pressed 0: key is released

3.23 Electronic Codebook Mode Encryption

3.23.1 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).

3.23.2 Features

- 128 bit AES encryption
- Supports standard AES ECB block encryption and decryption
- Memory pointer support

3.23.3 Block Diagram

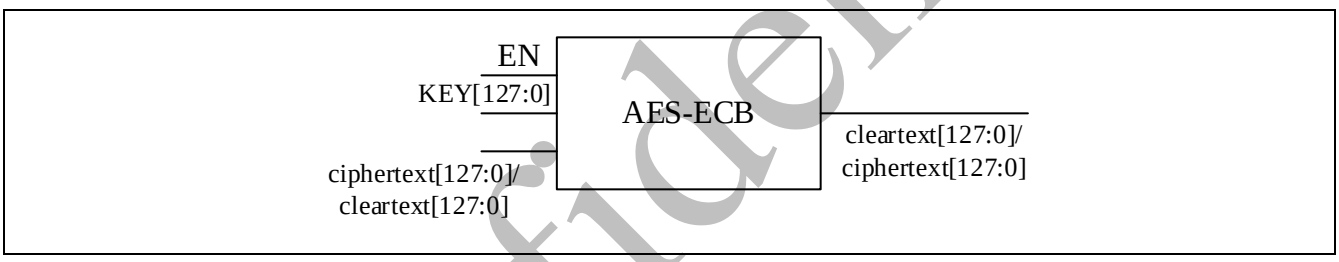


Figure 3-146 AES-ECB Diagram

3.23.4 Functional Description

AES ECB access to Link Layer Data RAM for in-place operations on cleartext and ciphertext during encryption or decryption.

3.23.5 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

3.23.6 AES-ECB Register Description

3.23.6.1 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

3.23.6.2 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.

3.23.6.3 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.

3.23.6.4 SECURE4

Register	Offset	R/W	Description	Reset Value
SECURE4	0x478	R/W		0x0000_0000

Bits	Description	
[31:16]	SECURE4_SEC_KEY2_RX_ADDR	This field specifies the key pointer for packet 1.
[15:0]	SECURE4_SEC_KEY1_RX_ADDR	This field specifies the key pointer for packet 0.

3.23.6.5 SECURE5

Register	Offset	R/W	Description	Reset Value
SECURE5	0x47C	R/W		0x0000_0000

Bits	Description	
[31:16]	SECURE5_SEC_KEY4_RX_ADDR	This field specifies the key pointer for packet 3.
[15:0]	SECURE5_SEC_KEY3_RX_ADDR	This field specifies the key pointer for packet 2.

3.23.6.6 SECURE6

Register	Offset	R/W	Description	Reset Value
SECURE6	0x480	R/W		0x0000_0000

Bits	Description	
[31:16]	SECURE6_SEC_KEY6_RX_ADDR	This field specifies the key pointer for packet 5.
[15:0]	SECURE6_SEC_KEY5_RX_ADDR	This field specifies the key pointer for packet 4.

3.23.6.7 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_RX_ADDR	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).

3.23.6.8 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).

3.23.6.9 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).

3.23.6.10 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).

3.24 Random Number Generators

3.24.1 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.

3.24.2 Features

- 32-bit random number
- Support scramble function

3.24.3 Block Diagram

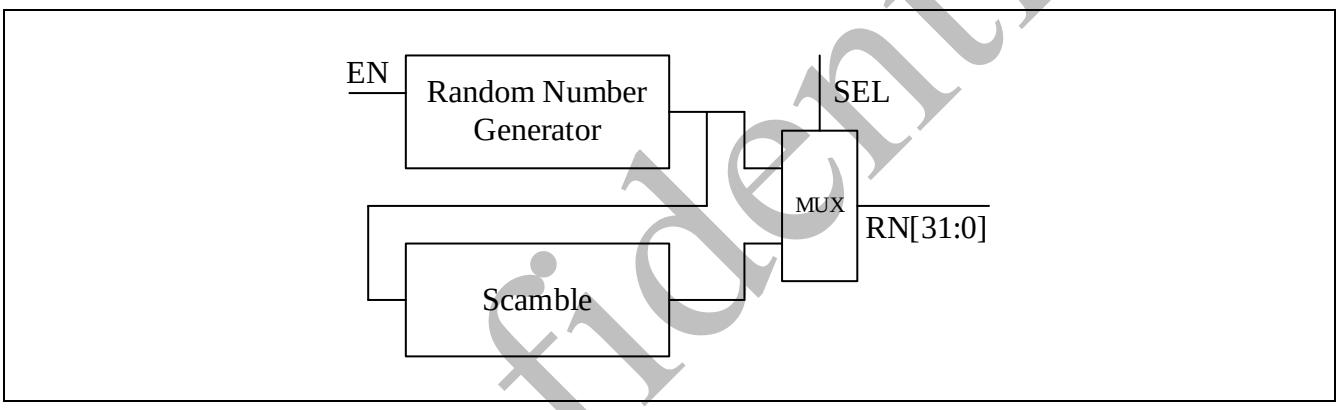


Figure 3-147 RNG Block Diagram

3.24.4 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.

3.24.5 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

3.24.6 RNG Register Description

3.24.6.1 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.

3.24.6.2 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.

3.24.6.3 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.

3.24.6.4 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.

3.24.6.5 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.

3.25 Real Time Counter

3.25.1 Overview

The Real time counter (RTC) module provides a generic, low power timer on the low-frequency clock source.

3.25.2 Features

- 32-bit counter
- Support compare function
- Support interrupt
- Support 2 clock source 32kHz RC or 32.768kHz XTAL

3.25.3 Block Diagram

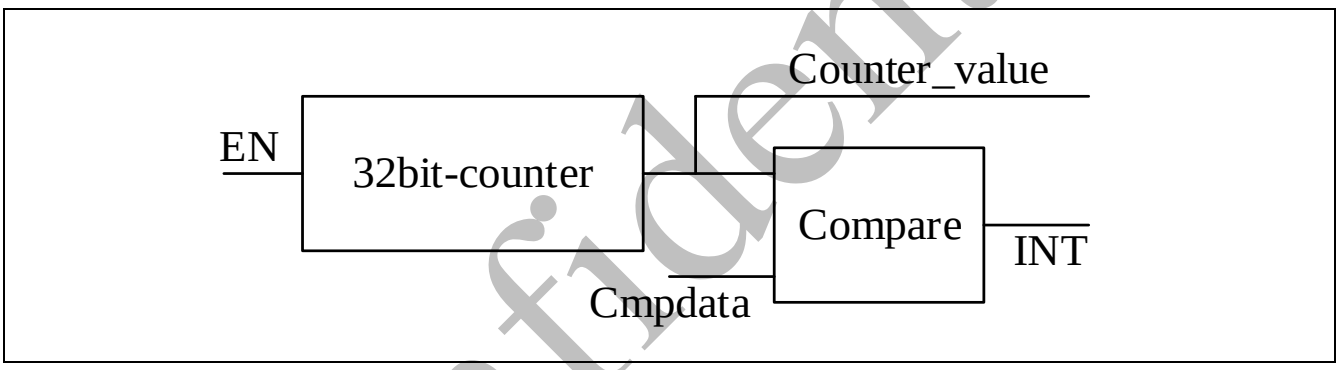


Figure 3-148 RTC Block Diagram

3.25.4 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 compare function, compare value is set by software, when counter value counts to compare value, a interrupt is generated.

3.25.5 RTC 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				
SLPTMR1	0x000C	R/W		0x0000_0004
SLPTMR3	0x0014	R/W		0x0000_0000
SLPTMR4	0x0018	R/W		0x0000_0000

3.25.6 RTC Register Description

3.25.6.1 SLPTMR1

Register	Offset	R/W	Description	Reset Value
LPTMR1	0x000C	R/W		0x0000_0004

Bits	Description	
[31:10]	Reserved	Reserved.
[9]	SLPTMR1_SLPTMR_DIN_ERR_FLG	This flag is generated for wrong tmr_din configuration i.e. when the slptmr_din is lower than the tmr current value.
[8]	SLPTMR1_SLPTMR_DIN_ERR_FLG_MASK	The mask signal of the din_error flag IRQ - 1'b0: no IRQ is generated - 1'b1: IRQ is generated
[7]	SLPTMR1_SLPTMR_DIN_ERR_FLG_CLR	The clear signal of the din_error flag.
[6]	SLPTMR1_SLPTMR_CMPR_BUS_ENA	This field specifies the control compare unit 2 status function. - 1'b0: Disable - 1'b1: Enable
[5]	SLPTMR1_SLPTMR_CMPR_BUS_IRQ_MASK	This field specifies the mask signal for the compare unit 2 interrupt.
[4]	SLPTMR1_SLPTMR_CMPR_BUS_IRQ_CLR	This field specifies the Compare unit 2 interrupt clear.
[3]	SLPTMR1_SLPTMR_CMPR_BUS_IRQ	This field specifies the compare unit 2 interrupt status.
[2]	Reserved	Reserved.
[1]	SLPTMR1_SLPTMR_ENA	This field specifies the Enable sleep timer.
[0]	Reserved	Reserved.

3.25.6.2 SLPTMR3

Register	Offset	R/W	Description	Reset Value
SLPTMR3	0x0014	R/W		0x0000_0000

Bits	Description	
[31:0]	SLPTMR3_SLPTMR_CURRENT	This field specifies the sleep timer current value.

3.25.6.3 SLPTMR4

Register	Offset	R/W	Description	Reset Value
SLPTMR4	0x0018	R/W		0x0000_0000

Bits	Description	
[31:0]	SLPTMR4_SLPTMR_CMPR_BUS_DIN	This field specifies the compare unit 2 input (compare values).

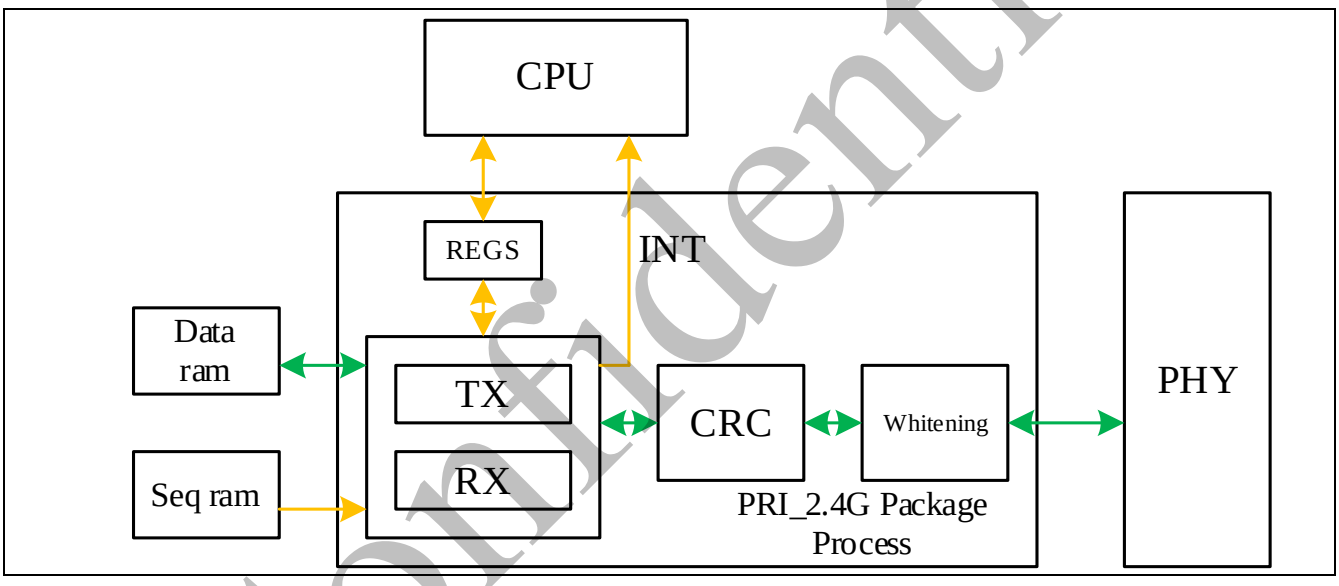
4 2.4 GHz proprietary protocols

4.1 System features

- Support 1M and 2M PHY
- XN297L,PAN1026 Transceiver protocol compliant
- Support No Acknowledge, Acknowledge and Acknowledge with payload
- Support CRC8, CRC16 and CRC24
- Support whitening

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:

Preamble (3Bytes)	Addr (3-5Bytes)	Payload (1-64Bytes)	Crc24/crc16/crc8 (3/2/1Bytes)

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：

Preamble (3Bytes)	Addr (3-5Bytes)	Signal (10bits)	Payload (0-64Bytes)	Crc24/crc16/crc8 (3/2/1Bytes)

Where Signal includes：

Length (7bits)	PID (2bits)	NOACK (1bit)

Preamble: 3bytes in total, 0x710F55, sent from the high bit.

Addr: 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,pri_r00[15]), Signal(Normal mode without this segment), Payload and Crc24. Initial values are configurable.

CRC: Addr(Optional,pri_r00[15]), 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

The normal mode is compatible with nrf normal mode.

Preamble (1Bytes)	Addr (3-5Bytes)	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 Addr and Length can be configured, and the CRC can choose

crc24, crc16 and 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 normal_m1 mode is compatible with Bluetooth frame structure mode (CRC calculation range does not include the Addr).

The frame structure is the same as the normal mode frame structure

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

The enhanced mode is compatible with nrf_enh0 mode and some nrf_enh1mode(CRC calculation range does not include the Addr).

Preamble (1Byte)	Addr (3-5Bytes)	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 first bit of addr is 1, preamble is 10101010, otherwise it is 01010101. It is sent from the high bit.

Addr: Configured by the register, variable length (3-5bytes).

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：0~255bytes when length is 8-bits; 0-63bytes when length is 6-bits

Crc: Range(Addr+)Header+Length+Signal+Payload

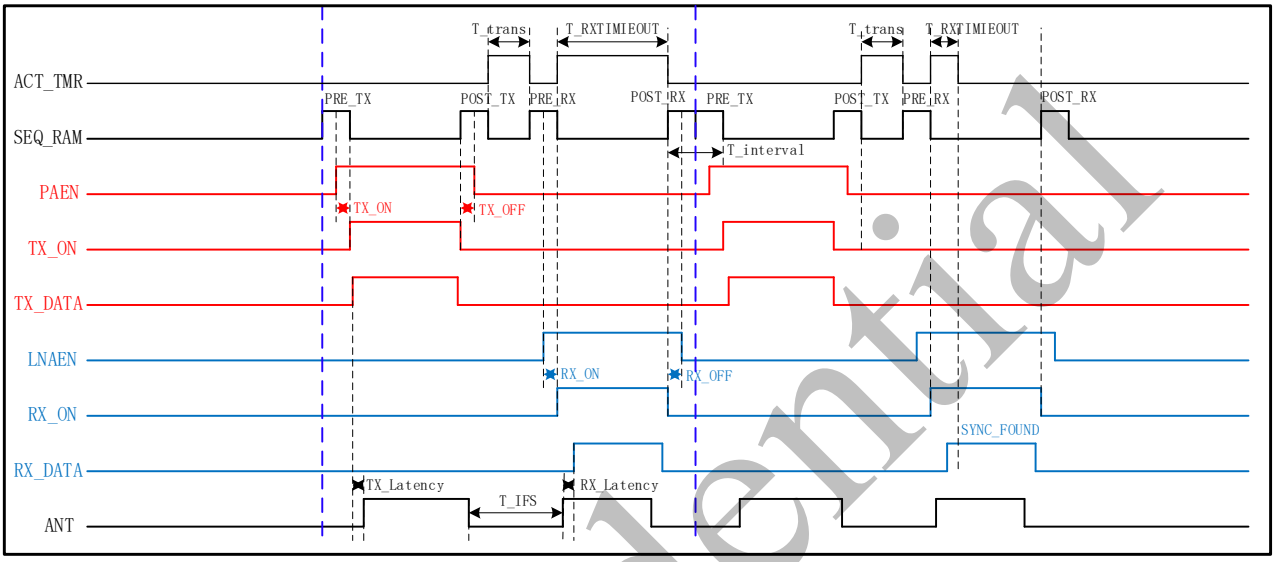
Accaddr_crcscr_dis configuration determines whether the Addr field is included or not. Accaddr_crcscr_dis=1: not included; Accaddr_crcscr_dis=0: included

Note: The nrf_enh1 mode does not support the inclusion of the Addr field.

Whitenning range: Addr (Optional, configured by Accaddr_crcscr_dis), Signal(Normal mode without this segment), Header, Length, Payload and Crc. Initial values are configurable.

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

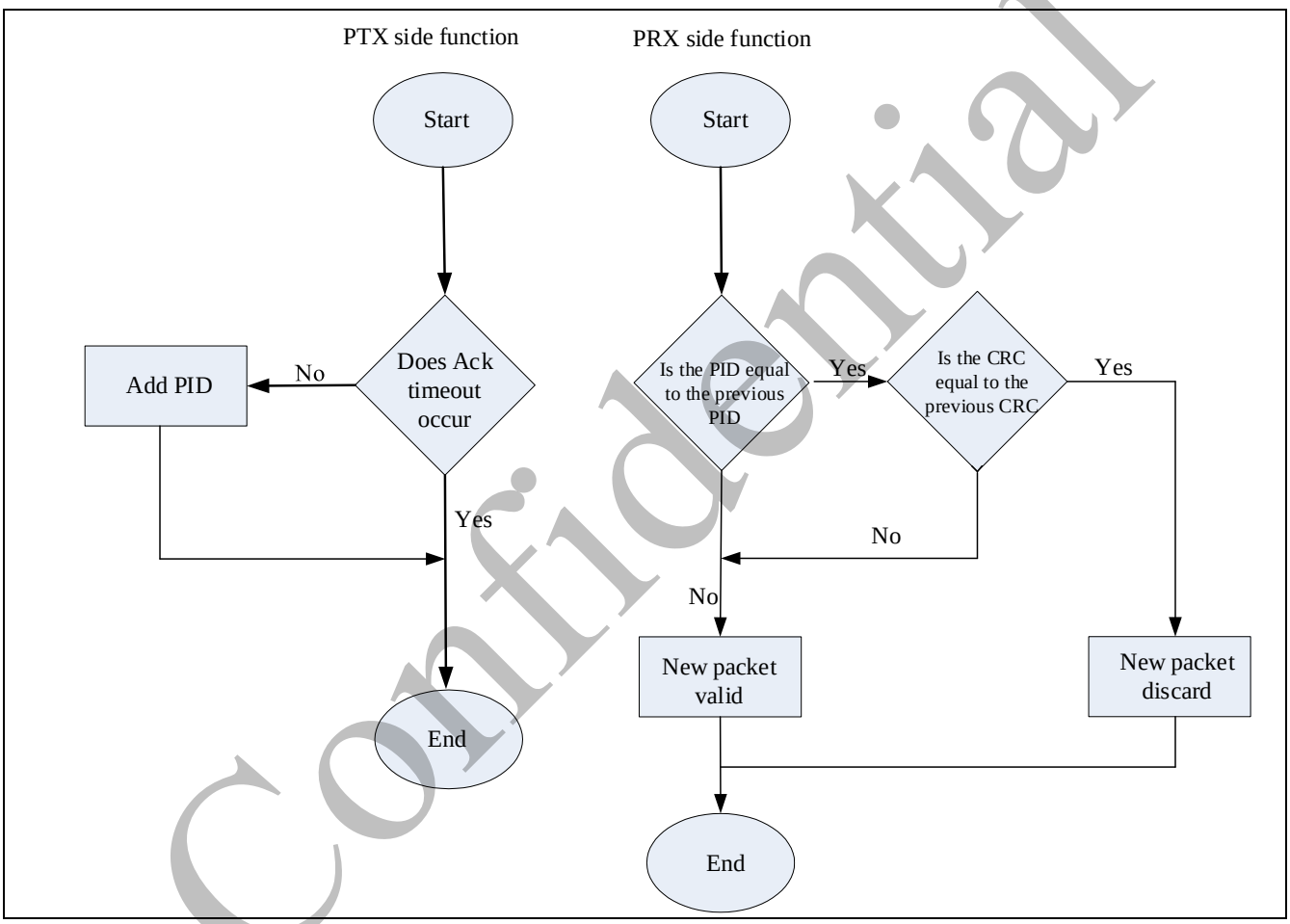


Figure 4-1 PID generation and detection

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_CRCSCR_DIS	0：crc will include access address，header，payload scr will include access address，header，payload，crc 1：crc will just include header，payload scr will include header，payload，crc
[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: invalid 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 not enabled
[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: Enabled 0: Off
[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） 01: ble 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
[26:25]	RX_PID_MANUAL	rx pid configured externally by user
[24:20]	Reserved	Reserved
[21]	ENDIAN	Active @nrf mode 0: little endian 1: big endian Note：when Nrf and xn297 mode should set endian to 1
[20]	TX_DONE_IRQ_EN	Configured to 0, no tx_done_irq is generated
[19]	RX_DONE_IRQ_EN	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]	PID_MANUAL_EN	Configured to 1, allows user external configuration of tx_pid_manual and rx_pid_manual
[1:0]	TX_PID_MANUAL	tx pid configured externally by user

4.6.3 PRI_R02

Register	Offset	R/W	Description	Reset Value
pri_r02	0x4c8	R/W		0x0

Bits	Descriptions	
[31]	Reserved	Reserved
[30:16]	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]	RX_ADDR[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
[15:12]	Reserved	Reserved
[11]	NORMAL_M1	Normal_m1 mode, cannot be used in conjunction with enhance mode 0: turned off 1: turned on, 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]	RX_ADDR[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]	TX_ADDR[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]	Reserved	Reserved
[14:8]	WHITENING_INIT	The initial value of whitening(default: 0x7f)
[7:0]	TX_ADDR[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

4.7 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: 1,3bytes; 2, 4bytes; 3 5bytes

f. Configure the device address high byte PRI_R06_TX_ADDR_MSB, only used in 5bytes address width

g. Configure the device address low bytes PRI_R05_TX_ADDR_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: 1,3bytes;2, 4bytes; 3 5bytes
   e. Configure the device address high byte PRI_R04_RX_ADDR_MSB, only used in 5bytes address width
   f. Configure the device address low bytes PRI_R03_RX_ADDR_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_RX_ADDR_MSB, only used in 5bytes address.

n. Configure the peer device address low bytes PRI_R03_RX_ADDR_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_TX_ADDR_MSB, only used in 5bytes address.

n. Configure the peer device address low bytes PRI_R05_TX_ADDR_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_RX_ADDR_MSB, only used in 5bytes address

m. Configure the peer device address low bytes PRI_R03_RX_ADDR_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_TX_ADDR_MSB, only used in 5bytes address

m. Configure the peer device address low bytes PRI_R05_TX_ADDR_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 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.

4.7.4.1 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.

4.7.4.2 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 oscillator
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.2021	Initial
1.1	Aug.2022	Update
1.2	Sep.2022	Optimized pin description
1.3	Nov.2022	Update some pin names and delete INT0/INT1/INT2 pins; Added QFN48 package.
1.4	Dec.2022	Updated layout, updated parameters of 'Low Power' and 'ESD' in Key Features.
1.5	Feb.2023	Updated ESD parameters
1.6	Mar.2023	Updated the description of "STANDBY_M1" in Table 3-8; updated the Abbreviation.
1.7	Jun.2023	Updated GPIO Register Map GPIO Base Address GP_BA to 0x4002_0000, updated the BLE5.1 to BLE5.3, updated the PDEN[n] bit in 3.9.8.2, delete ANT+
1.8	Sep.2023	Updated the operating voltage and some errors, removed the BLE-REGS.
1.9	Apr.2024	Added a register description at SysTick Control and Status Register (SYST_CTRL).

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